PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : C12Q 1/68

A2

(11) International Publication Number: WO 00/40758

(43) International Publication Date: 13 July 2000 (13.07.00)

(21) International Application Number: PCT/US00/00458

(22) International Filing Date: 6 January 2000 (06.01.00)

(30) Priority Data:
60/115,284 6 January 1999 (06.01.99) US

(71) Applicant: HYSEQ INC. [US/US]; 670 Almanor Avenue, Sunnyvale, CA 94086 (US).

(72) Inventors: DRMANAC, Radoje; 850 East Greenwich Place, Palo Alto, CA 94303 (US). DRMANAC, Snezana; 850 East Greenwich Place, Palo Alto, CA 94303 (US). KITA, David; 899 Bounty Drive, #204, Foster City, CA 94404 (US). COOKE, Cory; 400 East Poplar #1, San Mateo, CA 94401 (US). XU, Chongjun; 4918 Manitoba Drive, San Jose, CA 95130 (US).

(74) Agent: WILLIAMS, Joseph, A., Jr.; Marshall, O'Toole, Gerstein, Murray & Borun, 6300 Sears Tower, 233 South Wacker Drive, Chicago, IL 60606 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: ENHANCED SEQUENCING BY HYBRIDIZATION USING POOLS OF PROBES

(57) Abstract

The invention provides methods for sequencing by hybridization (SBH) using pools of probes that allow greater efficiency in conducting SBH by reducing the number of separate measurements of hybridization signals required to identify each particular nucleotide in a target nucleic acid sequence. The invention also provides pools and sets of pools of probes, as well as methods of generating pools of probes.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece	**ML**	Mali	**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**MN**	Mongolia	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MR**	Mauritania	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MW**	Malawi	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MX**	Mexico	**US**	United States of America
CA	Canada	**IT**	Italy	**NE**	Niger	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NL**	Netherlands	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NO**	Norway	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NZ**	New Zealand	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**PL**	Poland		
CM	Cameroon	**KR**	Republic of Korea	**PT**	Portugal		
CN	China	**KZ**	Kazakstan	**RO**	Romania		
CU	Cuba	**LC**	Saint Lucia	**RU**	Russian Federation		
CZ	Czech Republic	**LI**	Liechtenstein	**SD**	Sudan		
DE	Germany	**LK**	Sri Lanka	**SE**	Sweden		
DK	Denmark	**LR**	Liberia	**SG**	Singapore		
EE	Estonia						

ENHANCED SEQUENCING BY HYBRIDIZATION USING POOLS OF PROBES

This application claims priority of U.S. provisional application no. 60/115,284 filed January 6, 1999, the disclosure of which is incorporated herein by reference.

FIELD OF THE INVENTION

This invention relates in general to methods and apparatus for nucleic acid sequence analysis, in particular sequence analysis using sequencing by hybridization.

BACKGROUND

The rate of determining the sequence of the four nucleotides in nucleic acid samples is a major technical obstacle for further advancement of molecular biology, medicine, and biotechnology. Nucleic acid sequencing methods which involve separation of nucleic acid molecules in a gel have been in use since 1978.

The traditional method of determining a sequence of nucleotides (i.e., the order of the A, G, C and T nucleotides in a sample) is performed by preparing a mixture of randomly-terminated, differentially labelled nucleic acid fragments by degradation at specific nucleotides, or by dideoxy chain termination of replicating strands. Resulting nucleic acid fragments in the range of 1 to 500 bp are then separated on a gel to produce a ladder of bands wherein the adjacent samples differ in length by one nucleotide.

The present invention relates to an alternative methodology for sequencing a target nucleic acid known as sequencing by hybridization (SBH). The array-based approach of SBH does not require single base resolution in separation, degradation, synthesis or imaging of a nucleic acid molecule. Using mismatch discriminative hybridization of short oligonucleotides K nucleotides in length, lists of constituent k-mer oligonucleotides may be determined for target nucleic acid. Sequence for the target nucleic acid may be assembled by uniquely overlapping scored oligonucleotides.

Nucleic acid sequencing by hybridization shares interesting parallels with conducting a computer search of a text file for a particular word or a phrase. In each case, a large string of characters is probed with a specific shorter string to detect matching sequences. In a computer text search, the search string or strings (key words) are used to browse a large Internet or local data base to identify the subset of specific documents containing perfect sequence matches, which is then retrieved for further review or analysis. In SBH, oligonucleotide probes ranging from 4 to 25 characters in length are used to browse libraries of nucleic acid segments to identify nucleic acid molecules containing exact complementary sequences. These molecules may then be further analyzed by mapping or clustering, or by partial or full sequencing.

In the case of a hybridization search of four simple DNA samples with four different 5-mer probes (which could be called key words, or strings), each sample binds a different combination of probes, leading to a characteristic hybridization pattern. Each positive binding (or hybridization) event in a given DNA sample provides a discrete piece of information about its sequence. Neither the frequency nor location of the string within the DNA molecule is obtained from a hybridization search, as is also the case in most computer text searches. For example, a positive search result for the word "tag" in a set of document titles does not identify whether the word is positioned at the beginning, middle, or end of the selected titles, nor whether it occurs once, twice, or many times in any of these titles. Similarly, the entire DNA is sampled by random probe-binding trials, without determination of exactly where in the chain particular probes bind.

In a computer search of English language text, the complexity of the English alphabet (26 letters) generally allows a meaningful search of a given text to be done with one or a few specific words. With a DNA search, the simple four-letter genetic alphabet requires use of either more or longer "words" (strings) to precisely identify a specific DNA. A simple word like "cat" might yield useful results in a computer search of the Internet, but the genetic triplet "CAT" occurs far too frequently (about once in every sixty-four triplets) to be of much use in DNA identification. The lengths of the DNA string (sequence) and the probe (interrogating string) are important parameters in devising a successful SBH experiment. By choosing appropriate probe and sample lengths, a researcher can obtain useful sequence data.

The first potential probe binding site in a nucleotide sequence chain starts at the first base and extends for the length of the probe. The second probe binding site starts at the second base and overlaps the first probe binding site, less one base. This means that if a complete (or sufficient) set of probes is tested, the end of each positive probe overlaps with the beginning of another positive probe, except in the case of the last positive probe in the target. In each sequence assembly cycle, four potential overlap probes are checked. Starting with a positive probe AAATC, the next positive overlapping probe to the right may be AATCA, AATCC, AATCG or AATCT. Of these probes, only AATCG is found to be positive and is used for further assembly. The cycles are repeated in both directions until all positive probes are incorporated and the complete sequence is assembled. By extension, the same process applies to a longer target nucleic acid if enough probes of appropriate length are used to identify uniquely overlapped strings within it.

The use of overlapping positive probes is a key aspect of SBH methods. This "overlap principle" allows the identification of sequences within a target DNA that are longer than any of the probes used in the assembly process. Probe overlap allows indirect assignment of one out of four bases for each position in the analyzed DNA chain without performing any actual positional measurements on the sample. The base/position information is in fact derived from the known sequences of the oligonucleotide probes obtained by accurate chemical synthesis.

Thus, a DNA hybridization search is effectively a highly parallel molecular computation process with fully random access to the "input data," in this case a polynucleotide chain that may be thousands of bases long. These fundamental characteristics of the SBH process confer unique opportunities for miniaturization and parallel analyses, leading to speed and cost efficiencies not available with other sequencing methods.

Because the sequences of DNA molecules are non-random and irregular, statistical artifacts arise that must be addressed in SBH experiments. Even when the lengths of DNA targets and probes are selected to achieve a statistical expectation that each probe sequence occurs no more than once in the target, so-called "branching ambiguities" can occur. (Drmanac et al., Yugoslav Patent Application 570/87 (1987) issued as U.S. Patent 5,202,231 (1993); Drmanac et al., "Sequencing of Megabase Plus DNA by Hybridization: Theory of the Method," *Genomics*, 4:114-128 (1989).) Take the case of three probes that positively hybridize to a target DNA: TAGA, AGAC and AGAT.

Both the second and the third probes overlap with the first probe, sharing the bases AGA and giving extended sequences TAGAC and TAGAT, respectively. Due to the occurrence of the sequence AGA in both the second and third probes (e.g. due to double AGA occurrence in the target), there is not enough information available to decide which of the two probes is actually the one that overlaps with the first probe in the sample. Sequence assembly can thus proceed along either of the two branches, only one of which may be correct. Branching ambiguities may be resolved if a reference sequence for the target is known.

By using all possible probes of a given length, a researcher can unambiguously determine a target nucleotide sequence, provided the target nucleic acid is short enough that most overlap sequences occur no more than once. The only other exception to this rule is tandem repeat regions (e.g.: AAAAAAAAAA, ACACACACAC) that are longer than the probe length. In such cases, the exact length of these repeats may be determined by use of a special subset of longer probes. Longer targets may require longer probes for unambiguous sequence determination. A variety of ways have been proposed to increase the read length with a given set of probes, or to reduce the number of experimental probe/target scores needed to sequence a target nucleic acid. These include the use of redundant combinations of probes, competitive hybridization and overlapped clones (Drmanac et al., Yugoslav Patent Application 570/87 (1987) issued as U.S. Patent 5,202,231 (1993); Drmanac et al., "Sequencing of Megabase Plus DNA by Hybridization: Theory of the Method," *Genomics*, 4:114-128 (1989)), gapped probes (Bains et al., "A Novel Method for Nucleic Acid Sequencing," *J. Theor. Biol.*, 135:303-307 (1988)) and binary probes (Pevzner et al., "Towards DNA Sequencing Chips," *Mathematical Foundations of Computer Science 1994* (Eds. I. Privara, B. Rovan, P. Ruzicka,) pp. 143-158, The Proceedings of 19th International Symposium, MFCS '94, Kosice, Slovakia, Springer-Verlag, Berlin (1995)), continuous stacking hybridization (Khrapko et al., "An Oligonucleotide Hybridization Approach to DNA Sequencing," *FEBS Letters*, 256:118-122 (1989), and the simultaneous sequencing of similar genomes (Drmanac et al., "Sequencing by Hybridization (SBH) With Oligonucleotide Probes as an Integral Approach for the Analysis of Complex Genomes," *International Journal of Genomic Research*, 1(1): 59-79 (1992).

There are several approaches available to achieve sequencing by hybridization. In a process called SBH Format 1, nucleic acid samples are arrayed, and labeled probes are hybridized with the samples. Replica membranes with the same sets of sample nucleic acids may be used for parallel scoring of several probes and/or probes may be multiplexed (i.e., probes containing different labels). Nucleic acid samples may be arrayed and hybridized on nylon membranes or other suitable supports. Each membrane array may be reused many times. Format 1 is especially efficient for batch processing large numbers of samples.

In SBH Format 2, probes are arrayed at locations on a substrate which correspond to their respective sequences, and a labelled nucleic acid sample fragment is hybridized to the arrayed probes. In this case, sequence information about a fragment may be determined in a simultaneous hybridization reaction with all of the arrayed probes. For sequencing other nucleic acid fragments, the same oligonucleotide array may be reused. The arrays may be produced by spotting or by in situ synthesis of probes.

In Format 3 SBH, two sets of probes are used. In one embodiment, a set may be in the form of arrays of probes with known positions in the array, and another, labelled set may be stored in multiwell plates. In this case, target nucleic acid need not be labelled. Target nucleic acid and one or more labelled probes are added to the arrayed sets of probes. If one attached probe and one labelled probe both hybridize contiguously on the target nucleic acid, they can be covalently ligated, producing a detected sequence equal to the sum of the length of the ligated probes. The process allows for sequencing long nucleic acid fragments, e.g. a complete bacterial genome, without nucleic acid subcloning in smaller pieces.

However, to sequence long nucleic acids unambiguously, SBH involves the use of long probes. As the length of the probes increases, so does the number of probes required to generate sequence information. Each 2-fold increase in length of the target requires a one-nucleotide increase in the length of the probe, resulting in a four-fold increase in the number of probes required (the complete set of probes of length K contains 4^k probes). For example, de novo sequencing without additional mapping information of 100 nucleotides of DNA requires 16,384 7-mers; sequencing 200 nucleotides requires 65,536 8-mers; 400 nucleotides, 262,144 9-mers; 800 nucleotides, 1,048,576 10-mers;

1600 nucleotides, 4,194,304 11-mers; 3200 nucleotides, 16,777,216 12-mers; 6400 nucleotides, 67,108,864 13-mers; and 12,800 nucleotides requires 268,435,456 14-mers.

From any given sequence, however, most of the probes will be negative, and thus much of the information is redundant. For sequencing a 200 bp target nucleic acid with 65,536 8-mers, for example, about 330 measurements (positive and negative) are made for each base pair (65,536 probe measurements/200 bp). For sequencing a 6400 bp sequence with 67,108,864 13-mer probes, the measurement redundancy increases to about 10,500. An improvement in SBH that increases the efficiency and reduces the number of necessary measurements would greatly enhance the practical ability to sequence long pieces of DNA de novo. Such an improvement would, of course, also enhance resequencing and other applications of SBH.

Of interest are disclosures of the use of "binary" pools [see Pevzner and Lipschutz, in Mathematical Foundations of Computer Science 1994, Springer-Verlag, Berlin, pages 143-158 (1995?)], "alternating" probes [Pevzner and Lipschutz, supra], "gapped" probes [Pevzner and Lipschutz, supra; Bains and Smith, J. Theor. Biol., 135:303-307 (1988)], redundant combinations (pools) of probes [Drmanac et al., US Patent No. 5,202,231], probes with degenerate ends in SBH [Bains, Genomics, 11:294-301 (1991)]. See also pools of multiplexed probes [Drmanac and Crkvenjakov, Scientia Yugoslavica, 16(1-2):97-107 (1990)].

Also of interest is the suggestion in WO 95/09248 suggests that extension of the sequence of probe X may be carried out by comparing signals of (a) the four possible overlapping probes generated by a one base extension of the sequence of X and (b) the three single mismatch probes wherein the mismatch position is the first position of X, and adding a base extension only if probe X and the probe created by the base extension have a significantly positive signal compared to the other six probes.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a flow chart for an algorithm to generate pools of probes.

Figure 2 is diagram of a computing device.

Figure 3 is a flow chart describing an algorithm to filter out false positive probes in a set of probes.

Figure 4 shows a flow chart for sequence analysis using Format 3.

SUMMARY OF THE INVENTION

The present invention enhances SBH methods by providing methods and pools of probes that allow greater efficiency in conducting SBH. The use of pools of probes allows a great reduction in the level of redundancy (R), i.e., the number of separate measurements of hybridization signals, required to identify each particular nucleotide in a target nucleic acid sequence.

The present invention also provides pools and sets of pools of probes, as well as methods of synthesizing pools of probes. In such a method of synthesizing an pool of probes, maximal randomization of the probes within a pool is achieved if not more than two different bases are incorporated at one position, and/or if all 6 possible base mixes (A+T, A+C, A+G, T+C, T+G, C+G) are used equally in the synthesis.

The present invention also provides improved SBH sequence assembly methods involving, e.g., the use of an initial filtering algorithm to remove probes that fail to overlap with a prespecified number of other probes, the use of rescoring to better discriminate true positive probes from false positive probes, e.g., by taking into account scores of probes containing a single or double mismatch, the use of continuous value scores for all probes in sequence assembly rather than scoring probes as positive/negative wherein an overlapping sequence comprising 3 or more probes is scored for probability of correctness based on scores of its constituent probes, the use of statistical analysis of scores or probabilities of the probes within an assembled sequence to determine the likelihood that an assembled sequence is the correct target sequence, and the use of likelihoods or other probability scores to determine whether a mutation exists in a reference sequence.

The invention provides methods of identifying one or more sequences of a target nucleic acid comprising: a)contacting a target nucleic acid with a first set of pools of probes, wherein at least one pool in the set comprises a mixture of two or more probes having different sequences in information regions of the probes, under conditions which produce, on average, more probe:target hybridization with probes which are perfectly complementary to the target nucleic acid in the information region of the probes than with probes which are mismatched in the information regions; b) detecting a first subset of pools for which a level of hybridization indicates that there is at least one perfectly complementary probe within each pool; and c) identifying one or more sequences of the target nucleic acid from the first subset of pools detected in step (b) by compiling overlapping sequences of the information regions of the probes in the subset of detected pools, wherein one or more pooling false positive probes are eliminated as a result of compilation of overlapping sequences. In one aspect, the method, further comprises, following step (b) and before step (c), the steps of: a) contacting the target nucleic acid with a second set of pools of probes containing at least one probe having the same information region as a probe in the first set, b) detecting a second subset of pools for which the level of hybridization indicates that there is at least one perfectly complementary probe within each pool; and c) eliminating probes with the same information regions present in both the first set of pools of probes and the second set of pools of probes that are not present in both the first detected subset of pools and the second detected subset of pools. In one aspect, the first and second sets of pools of probes comprise the same information regions. In another aspect, the first and second sets of pools of probes comprise the same probes.

The invention also provides methods of identifying one or more sequences of a target nucleic acid comprising: a) contacting a target nucleic acid with a first set of pools of probes, wherein at least one pool in the set comprises a mixture of two or more probes having different sequences in information regions of the probes, under conditions which produce, on average, more probe:target hybridization with probes which are perfectly complementary to the target nucleic acid in the information region of the probes than with probes which are mismatched in the information regions; b) assigning a hybridization score to each probe wherein each probe within a pool is assigned the same hybridization score, and c) identifying one or more sequences of the target nucleic acid by analysis of hybridization scores of overlapping probes, wherein one or more probes with false high scores arising from pooling of probes are eliminated by analysis of hybridization scores of overlapping. In one aspect a statistical analysis of hybridization scores is performed in step (c). In another aspect of the method, step (c) further comprises calculating a score for the identified one or more sequences of the target nucleic acid.

In another embodiment, the method further comprises, after step (b) and before step (c), the steps of: a) contacting the target nucleic acid with a second set of pools of probes containing at least one probe having the same information region as a probe in the first set, b) assigning a hybridization score to each probe wherein each probe within a pool is assigned the same hybridization score. In addition, the invention provides a method further comprising the step of c) eliminating the higher of two scores for probes present in both the first set and second set of pools of probes. In one aspect of the invention, the first and second sets of pools of probes comprise the same information regions, and in another aspect, the first and second sets of pools of probes comprise the same probes.

Methods of the invention include those wherein the target nucleic acid is labeled and those wherein the probes are labeled. In one aspect, the label is a fluorophore. In another aspect, the label is attached to a terminal nucleotide and or to an internal nucleotide. Methods include those wherein the set of pools of probes is immobilized on one or more solid supports, and those wherein the pools of probes are arranged in a spatially-addressable array in which each pool has a unique address. In other aspects, methods are provided wherein the target nucleic acid is immobilized on one or more solid supports.

The invention further provides methods of identifying one or more sequences of a target nucleic acid comprising: a) contacting a target nucleic acid with a first set of pools of immobilized probes and a first set of pools of labeled probes, wherein at least one pool in either the first set of pools of immobilized probes or the first set of pools of labeled probes, or both, comprises a mixture of two or more probes having different sequences in the information regions of the probes, under conditions which produce, on average, more probe:target hybridization for probes which are perfectly complementary to the target nucleic acid in the information region than with probes which are mismatched in the information region; b) covalently joining adjacently hybridized immobilized probes and labeled probes to provide a first set of covalently joined probes; c) detecting a first subset of pools of covalently joined probes for which a level of hybridization indicates that there is at least one perfectly complementary covalently joined probe within each pool; and d) identifying one or more sequences of the target nucleic acid from the first subset of covalently joined pools or probes detected in step (c) by compiling overlapping sequences of the information regions of covalently joined probes in the subset of detected pools, wherein one or more covalently joined pooling false positive probes are eliminated as a result of compilation of overlapping sequences.

In another embodiment, the method further comprising, following step (c) and before step (d), the steps of: a) contacting the target nucleic acid with a second set of pools of immobilized probes and a second set of pools of labeled probes, wherein at least one probe in the second set of immobilized probes has the same information region as a probe in the first set of pools of immobilized probes, or at least one probe in the second set of labeled probes has the same information region as a probe in the first set of pools of labeled probes, b) covalently joining adjacently hybridized immobilized probes and labeled probes to provide a second set of covalently joined probes; c) detecting a second subset of covalently joined pools or probes for which a level of hybridization indicates that there is at least one perfectly complementary probe within each pool; and d) eliminating covalently joined probes with the same information regions present in both the first subset of covalently joined pools of probes and the second subset of covalently joined pools of probes that are not present in both the first detected subset of covalently joined pools of probes and the second detected subset of covalently joined pools of probes.

The invention also provides methods of identifying one or more sequences of a target nucleic acid comprising: a) contacting a target nucleic acid with a first set of pools of immobilized probes and a first set of pools of labeled probes, wherein at least one pool in the first set of pools of immobilized probes or at least one pool in the first set of pools of labeled probes or both, comprises a mixture of two or more probes having different sequences in the information regions of the probes, under conditions which produce, on average, more probe:target hybridization for probes which are perfectly complementary to the target nucleic acid in the information region than with probes which are mismatched in the information region; b) covalently joining adjacently hybridized immobilized probes and labeled probes to provide a first set of covalently joined probes;

c) assigning a hybridization score to each covalently joined probe in the first set of covalently joined probes wherein each probe within a pool is assigned the same hybridization score, and d) identifying one or more sequences of the target nucleic acid from overlapping covalently joined probes by analysis of hybridization scores of overlapping covalently joined probes wherein one or more covalently joined probes with false high scores arising from pooling of probes are eliminated by analysis of hybridization scores of overlapping probes. In one aspect, the method of the invention further comprises after step (c) and before step (d) the steps of: a) contacting the target nucleic acid with a second set of pools of immobilized probes and a second set of pools of labeled probes, wherein at least one probe in the second set of immobilized probes has the same information region as a probe in the first set of pools of immobilized probes, at least one probe in the second set of labeled probes has the same information region as a probe in the first set of pools of labeled probes, b) covalently joining adjacently hybridized immobilized probes and labeled probes to provide a second set of covalently joined probes; c) assigning a hybridization score to each covalently joined probe of the second set wherein each probe within a pool of covalently joined probes is assigned the same hybridization score. In still another aspect, the invention further comprises the step of d) eliminating the higher of two scores for covalently joined probes present in both the first set and second set of covalently joined pools of probes.

The invention provides methods wherein the first and second sets of pools of immobilized probes or sets of pools of labeled probes or both comprise the same information regions, as well as methods wherein the first and second sets of pools of immobilized probes or sets of pools of labeled probes or both comprise the same probes.

The invention provides methods where probes are labeled with a fluorophore, as well as methods wherein a label of the labeled probe is attached to a terminal nucleotide and/or attached to an internal nucleotide. Methods are provided wherein the set of pools of immobilized probes is immobilized on one or more solid supports, an/or the sets of pools of immobilized probes are arranged in a spatially-addressable array in which each pool has a unique address.

The invention further provides methods wherein a statistical analysis of hybridization scores is performed, as well as methods a step comprising calculating a score for the identified one or more sequences of the target nucleic acid

The invention also provides methods wherein the pools of immobilized probes each consist of one probe, as well as methods wherein the pools of labeled probes each consist of one probe.

The invention also provides methods of sequencing a target nucleic acid, or determining the putative presence of a nucleotide sequence in a target nucleic acid, comprising the steps of: (a) contacting a target nucleic acid with a set of pools of probes wherein each pool comprises a mixed plurality of different probes (preferably of predetermined length and predetermined sequence), wherein the nucleotide sequences of at least two different probes in the pool differ within their information region, under conditions which discriminate in most cases between probe:target hybrids which are perfectly complementary in the information region of the probe and probe:target hybrids which are mismatched in the information region of the probe; (b) detecting those pools of probes of the set of pools of probes which hybridize with the target nucleic acid; and (c) determining the sequence of the target nucleic acid from the subset of pools detected in step (b) by compiling the overlapping sequences of the information regions of the probes in the detected pools, wherein pooling false positives are eliminated as a result of compilation of overlapping sequences.

The pooling methods of the invention may be applied to either Format 1, 2 or 3 SBH. Thus, the probes may be labeled or the target nucleic acid may be labeled. Labels may be fluorophores and may be attached to a terminal nucleotide or an internal nucleotide. Either the probes or the target nucleic acids may be immobilized on a solid support and/or arranged in a spatially-addressable array. In Format 1, the probes are labeled and the target nucleic acid(s) is(are) immobilized on a solid support. In Format 2, the target nucleic acid is labeled and the probes are immobilized on a solid support. In Format 3, some probes are immobilized and some probes are labeled, and either immobilized probes or labeled probes (in solution), or both, may be arranged in subpools. When both immobilized and labeled probes are pooled, the combination of immobilized and labeled probe subpools makes up a pool. For example, a Format 3 pooling method in which both the immobilized and the labeled probes are pooled may comprise the steps of: (a) contacting a target nucleic acid with a set of subpools of probes, wherein each probe is immobilized on a solid support, wherein the subpool comprises a mixed plurality of different probes (preferably of predetermined length and predetermined sequence), and wherein the nucleotide sequences of at least two different probes in the pool differ within their information region, under conditions which discriminate in most cases between probe:target hybrids which are perfectly complementary in the information region and probe:target hybrids which are mismatched in the information region; (b) contacting the array and target nucleic acid with a subpool of labeled probes (preferably of predetermined length and predetermined sequence) under conditions which discriminate between perfectly complementary labeled probe:target complexes and mismatched labeled probe:target complexes; (c) covalently joining adjacently hybridized immobilized probes and labeled probes; (d) identifying which pools of subpools of probes hybridized to the target nucleic acid; and (e) determining the sequence of the target nucleic acid by overlapping the sequences of the region of probes in the detected pools that hybridized, wherein false positives due to pooling are eliminated as a result of compilation of overlapping sequences.

Alternatively, only the labeled probes in solution may be pooled, while each immobilized probe has a unique address in a spatially addressable array. In another embodiment, only the immobilized probes may be pooled, and each pool may be associated with a unique address in a spatially addressable array.

The invention further provides a set of pools of probes wherein each probe comprises an information region, wherein said set of probes is sufficient to determine the sequence of an unknown target nucleic acid by overlapping sequences of the information region of two or more probes, and wherein at least one pool comprises two or more probes having different sequences in the information regions and having the same label or no label, and wherein the set of the pools of probes also satisfies one or more of the following rules describing the information regions of the probes, said rules selected from the group consisting of: (a) a consensus sequence of at least one pool in the set consists only of the letters selected from the group consisting of V, H, D, B, and N as defined in Table A below; (b) a consensus sequence of probes in each pool in the set comprises more than three different letters selected from the group consisting of A, C, G, T, U, M, R, W, S, Y, K, V, H, D, B, and N as defined in Table A below; (c) consensus sequences from each informative position of all pools in the set comprise more than eight letters selected from the group consisting of A, C, G, T, U, M, R, W, S, Y, K, V, H, D, B, and N as defined in Table A below; and (d) consensus sequences from each information region of all pools in the set comprise more than five different letters selected from the group consisting of A, C, G, T, U, M, R, W, S, Y, K, V, H, D, B, and N and at least one of the five letters is selected from the group consisting of M, R, W, S, Y, and K as defined in Table A below. A consensus sequence is determined by alignment of bases in probes within or among pools limiting degeneracy at each aligned position to either one, two, three or four possible bases at that position. Letter coding for all levels of degeneracy are shown in Table A. Alternatively, rules for the set may be selected from the group consisting of (a) no two probes of length K within the pool overlap by K-1 bases; (b) no two probes pools within a pool are reverse complements; (c) less than 50% of the probes in the set are repeated in any two pools within the set; (d) when two or more probes in a pool vary at a nucleotide position, there are no more than three different bases at that varied position; (e)no two probes in a pool overlap by a significant number of bases; and (e) there exists at least one nucleotide position wherein all probes within the pool are identical.

In one aspect, the set of pools of probes of the invention comprises all possible probes of the same length K, where K is greater than 3. In another aspect, each pool comprises more than 16 different probes or at least 32 different probes. In another aspect, the pools are arranged in a spatially-addressable array, and wherein each pool has an address. In still another aspect, at least two pools are mixed, wherein any two pools that are mixed are associated with different labels, and wherein all probes in a single pool are associated with the same label.

TABLE A

Letter	Definition
A	A
C	C
G	G
T	T
U	U
M	A or C
R	A or G
W	A or T/U
S	C or G
Y	C or T/U
K	G or T/U
V	A or C or G
H	A or C or T/U
D	A or G or T/U
B	C or G or T/U
N	A or C or G or T/U

DETAILED DESCRIPTION OF THE INVENTION

The present invention provides an improvement on the basic SBH method. According to the present invention, pools of probes are used to reduce the redundancy normally found in SBH protocols and to reduce the number of hybridization reactions needed to determine a target DNA sequence unambiguously.

For SBH of nucleic acids that are 1000-3200 bases in length, progressively more probes (from about 16,000 7-mers to 16 million 12-mers) must be scored. The present invention provides improved probe pooling and sequence assembly strategies that may significantly reduce the number of independent probe synthesis reactions, hybridization reactions, probe array size and readout time by about 10- to over 1000-fold. The invention is based on the discovery that when pools of unexpectedly large numbers of probes with diverse (dissimilar) sequences are hybridized with the target nucleic acid, and all probes in a pool are scored as positive if any one probe in the pool is positive, assembly of incorrect sequence is unlikely (i.e., the resulting assembled sequence will most likely be the correct target nucleic acid sequence) despite the co-scoring of negative probes in the pool as positive.

False positives due to experimental error (defined as "experimental false positives" herein) are observed with conventional SBH, and may be as high as 2-3% of all probes. Such experimental false positives may be due to errors in probe synthesis (for example, the probe fixed to a spot on an array may have an erroneous nucleotide sequence), errors in hybridization or errors in scoring (for example, probes that are not full match probes may be erroneously scored as positive, due to unusually strong hybridization signals from single mismatch probes, unusually weak hybridization signals from full match probes, problems in setting the threshold of positive/negative scores, or errors in reading data).

In the methods of the present invention, the assignment of the same hybridization score to all probes in a pool (which in positive pools results in co-scoring of negative probes as positive) intentionally introduces additional false positives due to the pooling of probes, called "pooling false positives" herein. When positive/negative scoring is used in de novo sequencing, the total fraction of positive probes (Fp= number of positive probes/number of all probes), which includes true positives, experimental false positives and pooling false positives, can be no more than 25%, and is preferably 20% or less, and more preferably 10% or less.

According to one embodiment, all probes from a positive informational pool are initially scored as positive, and pooling false positives (as well as experimental false positives) are rejected as a result of their failure to overlap with other positive probe sequences in the same or in other positive pools. In comparison, prior strategies that used redundant or binary pools of probes (with different informational content) contemplated an identification of the true positive probes in the pool before overlapping the sequences of the positive probes.

The use of pools reduces dramatically the need for independent probe scoring, especially for longer probes needed for de novo sequencing of kb range targets. Also, a very small number of a few thousand scores may be sufficient to determine the sequence of a target nucleic acid only a few hundred bases in length. The use of pools is applicable to all SBH formats, including a format wherein pools of labeled probes are hybridized to immobilized target samples (Format 1 SBH), a format wherein labeled target is hybridized to immobilized pools of probes (e.g., one pool per spot, Format 2 SBH), and a format that utilizes both arrayed probes and labeled probes in solution (Format 3 SBH). Several options are available in Format 3 SBH; the immobilized probes can be pooled, or the labeled probes can be pooled, or both immobilized and labeled probes can be pooled.

Pools are demonstrated herein to allow sequence assembly with a minimal number of experimental hybridization scores. However, pools provide additional challenges for potential optimization of sensitivity and specificity, such as the ability to detect the positive signal of one out of thousands of labeled probes hybridized together in one pool, and the ability to discriminate negative pools having a cumulative hybridization signal that may be close to the strength of a positive signal, based on hybridization of several single mismatch probes and one end-mismatch probe to the target nucleic acid (while keeping the fraction of pools with a positive signal to an Fp=1/5).

a. Definitions

"Probes" refers to relatively short pieces of nucleic acids, preferably DNA. Probes are preferably shorter than the target DNA by at least one nucleotide, and more preferably they are of a length commonly referred to as oligonucleotides, that is they are

25 nucleotides or fewer in length, still more preferably 20 nucleotides or fewer in length. Of course, the optimal length of a probe will depend on the length of the target nucleic acid being analyzed, and the availability of additional reference sequence or mapping information. For de novo sequencing, without additional mapping information, of a target nucleic acid composed of about 100 or fewer nucleotides, the probes are at least 7-mers; for a target of about 100-200 nucleotides, the probes are at least 8-mers; for a target nucleic acid of about 200-400 nucleotides, the probes are at least 9-mers; for a target nucleic acid of about 400-800 nucleotides, the probes are at least 10-mers; for a target nucleic acid of about 800-1600 nucleotides, the probes are at least 11-mers; for a target of about 1600-3200 nucleotides, the probes are at least 12-mers, for a target of about 3200-6400 nucleotides, the probes are at least 13-mers; and for a target of about 6400-12,800 nucleotides, the probes are at least 14-mers. For every additional two-fold increase in the length of the target nucleic acid, the optimal probe length is one additional nucleotide.

Those of skill in the art will recognize that for Format 3 SBH applications, the above-delineated probe lengths are post-ligation and/or post-extension as described herein. Thus, as used throughout, specific probe lengths refer to the actual length of the probes for format 1 and 2 SBH applications and the lengths of ligated probes in Format 3 SBH. When probes are extended by one base using DNA polymerase to incorporate differentially labeled dideoxynucleotides (thereby allowing identification of the single incorporated base by detecting the label), the probe length would refer to the length post-extension.

Probes are normally single stranded, although double-stranded probes may be used in some applications. While typically the probes will be composed of naturally-occurring bases and native phosphodiester backbones, they need not be. For example, the probes may be composed of one or more modified bases, such as 7-deazaguanosine, or one or more modified backbone interlinkages, such as a phosphorothioate. The only requirement is that the probes be able to hybridize to the target nucleic acid. A wide variety of modified bases and backbone interlinkages that can be used in conjunction with the present invention are known, and will be apparent to those of skill in the art.

The length of the probes described above and throughout the specification refers to the length of the informational content (*i.e.*, the information region or the informative region) of the probes, not necessarily the actual physical length of the probes. The probes used in SBH frequently contain degenerate ends that do not contribute to the information content of the probes. For example, SBH applications frequently use mixtures of probes of the formula $N_xB_yN_z$, wherein N represents any of the four nucleotides and varies for the polynucleotides in a given mixture, B represents any of the four nucleotides but is the same for each of the polynucleotides in a given mixture, and x, y, and z are all integers. Preferably, x is an integer between 0 and 5, y is an integer between 4 and 20, and z is an integer between 0 and 5. Hybridization discrimination of mismatches in these degenerate probe mixtures refers only to the length of the informational content, not the full physical length.

"Target Nucleic Acid" refers to the nucleic acid of interest, typically the nucleic acid that is sequenced in the SBH assay. The nucleic acid can be any number of nucleotides in length, depending on the length of the probes, but is typically on the order of 100, 200, 400, 800, 1600, 3200, 6400, or even more nucleotides in length. The target nucleic acid may be composed of ribonucleotides, deoxyribonucleotides or mixtures thereof. Typically, the target nucleic acid is a DNA. While the target nucleic acid can be double-stranded, it is preferably single stranded so that hybridization to the probe can occur. Moreover, the target nucleic acid can be obtained from virtually any source. Depending on the length of the source nucleic acid, it is preferably fragmented to form smaller targets prior to use in an SBH assay. Like the probes, the target nucleic acid can be composed of one or more modified bases or backbone interlinkages.

Nucleotide bases "match" or are "complementary" if they form a stable duplex by hydrogen bonding under specified conditions. For example. under conditions commonly employed in hybridization assays, adenine ("A") matches thymine ("T"), but not guanine ("G") or cytosine ("C"). Similarly, G matches C, but not A or T. Other bases which will hydrogen bond in less specific fashion, such as inosine or the Universal Base ("M" base, Nichols et al 1994), or other modified bases, such as methylated bases, for example, are complementary to those bases with which they form a stable duplex under specified conditions. A probe is said to be "perfectly complementary" or is said to be a "perfect match" if each base in the probe forms a duplex by hydrogen bonding to a base in the target nucleic acid according to the Watson and Crick base pairing rules (i.e., absent any surrounding sequence effects, the duplex formed has the maximal binding energy for a particular probe). "Perfectly complementary" and "perfect match" are also meant to encompass probes which have analogs or modified nucleotides. A "perfect match" for an analog or modified nucleotide is judged according to a "perfect match rule" selected for that analog or modified nucleotide (e.g., the binding pair that has maximal binding energy for a particular analog or modified nucleotide). Each base in a probe that does not form a binding pair according to the "rules" is said to be a "mismatch" under the specified hybridization conditions.

"Pools of probes" (or informative pools of probes) refers to pools of probes selected for their information content. Preferably, individual pools of probes comprise probes in which the information content (i.e., the sequence of the information region) differs in more than one position. Preferably, individual pools comprise probes that do not overlap over significant portions of their length (for example, an individu pool should not contain the probes AGGATCT and GGATCTG, because the two probes overlap with a one-nucleotide overhang). The probes in an pool need not all be of the same length.

A "set of pools" refers to a set of probes that is sufficient to identify or determine the sequence of a target nucleic acid by SBH, wherein the probes are grouped into pools. The content of the set will vary depending on the length of probes, the length of the target nucleic acid, and the type of sequencing application (*e.g.*, *de novo* sequencing, resequencing, detection of POLYMORPHISMS, diagnostic sequencing, forensic uses, etc.). For *de novo* sequencing, the set may be a set of all possible probes of length K but may alternatively be a subset thereof. For example, a set of all possible probes of length K can be reduced by 50% if reverse complements are eliminated. A universal set of probes includes sufficient probes to analyze a DNA fragment with prespecified precision, e.g. with respect to the redundancy of reading each base pair ("bp"). These sets may include more probes than are necessary for one specific fragment, but may include fewer probes than are necessary for testing thousands of DNA samples of different sequence with sequence-specific probes. In addition, some pools in the set may have only one probe.

b. Preparation of Probes

Probes may be prepared and optionally labeled as described in Int'l Publication No. WO 98/31836 published July 23, 1998 and WO 99/09217 published

February 28, 1999, both of which are incorporated herein by reference. See, e.g., Examples 1 through 4, 15 of WO 98/31836. Oligonucleotide probes may incorporate modified bases and may be labeled with fluorescent dyes, chemiluminescent systems, radioactive labels (e.g., ^{35}S, ^{3}H, ^{32}P or ^{33}P), non-radioactive isotopes, isotopes detectable by mass spectrometry (e.g., electrophore mass labels (EMLs), ligands which can serve as specific binding partners to a labeled antibody, enzymes, antibodies which can serve as a specific binding partner for a labeled ligand, antigens, groups with specific reactivity, and electrochemically detectable moieties.

The probes may optionally be disposed on a solid substrate (e.g., on arrays, particles or other solid supports) as described in Int'l Publication No. WO 98/31836 published July 23 1998 and WO 99/09217 published February 28, 1999, both of which are incorporated herein by reference. See, e.g., Examples 5 through 6, 15, and 32 of WO 98/31836. See also, e.g., Examples 33 through 36 of WO 99/09217. Probes may be fixed to a support (i.e., "fixed probes" or "immobilized probes") by a number of methods known to those skilled in the art, including by passive adsorption, by covalent binding (e.g., by formation of amide groups or phosphodiester linkage between the probe and the support), and by strong binding interactions such as biotin-streptavidin interaction (e.g., through immobilization of biotinylated probes on streptavidin-coated supports). For example, glass, polystyrene, Teflon, nylon, silicon or fluorocarbon supports may be used.

A variety of techniques have been described for synthesizing and/or immobilizing arrays of polynucleotides, including in situ synthesis, where the polynucleotides are synthesized directly on the surface of the substrate (see, e.g., U.S. Patent No. 5,744,305 to Fodor, et al.,) and attachment of pre-synthesized polynucleotides to the surface of a substrate at discrete locations (see, e.g., WO 98/31836, incorporated herein by reference). Additional methods are described in WO 98/31836, incorporated herein by reference, at pages 41-45 and 47-48, among other places, and in the references cited therein. The present invention is suitable for use with any of these currently available, or later developed, techniques. Additionally, methods for normalizing different quantities of compounds immobilized at each spot, such as those described in provisional U.S. Application Serial No. 60/111,961 (Attorney Docket No. 9598-068-888) incorporated herein by reference, may be advantageously used in the context of the present invention.

Oligonucleotides may be organized into arrays, and these arrays may include all or a subset of all probes of a given length, or sets of probes of selected lengths. Hydrophobic partitions may be used to separate probes or subarrays of probes. Arrays may be designed for various applications (e.g. mapping, partial sequencing, sequencing of targeted regions for diagnostic purposes, mRNA sequencing and large scale sequencing). A specific chip may be designed for a particular application by selecting a combination and arrangement of probes on a substrate.

In one embodiment, the substrate which supports the array of probes is partitioned into sections so that each probe in the array is separated from adjacent probes by a physical barrier which may be, for example, a hydrophobic material. In a preferred embodiment, the physical barrier has a width of from 100 μm to 30 μm. In a more preferred embodiment, the distance from the center of each probe to the center of any adjacent probes is 325 μm. These arrays of probes may be "mass-produced" using a nonmoving, fixed substrate or a substrate fixed to a rotating drum or plate with an ink-jet deposition apparatus, for example, a microdrop dosing head; and a suitable robotic system, for example, an anorad gantry. Alternatively, the probes may be fixed to a three-dimensional array (see, e.g., Example 33 of WO 99/09217 published February 28, 1999, incorporated herein by reference).

The probes in these arrays may include spacers that increase the distance between the surface of the substrate and the informational portion of the probes. The spacers may be comprised of atoms capable of forming at least two covalent bonds such as carbon, silicon, oxygen, sulfur, phosphorous, and the like, or may be comprised of molecules capable of forming at least two covalent bonds such as sugar-phosphate groups, amino acids, peptides, nucleosides, nucleotides, sugars, carbohydrates, aromatic rings, hydrocarbon rings, linear and branched hydrocarbons, and the like.

Reusable arrays may be produced as described in Int'l Publication No. WO 98/31836 published July 23, 1998 and WO 99/09217 published February 28, 1999, both of which are incorporated herein by reference. See, e.g., Example 18 of WO 98/31836. A reusable Format 3 SBH array may be produced by introducing a cleavable bond between the fixed and labeled probes and then cleaving this bond after a round of Format 3 analyzes is finished. If the labeled probes contain ribonucleotides or if a ribonucleotide is used as the joining base in the labeled probe, this probe may subsequently be removed, e.g., by RNAse or uracil-DNA glycosylate treatment, or

NaOH treatment. In addition, bonds produced by chemical ligation may be selectively cleaved.

Other variations include the use of modified oligonucleotides to increase specificity or efficiency, cycling hybridizations to increase the hybridization signal, for example by performing a hybridization cycle under conditions (e.g. temperature) optimally selected for a first set of labeled probes followed by hybridization under conditions optimally selected for a second set of labeled probes. Shifts in reading frame may be determined by using mixtures (preferably mixtures of equimolar amounts) of probes ending in each of the four nucleotide bases A, T, C and G.

Rather than being ordered on an array, the probes may alternatively be complexed (covalent or noncovalent) to discrete particles wherein the particles can be grouped into a plurality of sets based on a physical property. In a preferred embodiment, a different probe is attached to the discrete particles of each set, and the identity of the probe is determined by identifying the physical property of the discrete particles. In an alternative embodiment, the probe is identified on the basis of a physical property of the probe. The physical property includes any that can be used to differentiate the discrete particles, and includes, for example, size, fluorescence, radioactivity, electromagnetic charge, or absorbance, or label(s) may be attached to the particle such as a dye, a radionuclide, or an EML. In a preferred embodiment, discrete particles are separated by a flow cytometer which detects the size, charge, flourescence, or absorbance of the particle. See, e.g., Example 36 of WO 99/09217 published February 28, 1999, incorporated herein by reference.

The probes complexed with the discrete particles can be used to analyze target nucleic acids. These probes may be used in any of the methods described herein, with the modification of identifying the probe by the physical property of the discrete particle. These probes may also be used in a Format 3 approach where the "free" probe is identified by a label, and the probe complexed to the discrete particle is identified by the physical property. In a preferred embodiment, the probes are used to sequence a target nucleic acid using SBH.

Probes may be labeled with different labels and multiplexed in a set so that each probe of a set can be differentiated from the other probes in the same set by its label. See, e.g., Example 30 of WO 98/31836, incorporated herein by reference.

For example, different radioisotopes, fluorescent labels, chromophores, or EMLs, or mixtures thereof may be used for multiplexing.

c. Selection of Sets of Probes To Be Hybridized to Target Nucleic Acid

Sets of probes to be hybridized to target nucleic acid may be selected as described in Int'l Publication No. WO 98/31836 published July 23, 1998 and WO 99/09217 published February 28, 1999, both of which are incorporated herein by reference. See, e.g., Examples 1 through 4, 11 through 17, and 19 through 29 of WO 98/31836.

A universal set of probes includes sufficient probes to analyze a DNA fragment with prespecified precision, e.g. with respect to the redundancy of reading each base pair ("bp"). A small subset of probes may be selected that is still sufficient for reading every bp in any sequence with at least one probe. For example, 12 of the 16 possible 2-mers are sufficient to read 2 consecutive bases. A small subset for 7-mers, 8-mer and 9-mers for sequencing double stranded DNA may be about 3000, 10,000 and 30,000 probes, respectively.

A less than universal set of probes may also be selected to identify a target nucleic acid of known sequence and/or to identify alleles or mutants of a target nucleic acid with a known sequence. Such a set of probes contains sufficient probes so that every nucleotide position of the target nucleic acid is read at least once. Alleles or mutants are identified by the loss of binding of one of the "positive" probes. The specific sequence of these alleles or mutants may then be determined by interrogating the target nucleic acid with sets of probes that contain every possible nucleotide change and combination of changes at these probe positions.

Sets of probes may comprise 2 probes or more, 50 probes or more, preferably 100 probes or more, and more preferably 256 probes or more.

DNA or allele identification and a diagnostic sequencing process may include the steps of: selecting a subset of probes from a dedicated, representative or universal set to be hybridized with target nucleic acid(s) optionally disposed in an array; performing hybridization and scoring of the hybridization results, which can be carried out in parallel with multiple subsets of probes selected in step 1; optionally processing the hybridization results to obtain a final sequence analysis or to determine whether additional probes should be hybridized; and repeating the hybridization, scoring and optionally the processing steps for the remaining probes in the set until a final sequence analysis is obtained.

A known target nucleic acid may be sequenced as follows. One embodiment involves hybridization to the target of a sufficient set of probes that covers every base in the known reference sequence at least once. For this purpose, a specific set of probes may be synthesized for a standard sample. The results of hybridization with such a set of probes reveal whether and where mutations (differences) occur in non-standard samples. Further, this set of probes may include "negative" probes to confirm the hybridization results of the "positive" probes. To determine the identity of the changes, additional specific probes may be hybridized to the sample. This additional set of probes will have both "positive" (the mutant sequence) and "negative" probes, and the sequence changes will be identified by the positive probes and confirmed by the negative probes.

In another embodiment, all probes from a universal set may be hybridized to the target. Use of a universal set of probes in a multistep process allows scoring of a relatively small number of probes per sample; as noted above, successive hybridizations involving a first step of computing an optimal subset of probes to be hybridized first and, then, on the basis of the obtained results, a second step of determining which of the remaining probes in the set should be hybridized next. Both sets of probes may have negative probes that confirm the positive probes in the set. Further, the sequence that is determined may then be confirmed in a separate step by hybridizing the sample with a set of negative probes identified from the SBH results.

For SBH of a random nucleic acid sequence using errorless data (i.e., no experimental false negatives and no experimental false positives), the read length of the target nucleic acid is defined by the probe length if no additional information to the list of positive probes, K bases in length, is used (see Table 1). The limits are defined by the probability of repeating in a target nucleic acid a K-1 oligonucleotide sequence used for sequence assembly in a target sequence. Sequences with a biased content of nucleotides (e.g. AT or GC rich sequences) need even longer probes. The number of probes per base pair for optimal read length exponentially increases as longer probes are used, and an extremely small percentage of probes is positive. The explanation for this almost paradoxical inefficiency of long probes is in the completeness criteria. The incomplete assembly of sequences longer than 100 bases with 7-mers requires combining accurately determined sequences of about 20 bases in length in a different order. The incomplete assembly of sequences longer than 25kb with 15-mers means that only the order of accurately determined sequences of 2-5 kb (3-6x gel read length) may be incorrect. The missing information to map sufficiently long sequence segments may be easily provided (for example by restriction analysis) extending the potential target read length of 10-mers from 800 bases to over 2kb, and 15-mers from 25kb to over 200kb.

Table 1. Relationship of probe length, target sequence read length and %positive probes

No. bases in probe	No. possible probes	No. bases in target that can be read (for >90% assembly rate)	No. probes/ base	% probes that are positive
7	16,384	100	160	0.600
8	65,536	200	320	0.300
9	262,144	400	640	0.150
10	1,048,576	800	1,280	0.075
11	4,194,304	1,600	2,560	0.037
12	16,777,216	3,200	6,120	0.019
13	67,108,864	6,400	12,240	0.009
14	268,435,456	12,800	24,480	0.005
15	1,073,741,824	25,600	48,960	0.002

The use of an array of samples avoids consecutive scoring of many oligonucleotides on a single sample or on a small set of samples. This approach allows the scoring of more probes in parallel by manipulation of only one physical object. Subarrays of DNA samples 1000 bp in length may be sequenced in a relatively short period of time. If the samples are spotted at 50 subarrays in an array and the array is reprobed 10 times, 500 probes may be scored. In screening for the occurrence of a mutation, enough probes may be used to cover each base three times. If a mutation is present, several covering probes will be affected. The use of information about the identity of negative probes may map the mutation with a two base precision. To solve a single base mutation mapped in this way, an additional 15 probes may be employed.

These probes cover any base combination for two questionable positions (assuming that deletions and insertions are not involved). These probes may be scored in one cycle on 50 subarrays which contain a given sample. In the implementation of a multiple label scheme (i.e., multiplexing), two to six probes, each having a different label such as a different fluorescent dye, may be used as a pool, thereby reducing the number of hybridization cycles and shortening the sequencing process.

In more complicated cases, there may be two close mutations or insertions. They may be handled with more probes. For example, a three base insertion may be solved with 64 probes. The most complicated cases may be approached by several steps of hybridization, and the selecting of a new set of probes on the basis of results of previous hybridizations.

If subarrays to be analyzed include tens or hundreds of samples of one type, then several of them may be found to contain one or more changes (mutations, insertions, or deletions). For each segment where mutation occurs, a specific set of probes may be scored. The total number of probes to be scored for a type of sample may be several hundreds. The scoring of replica arrays in parallel facilitates scoring of hundreds of probes in a relatively small number of cycles. In addition, compatible probes may be pooled. Positive hybridizations may be assigned to the probes selected to check particular DNA segments because these segments usually differ in 75% of their constituent bases.

By using a larger set of longer probes, longer targets may be analyzed. These targets may represent pools of fragments such as pools of exon clones.

d. Designing and Optimizing Pools of Probes

Several considerations are involved in generating the pools of probes. First, the basic logic of pooling is to avoid putting together related pairs or sets of probes; that is, the pools should be designed to minimize offset overlaps within the sets. In particular, for probes of length K, the probes in a pool should preferably not contain overlaps of K - 1 nucleotides. For example if the AAAA(C,T)AAA pair of probes is present in one pool, the overlapping AAA(C,T)AAAC pair of probes should not be in the same pool. Additionally, probes in a pool are preferably not reverse complements of any other probes in the same pool. However, probes that are degenerate at specific internal positions may be placed in the same set. This property provides for particularly efficient methods of producing the probes, as many probes may be produced at once as degenerate pools of probes. For example, a pool of eight 7-mer probes might comprise probes of the sequence BYBRBSB, where B is a nucleotide that is the same for all probes in the pool, Y is C or T, R is A or G, and S is C or G. Similarly, a pool of sixteen 10-mers might comprise four degenerate positions with two mixed nucleotides at each position. Changing the combinations of the two mixed nucleotides, and the locations of the degenerate positions, is used to generate all pools. The preferred strategy is to have not more than two mixed nucleotides per degenerate position, although in some cases a degenerate position may contain three or four mixed nucleotides. Additionally, the probes in a pool should preferably vary at more than one position, and more preferably at more than two positions.

In an alternative embodiment, the probes may be synthesized individually and then mixed into appropriate pools; this method is particularly suited for small sets of shorter probes. In many cases, it will also be possible to construct the pool using simple random assignment of the probes to pools; if the number of pools is sufficiently large, then random assignment should generally yield pools that meet the above criteria.

Pools may also be efficiently prepared by mixed base synthesis. Maximal randomization is achieved if not more two different bases are incorporated at one position, and all 6 possible base mixes (A+T, A+C, A+G, T+C, T+G, C+G) are used equally. In this synthesis of pools, a small number of probes per pool will differ by one base only (one per each degenerated base position). The preparation of pools in mixed synthesis is applicable both for arrays of attached probes and for pools of labeled probes. The number of independent probe synthesis may be reduced from millions (see Table 3) to thousands if each pool is prepared in one reaction.

An example of mixed synthesis of 3-mer probes grouped in 8 pools, each containing eight 3-mers, is illustrated below. These pools should represent all 64 3-mer probes without repeating any probe in two pools. The pools may be prepared in 8 synthesis reactions by incorporating specified mixes of two nucleotides. There are 6 possible two-base mixtures (a+t, a+c, a+g, c+g, c+t, and g+t). Pool designing starts with the first base position. All four bases at the first position may be represented by synthesizing two pools of two bases, e.g. (a+t) and (c+g). Each of these two bases in two pools may be extended to form four pools of 2-mers; in the next round of extension four pools of 2-mers may be extended to form eight 3-mer pools.

Pool set 1	3-mer probes in each pool	overlapping probes
1. (a+t) (a+g) (a+c)	(aaa, aac, aga, agc, taa, tac, tga, tgc)	aaa/aac, taa/aaa,
2. (a+t) (a+g) (g+t)	(aag, aat, agg, agt, tag, tat, tgg, tgt)	aag/agg, aag/agt
3. (a+t) (c+t) (a+g)		
4. (a+t) (c+t) (c+t)		
5. (c+g) (a+c) (a+g)		
6. (c+g) (a+c) (c+t)		
7. (c+g) (g+t) (a+c)		
8. (c+g) (g+t) (g+t)		
Pool set 2		
1. (a+t) (a+g) (a+t)	(aaa, aat, aga, agt, taa, tat, tgs, tgt)	(aaa/aat, taa/aaa
2. (a+t) (a+g) (c+g)	(aac, aag, agc, agg, tac, tag, tgc, tgg)	(aag/agc, aag/agg
3. (a+t) (c+t) (a+c)		
4. (a+t) (c+t) (g+t)		
5. (c+g) (a+c) (a+g)		
6. (c+g) (a+c) (c+t)		
7. (c+g) (g+t) (a+t)		
8. (c+g) (g+t) (c+g)		

As illustrated above, none of pools in pool set 1 have two positions with the same mix of two bases, In the case of pool set 2, the first and last pool have the same mix at the first and the last position. Pools like (a+t) (a+t) (a+t) (aaa, aat, ata, att, taa, tat, tta, ttt) should be avoided because they will contain too many mutual overlaps (aaa/aat, aat/ata, ata/taa, taa/aaa, tat/ata, etc.)

Another example of mixed synthesis of 10-mer probes grouped in pools of 64 10-mers comprising all individual bases and all combinations of two bases is as follows:

(a,c)t(c,g)(g,t)a(a,g)cg(a,t)(c,t)

A combination of low complexity pools can also be prepared by mixed synthesis and pooling of pools.

The main possible negative result from using a set of pools of probes, when compared to the use of a set of individual probes, is that a particular set of positive pools can define not only the real sequence (the reason that those pools are positive) but also a different, false sequence that happened to share the same set of positive pools. Thus, the pooling would be non-informative if it contained sets of overlapping probes representing two different sequences in the same pools. The random pooling of probes minimizes that happening. Of course, in any set of randomly generated pools, many similar pools may be generated, and exceptional cases may arise. Thus, randomly generated pools may preferably be further optimized by extensive testing of sequences of certain length (for example, 10-30 nucleotides). The optimal results would be achieved if all possible sequences in such a size range are tested. For each of these sequences, positive pools are defined and then all possible sequences that can be generated from the same positive set of pools are assembled. If sequences of that length other than the starting sequence are formed, then some of the probes defining these sequences are moved to other pools. The process may be run in large number of optimization cycles to create optimum sets of pools of probes.

A flowchart of a pool selection process 100 employing the above criteria is illustrated in FIG. 1. The process 100 can be implemented by a human operator and/or a computing device (described below). The process 100 begins by generating a set of pools (step 102). In this example, sixteen pools, each containing sixty-four probes of length five, are generated. Of course, a person of ordinary skill in the art will readily appreciate that any number of pools containing any number of probes of any length may be similarly generated.

Next, at step 104, the process 100 retrieves the first pool in the set of generated pools. The retrieved pool is then examined in three different ways. First, the pool is examined to determine if any four positions of one probe contain the same bases as any other probe in the same pool (step 106). Second, the pool is examined to determine if any four consecutive positions of one probe match any four consecutive positions of another probe in the same pool (step 108). Third, the pool is examined to determine if there are any reverse complements (step 110).

If the first pool passes all of these tests (i.e., the NO path on steps 106, 108, and 110), and there are more pools in this set to be tested (step 112), then the process 100 moves on to the next pool in this set at step 104. If all of the pools in a particular set pass the above three tests, the set is recorded as a preferred set of pools at step 114. If any of the tests fail (i.e., a YES path on step 106, 108, or 110), and there are more sets of pools to be generated (step 116), then the process generates the next set of pools at step 102 and repeats the process. Step 116 may force the process to be exhaustive, it may try only certain predetermined sets, or it may try a predetermined number of sets.

If no valid set of pools is determined after all sets have been tested, the process indicates this result at step 118. In the event that no valid set of pools is determined, the process may modify the criteria used by steps 106, 108, and/or 110 to increase the likelihood that a valid set of pools will be found. For example, step 106 may be modified to determine if any three positions of one probe contain the same bases as another probe. Step 106 could also be modified to take the YES path only if three or more probes contain the same bases in a predetermined number of positions. Similarly, the requirement of steps 108 and/or 110 may be loosened by reducing the number of positions and/or increasing the number of matches needed to produce a YES result.

The desired number of pools and number of probes in each pool may be determined as follows. The pools are designed with a specified level of redundancy R, representing the number of score measurements (hybridizations) required to identify each nucleotide of a sequence. Positive probes are very rare (1 in 100 to 1 in 50,000, see Table 1). Thus, even after substantial probe pooling only a fraction of pools will be positi If random pools of probes are used in SBH for de novo sequencing, and complete errorless data is assumed, the optimal pool number and pool size can be determined by the probability that several consecutive false positive overlapping probes will occur. The fraction of pools expected to give a positive result (Fp) is P/T, where P is the total number of probes initially scored as positive and T is the total number of probes in the set. The Fp is inversely related to the level of redundancy R (Fp=1/R). The probability that a probe having a length of K bases (which is also referred to as a K-mer or a K-tuple) was falsely scored is thus 1/R, and therefore the probability that two falsely scored probes will consecutively overlap by chance (the "POC") is 4/R (or 4 x Fp). R must thus be greater than 4; otherwise, the probability of false overlaps will be equal to 1 (4 x 1/4). Preferably, R is less than about 100, or less than about 50, or less than about 20, or less than about 10, or about 5 or less, or between 4 and about 5. When the R value is about 10, then 1/R = 0.1, and thus about 10% of all pools are scored as positive when hybridized with a target sequence. For other sequencing applications, e.g., when the reference sequence is known, R may be very low (e.g., close to 4).

An R value of 5 still provides acceptable sequencing results, even for de novo sequencing. Because many consecutive false overlaps are required in order to assemble an incorrect sequence, the tolerable probability of false overlaps may be very close to 1 (i.e., Fp may be close to 1/4). For example, if Fp=1/5 then the POC, i.e. the probability of a false overlap occurring by chance, is 4/5 (4 x Fp) and assembly of an incorrect target sequence of length L is POC^L. For a 100 base target sequence, probability of assembling an incorrect sequence is about 2×10^{-10} for each attempt to assemble sequence (where for 10-mers, about 2×10^5 attempts are made to assemble sequence, or for 15-mers, about 2×10^8 attempts).

Once the value of R is chosen, it can be used to determine the appropriate number of pools and number of probes per pool. For a target sequence of a given length L, the total number of pools required is R x L. One of skill in the art would then determine the appropriate probe length K required to sequence a target of length L. Since the total number of probes in the complete set of K-mers is 4^k, the number of probes in each pool is then given by $4^k/(R \times L)$. The number of probes in all positive pools combined is $4^k/R$, since only one in R pools is positive. For example, if the desired redundancy is R = 10 and the target is of length L = 3000 nucleotides, then K = 12 and the total number of pools desired is approximately 10 x 3000 = 30,000, and the number of probes in each pool is $4^{12}/30{,}000 = 560$ probes.

The pools of probes may be generated each time sequencing of a target is desired. In practice, however, the present invention will probably be most useful if "standard" sets of pools are used. For a given sequence of length L, one of skill in the art will know what probe length K is required. All sequences that can be sequenced by probes of length K may then use the same set of pools of probes of length K. The number of pools and number of probes in each pool of the standard set will be determined by the maximal sequence length that may be determined by probes of length K. For example, sequencing DNA targets having lengths between 1601 and 3200 nucleotides requires probes of length K = 12, and the total set of probes of length 12 is $4^{12} = 16{,}777{,}216$ probes. Since the maximal sequence length L that may be sequenced by 12-mers is 3200 nucleotides, the maximal total number of pools required is 10 x 3200 = 32,000. The number of probes in each pool is then 16,777,216/32,000 = 525. Thus, the "standard" set of pools for sequencing DNA targets having lengths between 1601 and 3200 nucleotides will comprise 32,000 pools of probes, each containing 525 probes. The specific probes that comprise each pool are then selected according to the rules set out above. Preferably, the pools are also optimized according to the method described above. Once the pools are so defined and constituted, they can then be used for any sequence in the particular length range. Similar "standard" sets can be defined and optimized for any desired sequence length.

An incorrect sequence may be assembled from a small number of false positive probes if these false positive probes overlap with some true positive probes. Two classes of these false positive probes are illustrated in Table 2 below

1) Assembly of an incorrect sequence containing a single base substitution or insertion compared to the correct sequence requires a specific set of K false positive overlapped probes. Assembly of an incorrect sequence containing multiple base substitutions or insertions requires more than K probes. Assembly of an incorrect sequence containing a deletion of any size requires K-1 probes. Fewer than K false probes may be sufficient if the mutation is in a tandem repeat. There are about 7L possible different one-base changes.

2) Branching points may be created at K-2 or shorter repeats; for the K-2 case, a minimum of 4 specific false positive probes is required, but it can occur only in a few places in the target sequence; for the K-3 case, 8 probes are required and only dozen cases can exist in a target sequence of the appropriate length for K-mer probes;

```
        1)  atctgtgtctgaagtagtcc
               tgtgg
                gtggc
                 tggct
                  ggctg
                   gctga
            atctgtggctgaagtagtcc


2) a) bbbbbbatttcbbbbbgcactbbbbgtttgbbbacacgbbbbb
            atttg     gcacg    gtttc   acact

   b) bbbbbbatttgbbbacactbbbbgtttcbbbbbgcacgbbbbb
            atttc   acacg    gtttg     gcact
```

Table 2. Specific false positive 5-mer probes causing branching. 1) Only five (K=5) false positive probes, listed above, are sufficient to make a single base TtoG substitution. 2) Only four specific false positive probes can create a branching point for two pairs of K-2 repeats (a ttt pair and a cac pair). If these pairs of repeats are four bases long (equal to K-1 for 5-mers), they will represent regular branch points. In this case (K-2 repeats), without these four false positive probes a unique sequence would be assembled. If the listed single base mismatch probes are scored, a second sequence is also assembled where segments tttcbbbbbgccc and tttgbbbaccc from the real sequence are switched. To assemble this incorrect sequence false probes are incorporated and four correct probes are seen as false positive. The same conditions apply for all sequences having any base at positions denoted by b.

In these specific illustrated cases, the probability of one base extension is equal to Fp, not to 4Fp, because a specific probe that overlaps by K-1 is required. Thus, for Fp=1/5, the probability of assembling an incorrect sequence containing a deletion (the most prevalent error) is $(1/5)^{K-1}$. For sequencing a target of 800 bases with 10-mers, there are about 800x800/10 (6.4×10^4) possible deletions shorter than 80 bases, and the probability of each is $(1/5)^9$ or 5×10^{-7}. Thus, a deletion will assemble only once in more than 30 experiments; for sequencing a target of 3200 bases with 12-mers, one in 50 experiments will have a sequence containing a deletion.

It appears an Fp of 1/5 (i.e., a redundancy of about 5 or less) or more provides satisfactory results. Table 3 shows the number and the size of pools for probes ranging from 10 to 17 bases in length assuming a minimal redundancy of 5 scores per base. Obviously, for the longer probes, the large pools require readers with high sensitivity, very specific full match hybridization, and proper computation software and hardware for fast sequence assembly. Statistical analysis shows that over 100 kb may be sequenced in one reaction with 500,000 pools of 32,000 17-mers each, providing that the necessary sensitivity and specificity in scoring positive pools are achieved.

Table 3. Number and size of pools for Fp=1/5

Probe length	Number of probes	Read length	Number of pools	Probe/pool
10	1M	800	4,000	250
11	4M	1,600	8,000	500
12	16M	3,200	16,000	1,000
13	64M	6,400	32,000	2,000
14	256M	12,800	64,000	4,000
15	1B	25,600	128,000	8,000
16	4B	51,200	256,000	16,000
17	16B	102,400	500,000	32,000

The redundancy of 5 scores per target nucleic acid base is very close to 4 measurements per base required for gel and other methods that experimentally provide one measurement for each of four nucleotides at each position in the target nucleic acid. The redundancy of measurements should not be mistaken for the number of positive probes per target base pair. In SBH experiment, the number of positive probes (i.e., the number of reads) per target base is equal to the number of bases in the probes used (K). In contrast, gel sequencing reads each base only once. Thus for probes 10-20 bases in length and Fp=1/5, SBH provides 8-16 fold more passes than gel sequencing, which should increase the accuracy of identifying (or "calling") bases.

e. Preparation of Target Nucleic Acid

Target nucleic acid may be prepared, optionally labeled, and optionally disposed on a solid substrate as described in Int'l Publication No. WO 98/31836 published July 23, 1998 and WO 99/09217 published February 28, 1999, both of which are incorporated herein by reference. See, e.g., Examples 7 through 8 and 15 of WO 98/31836. Nucleic acids and methods for isolating and cloning nucleic acids are well known to those of skill in the art. See e.g., Ausubel et al., *Current Protocols in Molecular Biology, Vol. 1-2*, John Wiley & Sons (1989); and Sambrook et al., Molecular Cloning A Laboratory Manual, 2nd Ed., Vols. 1-3, Cold Springs Harbor Press (1989), both of which are incorporated by reference herein.

A nucleic acid sample to be sequenced may be fragmented or otherwise treated (for example, by the use of recA) to avoid hindrance to hybridization from secondary structure in the sample. The sample may be fragmented by, for example, digestion with a restriction enzyme such as Cvi JI, physical shearing (e.g. by ultrasound or low pressure), treatment with uracil DNA glycosylase, or by NaOH treatment. The resulting fragments may be separated by gel electrophoresis and fragments of an appropriate length, such as between about 10 bp and about 40 bp, may be extracted from the gel. The minimal length of a nucleic acid fragment suitable for SBH analysis is about 2 x K, where K is the probe length. Nucleic acid may also be obtained by a process that yields a single stranded product, e.g., asymmetric PCR (as described in co-owned U.S. Application Serial No. 60/148,942 filed August 13, 1999 entitled "The Use of Asymmetric PCR for SBH."

Even short nucleic acid fragments can provide useful information. For example, mutations in a gene such as single nucleotide POLYMORPHISMS or other allelic variants can be identified from a small fragment of the gene. For example, if the location of the mutation is known, PCR can be used to amplify a suitable fragment that includes the location of interest. Even if large amounts of DNA are to be analyzed (e.g., for genotyping of a genome or portion thereof), the preparation of low complexity targets is possible by, e.g., cleaving the genomic DNA with one or two restriction enzymes to generate 3 million fragments of about 1000 bp, ligating these fragments to adaptors bound to a solid support, then cleaving the 1000 bp fragments again down to about 30 bp fragments. This results in greatly reduced target complexity and provides short fragments that potentially encode 1 million polymorphic sites.

As another example, SBH analysis can be used to detect the presence or expression of a gene or mRNA in a sample even if only the 3' or 5' end of the gene or mRNA is analyzed. Particularly for mRNA, oligo-dT priming can be used to generate small fragments of the 3' end of mRNAs which can then be analyzed by SBH.

In a preferred embodiment, the "fragments" of the nucleic acid sample cannot be ligated to other fragments in the pool. Such a pool of fragments may be obtained by treating the fragmented nucleic acids with a phosphatase (e.g., calf intestinal phosphatase). Alternatively, nonligatable fragments of the sample nucleic acid may be obtained by using random primers (e.g., N_5-N_9, where N = A, G, T, or C) in a Sanger-dideoxy sequencing reaction with the sample nucleic acid. This will produce fragments of DNA that have a complementary sequence to the target nucleic acid and that are terminated in a dideoxy residue that cannot be ligated to other fragments.

Partitioned membranes allow a very flexible organization of experiments to accommodate relatively larger numbers of samples representing a given sequence type, or many different types of samples represented with relatively small numbers of samples. A range of 4-256 samples can be handled with particular efficiency. Subarrays within this range of numbers of dots may be designed to match the configuration and size of standard multiwell plates used for storing and labeling oligonucleotides. The size of the subarrays may be adjusted for different number of samples, or a few standard subarray sizes may be used. If all samples of a type do not fit in one subarray, additional subarrays or membranes may be used and processed with the same probes. In addition, by adjusting the number of replicas for each subarray, the time for completion of identification or sequencing process may be varied.

f. Hybridization of Pools of Probes

In use, the defined pools of probes are then hybridized with the target nucleic acid. The hybridization conditions used will depend upon, among other factors, the G+C content of the sequence of interest and the lengths of the probes in the pools. Hybridization and washing conditions may be selected to allow detection of substantially perfect match hybrids (such as those wherein the fragment and probe hybridize at six out of seven positions), may be selected to allow differentiation of hybridization signals from perfectly complementary (full match) probes and single base pair mismatch probes, or may be selected to permit detection only of perfectly matched hybrids. Hybridization and washing conditions useful for discriminating between perfect complements and mismatches in the informational content of the probes for a variety of hybridization arrays have been described in the art. For example, hybridization and washing conditions useful for discriminating complementary and mismatched hybrids in a variety of SBH and other applications are described in U.S. Patent No. 5,525,464 to Drmanac et al., and Int'l Publication Nos. WO 95/09248, WO 98/31836 published July

23, 1998, and WO 99/09217 published February 28, 1999, all of which are incorporated herein by reference. A particularly detailed discussion of the theoretical and practical considerations involved in determining hybridization conditions, and including a discussion of the advantages of low-temperature washing steps, may be found in WO 98/31836, incorporated herein by reference, particularly pages 50-62 and Examples 9 through 10, 12, 13, 15 and 26. See also, e.g., Example 37 of WO 99/09217. Additional guidance may be found in Harmes and Higgins, Nucleic Acid Hybridization: A Practical Approach, 1985, IRL Press, Oxford, England.

Suitable hybridization conditions may be routinely determined by optimization procedures or pilot studies. Such procedures and studies are routinely conducted by those skilled in the art to establish protocols for use in a laboratory. See e.g., Ausubel et al., *Current Protocols in Molecular Biology, Vol. 1-2*, John Wiley & Sons (1989); Sambrook et al., Molecular Cloning A Laboratory Manual, 2nd Ed., Vols. 1-3, Cold Springs Harbor Press (1989); and Maniatis et al., *Molecular Cloning: A Laboratory Manual*, Cold Spring Harbor Laboratory Cold Spring Harbor, New York (1982), all of which are incorporated by reference herein. For example, conditions such as temperature, concentration of components, hybridization and washing times, buffer components, and their pH and ionic strength may be varied.

In embodiments wherein the labeled and immobilized probes are not physically or chemically linked, detection may rely solely on washing steps of controlled stringency. Under such conditions, adjacent probes have increased binding affinity because of stacking interactions between the adjacent probes. Conditions may be varied to optimize the process as described above.

In embodiments wherein the immobilized and labeled probes are ligated, ligation may be implemented by a chemical ligating agent (e.g. water-soluble carbodiimide or cyanogen bromide), or a ligase enzyme, such as the commercially available T_4 DNA ligase may be employed. The washing conditions may be selected to distinguish between adjacent versus nonadjacent labeled and immobilized probes exploiting the difference in stability for adjacent probes versus nonadjacent probes.

Agents which destabilize the binding of complementary polynucleotide strands (decrease the binding energy), or increase stability of binding between complementary polynucleotide strands (increase the binding energy) may also be used. In preferred embodiments, the agent is a trialkyl ammonium salt, sodium chloride, phosphate salts, borate salts, organic solvents such as formamide, glycol, dimethylsulfoxide, and dimethylformamide, urea, guanidinium, amino acid analogs such as betaine, polyamines such as spermidine and spermine, or other positively charged molecules which neutralize the negative charge of the phosphate backbone, detergents such as sodium dodecyl sulfate, and sodium lauryl sarcosinate, minor/major groove binding agents, positively charged polypeptides, and intercalating agents such as acridine, ethidium bromide, and anthracine. In a preferred embodiment, an agent is used to reduce or increase the T_m of a pair of complementary polynucleotides. In a more preferred embodiment, a mixture of the agents is used to reduce or increase the T_m of a pair of complementary polynucleotides. In a most preferred embodiment, an agent or a mixture of agents is used to increase the discrimination of perfect matches from mismatches for complementary polynucleotides. In a preferred embodiment, the agent or agents are added so that the binding energy from an AT base pair is approximately equivalent to the binding energy of a GC base pair. The energy of binding of these complementary polynucleotides may be increased by adding an agent that neutralizes or shields the negative charges of the phosphate groups in the polynucleotide backbone. See, e.g., Example 37 of WO 99/09217 published February 28, 1999, incorporated herein by reference.

g. Sequence Assembly

Data may be scored and analyzed and sequence assembled generally as described in Int'l Publication No. WO 98/31836 published July 23, 1998 and WO 99/09217 published February 28, 1999, both of which are incorporated herein by reference. See, e.g., Examples 11 through 17 and 28 through 29 of WO 98/31836. For example, subfragments may be generated by overlapping positive probe sequences until an ambiguity arises because of a branch point (i.e., a probe sequence is repeated in the target nucleic acid), or because of a repetitive sequence longer than the probe, or because of an uncloned segment. Subfragments may be linearly ordered to regenerate the complete sequence of the target nucleic acid fragment by a variety of techniques known in the art, e.g., hybridization with longer probes spanning the site of overlap alternatives, competitive hybridization, ligation of alternative end to end pairs of probes spanning the site of ambiguity or single pass gel analysis.

The assembly process using information from pools of probes is similar to that of conventional SBH in the presence of many false positive probes. Ideally (where there are no false negatives and no branch points), the result of this sequence assembly would be one long correct sequence and shorter incorrect sequences or individual probes that cannot be extended by overlapping with other probes. The creation of many more false positive probes than true positive probes by pooling of probes causes many more shorter sequences to be assembled. Sequences are particularly prone to error at their 5' and 3' ends, so that knowledge of correct 5' or 3' end sequences (for example primer sequences) allows better selection of the correct sequence. In addition, knowledge of the real length of the target sequence allows better selection of a correct sequence.

Data from the positive pools of probes may be analyzed by an algorithm, preferably performed on a computer. In one embodiment, the analysis begins from a probe of known sequence, for example a PCR primer that was used to generate the target nucleic acid. Alternatively, when no sequence is known, all probes in a selected positive pool are used as starting probes. The computer then identifies positive pools that contain probes having sequence that contiguously overlap the sequence of the starting probe(s). Preferably, the overlaps are identified in a K-1 fashion (i.e., probes that, when the contiguous region of overlap is aligned, have single nucleotide overhangs); however, in some embodiments (particularly involving accounting for false negative data, i.e., missing probes), K-2 or larger overlaps may be used, preferably K-2 overlaps. The standard K-mer analysis is described in U.S. Patent Nos. 5,202,231 and 5,525,464. If the pools are designed properly, each pool should contain at most one probe that potentially overlaps the starting probe. Ideally, only one pool is positive, and the next nucleotide is thus identified unambiguously. In this case, the process is repeated with the next overlapping sequence, and so on through the entire sequence.

Because some false negatives probes (i.e., missing probes) are expected, the sequence assembly algorithm must allow some K-2 or shorter overlaps. If only K-1 overlaps are used in every assembly step, then Fp must be reduced four fold by using four fold smaller pools. If sequence is assembled only using K-2 overlaps, the assumed false negative rate is 50%. Typical experimental false negative rates are more likely to be between 3 and 10%, so that many fragments of e.g., 20-50 bases may be assembled without using a K-2 overlap, and longer sequences may be assembled by allowing a limited use (e.g., one per 20 sequence extension cycles, on average) of K-2 overlaps.

Although sequence assembly when there are two consecutive false negative probes requires a K-3 overlap, that situation will happen very rarely (e.g., it may occur when a region of test DNA is inaccessible for hybridization). Thus, a percentage of false negative probes that is less than 10% false negative probes does not necessarily require a substantial increase in score redundancy per base (which in turn would require use of more and smaller sized pools).

Since the probes in a given pool are indistinguishable, a pool may give a positive result because a different probe in the pool hybridized to a different part of the target sequence. These "false" positives occur at frequency of 1/R. In such a case, two possibilities (or, in rare cases, even three or four) exist for the next nucleotide. The process is then repeated for each of these possible forks (also called branches). In most cases, no positive pools will overlap the sequence from the incorrect pool, and thus that fork will be discarded. However, at a probability of $1/R^2$, two consecutive incorrect positive pools may be found. In this case, the process is repeated for all positive forks. The probability of a fork extending b incorrect nucleotides is $1/R^b$, and thus false forks quickly disappear from the analysis, and the sequence may then be unambiguously determined. For R = 10, by the fourth incorrect position only a 1/10,000 chance of continuing remains. The normal problems of branch ambiguity found in SBH still occur, and these may be resolved by art-recognized methods (see, for example, Int'l Publication No. WO 98/31836, incorporated herein by reference). In particular, the pooling method facilitates the use of longer probes, which drastically reduces the number of branch points that need be resolved.

The probability of a fork in at least C cycles of overlaps (P_C) is $(4/R)^C$, and the average number of false forks of C or more cycles is $[(4^k)/R] \times P_C$. For R = 8, K = 12, and C = 20, $P_C(20) = 10^{-6}$, and the number of false C(20) or longer forks is $(16 \times 10^6)/8 \times (10^{-6}) = 2$. In this example, sequences of about 3200 nucleotides can be assemble in de novo SBH, and the pool size would be about $(16 \times 10^6)/(3200 \times 8) = 625$ probes; the number of pools would be 25,600.

R should be chosen such that it prevents branching out and looping, where a minimal false branch must be C = K, and must start and end with a positive probe; the other options are false branching out on many sites of normal sequence. For C=10, $P_C(10)$ (one way) $\approx 1/1000$, and the number of out branching in any of the two directions of a segment of correct sequence is [2x3200]/1000 ≈ 6 cases. By using length requirement and end sequence knowledge, these false sequences will be eliminated.

If all fragments assembled to about 20 cycles stop due to false negative probes (e.g., 5%), an overlap of K-2 must be used on about 200 different sites for a 3200 base target. Additional data from hybridization of an overlapping 7-mer probe will be sufficient for correct assembly in this case, and thus the algorithm will proceed with maximal overlap, then reducing by one nucleotide. Only branches longer than C can be used in further assembly to assure a small probability for false overlaps; in the K-2 step, the number of further used branches should be about 4-fold reduced compared to the number of scored K-mer probes.

The extensive occurrence of falsely scored pools of probes (either false positive or false negative) requires use of smaller pools. Experimental false positives must be included in the P/T ratio of 1/5. There may be experimental false positives because, e.g., the cumulative scores of several single mismatch and double mismatch probes may produce a strong enough signal to cause the pool to be scored as positive, and may dramatically increase the probability of scoring the sets of probes around K-2 or K-3 branch repeats.

The methods of the present invention utilizing pools of probes are particularly suited to Format 3 SBH, although the methods are useful in all formats. Format 3 SBH should be more accurate for scoring long probes because it uses two short, more discriminative probe modules and potentially adds the enzymatic specificity of ligase. Furthermore, Format 3 allows combinatorial scoring of large number of probes. In combination with the synthesis of pools for each of two modules, the synthesis of all 250 million possible 14-mers may be done in two sets of only 512 pools of 32 fixed and labeled 7-mers.

In Format 3, chips with a complete set of probes of a given (relatively short) length may be hybridized with targets and labeled pools of probes of short lengths. For example, a 6400 nucleotide target can be sequenced at a redundancy of R = 20 with 7 + 6 = 13-mers using 131,072 dots: 32 chips each containing 4096 pools of four 7-mers as fixed probes interrogated with 32 pools of 128 6-mers as labeled probes. A saving of 500-fold would thus be achieved.

The pooling method also provides other specific advantages in Format 3 SBH. In particular, positive and negative fixed probe information may be used to reduce the number of false positives of combined oligonucleotide sequences. For example, 16 arrays containing 1024 5-mers probed with 16 pools of 64 labeled 5-mers and 500 nucleotide targets allow scoring of all 10-mers in about 16,000 dot scores, with each score representing 64 10-mers. However, many of the 10-mers in each of the positive pools can be removed from further consideration because the labeled 5-mers that form them have not been found positive as fixed 5-mers. A related method is discussed below.

In an alternative embodiment, the pooling method can be used in conjunction with multiplex labeling to further reduce the number of array locations (oligonucleotide spots) required and/or to further reduce the number of hybridization reactions required. Multiplexing involves using more than one distinguishable label (such as different fluorophores, chromophores, EML, or radioactive labels, or mixtures thereof) to identify the pool, thus allowing different pools to be combined in a single location in an array or to be combined in a single solution for hybridizing with an array. For example, labeling each of four pools with one of four fluorescent dyes can allow the four pools to be mixed together in a single solution for hybridizing with an array, reducing the number of hybridization reactions required by an additional factor of four. Alternatively, the same principle of pools can be applied in conjunction with, e.g., beads or molecular labels for sequencing longer DNA.

In an alternative embodiment, the pooling method may use continuous score values for all positive hybridizations, instead of +/– calls. This embodiment would allow probabilities to be calculated for each fork.

Sequence assembly can also be carried out without assigning a "positive" or "negative" score to the probes (or pools of probes). In practice, this may be carried out by overlapping multiple (e.g., two or more, or three or more, or four or more, or five or more) consecutive probe sequences and calculating a cumulative score, or performing statistical analysis, for the overlapped sequence based on the scores of each of its constituent probe sequences. If all of the probes were positive probes, the sum of the scores should be very high. This method of overlapping using "continuous scoring" is advantageous because it is not possible to perfectly discriminate between the scores of full match and mismatched probes even under optimal hybridization and washing conditions. Thus, although on average the hybridization signals of full match probes are significantly different from the hybridization signals of mismatch probes, the actual signal distributions of full match probes and mismatch probes can overlap. In conventional SBH, an optimal threshold is determined, and probes with scores above this threshold are called "positive" while probes with scores below this threshold are called "negative". Thus, some full match probes will be falsely scored as negatives. In contrast, the continuous scoring method of overlapping allows these false negative probes to be incorporated into the assembled sequence despite the fact that their scores fell below the positive threshold.

The method of overlapping using "continuous scoring" improves the accuracy of sequence assembly by using, e.g. cumulative scores of multiple overlapping probes. For example, starting from a known primer or a true positive probe, all possible combinations of N (e.g., where N is 2, 3, 4, 5 or more) overlapping probes may be assembled and the cumulative score calculated for each overlapping combination. An optimal N (number of overlapping probes for which a cumulative score is calculated) is in the range from 3 to K-2, where K is the number of bases in the probe. The overlapping combination(s) with the highest cumulative scores are carried forward to the next base extension. If the overlapping combination is the correct sequence, the constituent full match probes should have very high scores and the cumulative score should also be high. The process is then repeated for the next base extension. Statistically significant combinations or sets of combinations determine the next base. The significance level may be calculated for each base relative to the other three bases.

Exemplary statistical analyses of the hybridization scores of overlapping probes include summation, multiplication, averaging, median scores, and maximum likelihood statistical analyses. For example, for any set of 3 or more (or 4 or more, or 5 or more, etc.) overlapping probes, a median score may be calculated. Alternatively, an average score of the overlapping probes can be calculated after removing the smallest and largest score to minimize the influence of potential false negatives and experimental or pooling false positives.

In another embodiment, a maximum likelihood statistical analysis may be applied as follows. For each probe from a group of overlapping probes, the probability that this probe is a perfectly matched probe may be calculated (e.g. from the distribution density of probes in the perfectly matched category). For other probes not in that group of overlapping probes, the probability of each probe being a specific type of mismatched probe (or any type of mismatched probe) may also be calculated. From these probabilities, a probability that the sequence defined by the group of overlapped probes is the correct sequence may be calculated by multiplying the probabilities of each of the overlapped probes.

In addition to the statistical analysis of scores of a group of overlapping probes, statistical analysis may also be performed on cumulative scores or on median or average scores or probabilities for various groups of overlapping probes representing different sequences. For example, starting from one known sequence and using 10 overlapping probes, 1,048,576 different groups of 10 overlapping probes may be formed. Each of the four possible bases that may be the next base extension of the known starting sequence is represented with 262,144 groups. If, for example, A is the correct base, several groups (usually the correct one and some with false probes especially at the end) will have a high cumulative, average or median score. The decision on which base extension of the starting sequence is the most likely may be obtained by calculating the median or average score of some number (2-30) of groups with the highest score for each base. In this analysis one or more bases may be determined that extend the starting sequence.

For assembling longer sequences by repeated determination of a single or more base extension of the starting sequence using overlapping probes that match with the previously determined or known sequence, only selected groups (rather than all groups) of overlapping probes from the previous cycle can be used. For example, only 16 or 64 groups with the highest score for each base may be used. Because only one or a few new probes will be added, some of those groups with the highest score will most likely have the new highest statistical value. This process may significantly expedite sequence assembly because only a few hundred groups of overlapping probes, instead of over one million groups of overlapping probes, need to be tested. The other option is to extend selected groups by a few overlapping probes; this leads to testing a larger number of groups but it may give more accurate statistics.

Optionally, 2-3 independent sets of 2-3 fold larger pools may be used for exponential reduction of false positives (0.1 x 0.1 x 0.1 for 10% false scored probes). Additionally, in some applications, it may be advantageous to have a single probe in more than one pool; however, the percentage of false negative probes may increase with this method.

The methods of the present invention utilizing pools can be used with other types of pools, such as redundant pools or binary pools. For example, in a combination of pooling and redundant pooling, two or more sets of pooled probes may be used. For each set, the probes are grouped into different pools, e.g., a probe that is grouped with one pool of probes in the first set will be grouped with a different pool of probes in the second set. Ideally, only probes that are positive in both sets are placed in the set of positive probes. Thus, although pooling false positives remain, fewer false positive probes are included in this set of positive probes used for sequence assembly. When continuous scoring is being used, each probe may be assigned the lowest of the two hybridization scores obtained from hybridization of the first and second sets.

Pools can be used with additional mapping information to allow assembly of longer sequences for a given probe length than specified in Table 3. In any SBH method, branch points produce ambiguities as to the ordered sequence of a fragment. In the assembly of relatively longer fragments, ambiguities may arise due to the repeated occurrence in a set of positively-scored probes of a K-1 sequence (i.e., a sequence shorter than the length of the probe). Additional mapping information may be used to order hybridization data where such ambiguities ("branch points") occur. For example, restriction mapping information can be used to map sequence subfragments.

In another embodiment, the sequence subfragments may be ordered by comparing the sequence of the subfragments to related sequences (e.g., a known sequence from a closely related species with over 80% sequence identity) and ordering the subfragments to produce a sequence that is closest to the related sequence. For example, according to Table 3, 15-mers should be used in order to assemble unambiguous 25 kb segments of genomic sequence. However, if additional mapping information is known, 12-mers may be used instead, resulting in 64-fold smaller pools. Although branching points will occur about every 200 bases, the assembled sequence subfragments can be mapped by matching them to known genomic sequence of a closely related species.

In yet another embodiment, primers for single pass gel sequencing through the branch points may be identified from the SBH sequence information or from known vector sequences, e.g., the flanking sequences to the vector insert site, and standard Sanger-sequencing reactions may be performed on the sample target nucleic acid. The sequence obtained from this single pass gel sequencing can be compared to the subfragments that read into and out of the branch points to identify the order of the subfragments.

Alternatively, the information needed to solve branching ambiguities may be directly provided by using a known reference gene sequence when assembling gene sequences from individual patient samples. When a reference sequence is known, the maximal read length in bases may be extended to approximately one tenth the number of probes used. To sequence human and other complex genomes (over 3 billion base pairs) as a single sample, about 70 billion 18-mer probes would be needed.

In addition, the number of tandem repetitive nucleic acid segments in a target fragment may be determined by single-pass gel sequencing. As tandem repeats occur rarely in protein-encoding portions of a gene, the gel-sequencing step will be performed only when one of these noncoding regions is identified as being of particular interest (e.g., if it is an important regulatory region).

h. Exemplary sequence assembly algorithm

The following sequencing algorithm is one way in which all nucleotide sequences consistent with a set of positive probes (PP) can be assembled. If the PP is obtained using pools, the PP set may optionally be "filtered" (as described below in Section i) or optionally "rescored" as described below, or both, before being used as the input PP set. A nucleotide sequence consistent with the input set of PP is composed for the most part of probe sequences from the PP. However, because false negatives are expected to occur, extension of assembled sequence must be allowed even if some probe sequences are "missing" from the PP in order to guarantee that the correct nucleotide sequence will be among the putative sequences generated.

In one exemplary embodiment, the sequencing algorithm can commence after the following fixed input parameters have been specified: a known 9 base primer (from which sequence assembly starts), a "cleaned" (or filtered) set of PP obtained as described immediately below, and preset parameters specifying the approximate length of the target nucleic acid sequence (MaxLength) and how many missed probe sequences (MaxMisses) and consecutive missed probe sequences (MaxConsecutiveMisses) can be allowed while sequencing (thus allowing, e.g., K-2 or K-3 overlapping of probe sequences despite some expected false negatives). MaxMisses may range from 1% to 10% but is preferably set to 5%. Even for modest values of MaxMisses (e.g., 8), the overwhelming majority of assembled sequences with that number of misses turn out to be incorrect. MaxConsecutiveMisses may range from 1 to 3 but is preferably set to 2. The MaxLength may be a fixed number or a range of numbers. For example, assembly of sequence from 5' to 3' can be done as follows:

1) At each position i (in this case meaning the i-th position after the primer sequence), the following 4 variables are updated:

(a) suffix9: the 9 consecutive bases before the current position i (i.e., the last 9 bases of the sequence that has been assembled so far);

(b) #misses: the total number of "misses" (missing probe sequences) within the sequence assembled so far;

(c) #cmisses: the number of consecutive misses (consecutive missing probe sequences) at the end of the sequence assembled so far.

(d) length (L): the length of the sequence assembled so far - 9 (nine is subtracted because the sequence started with a known 9-base primer)

2) When sequence assembly commences, initially the variables are set to:

i=1, suffix9=primer, #misses=0, #cmisses=0, L=0.

3) One base (one of A, C, G, or T) is temporarily added to the 3' end of suffix9 to make a 10-mer sequence. For convenience, this added base is referred to as "X", and the 10-mer created by addition of X to the end of suffix9 is denoted as "suffix9X".

4) X is added to the 3' end of the assembled putative sequence if suffix9X is in the PP set. Alternatively, X may be added to the 3' end of the putative sequence as a base from a missing (i.e. false negative) probe (the correctness of X can be verified with a later overlapping positive probe), provided that the total number of misses accumulated thus far in the sequence is less than the preset parameter MaxMisses, and the number of consecutive misses at the 3' end of the sequence is less than the preset parameter MaxConsecutiveMisses. This can be carried out as follows:

(a) If the sequence for suffix9X is in the PP, and if L < Maxlength - 9, X is added to the assembled sequence. (If L = Maxlength-9, then sequence assembly stops). Suffix9 is now updated to be the last 9 bases of suffix9X; L = L+1 (L is incremented by 1); #misses = #misses (# misses stays the same); and #cmisses = #cmisses (#cmisses stays the same).

(b) If suffix9X is not in the PP, if #misses < maxMisses, if #cmisses < MaxConsecutiveMisses, and if L < maxLength - 9, then X is added to the assembled sequence. Suffix9 is now updated to be the last 9 bases of suffix9X; L = L+1; #misses =

#misses+1 (#misses is incremented by 1); #cmisses = #cmisses + 1 (#cmisses is incremented by 1)

5) Steps 3 and 4 are repeated until the #misses reaches the MaxMisses, or until the #cmisses reaches the MaxConsecutiveMisses, or until sequencing stops (when L reaches Maxlength - 9).

At any i-th position, two or more possible "suffix9X"s may be present in the PP (particularly when pools are used). For example, suffix9X where X is A may be present in the PP and suffix9X where X is T may also be present in the PP. Both sequences would be held in memory as possible assembled sequences. In addition, when a missed probe is being allowed at an i-th position, X may be any one of A, C, G or T, so all four possible assembled sequences would be held in memory. Thus, multiple assembled sequences are typically kept in memory as "branching" possibilities until they are eliminated because they include too many missed probes. Each possible sequence can be called a node on a tree or graph. It is possible that multiple sequences can successfully assemble to the maxLength without violating the MaxMisses or MaxConsecutiveMisses limits; in this case, there will be multiple final putative assembled sequences. To reduce memory requirements, it is worthwhile to prune away useless nodes that can be shown not to lead to a node of MaxLength. Generally, this pruning method involves determining whether a node of length L leads eventually to a node of MaxLength; if not, then it is removed. The eliminated node's predecessors are also examined and are pruned if they lead to no other nodes except the node that was just eliminated. By applying this pruning recursively, all nodes that do not lead to a node of MaxLength are eliminated.

6) Each final putative assembled sequence is rechecked from 3' to 5'. Starting with the last position in the assembled sequence, the first base of the probe at the previous position is attached to the 5' end and the presence of this newly created 10-mer in the PP is checked. Typically, repeated 10-mers are excluded during this rechecking step, i.e. putative sequences are eliminated if the presence of the new 5' base would create a 10-mer that had already been "counted" as included in the assembled sequence.

i. Filtering Out False Positives

In order to reduce the computational requirements of the sequence assembly process described above, false positives may optionally be filtered by first overlapping sequences of small numbers of positive probes prior to assembling sequences of all of the positive probes into the complete putative target sequence. Each positive probe is overlapped with a small number (ranging from, e.g., K/2 to 2K) of other positive probes to provide an extended sequence in either direction, and only the extendible probes are kept in the set of positive probes that will be used to form the complete sequence.

The filtering algorithm removes false positives generated from any of the three most significant sources of false positives, i.e., single nucleotide mismatches, single nucleotide insertions, and single nucleotide deletions, as well as false positives due to target-independent probe-probe ligation. Filtering is particularly advantageous for SBH methods using pools of probes, because of the very high number of false positives generated by these methods.

In one embodiment, the set of positive probes (PP), e.g. 10-mers, which initially contains all probes in the positive pools may be "cleaned" by keeping only extendable probes (i.e., probes that overlap with other positive probes) in the set of all PPs, as follows:

1) Starting with the initial PP set, determine all possible 18-mers that can be constructed by overlapping 9 consecutive hypothetical 10-mers in the PP set. An exemplary 18-mer overlap is shown below:

```
GGTCTcccca         1
 GTCTCcccaa        2
  TCTCCccaag       3
   CTCCCcaagg      4
    TCCCCaaggc     5
     CCCCAaggcg    6
      CCCAAggcgc   7
       CCAAGgcgca  8
        CAAGGcgcac 9
```

2) Only those 10-mers that appear as the middle 10-mer of one of these 18-mers (i.e. the fifth 10-mer among the 9 overlapped 10-mers) and that were originally in the PP set are kept in the PP set.

3) In alternative embodiments, the stringent requirement that there be 9 consecutive 10-mers may be relaxed by allowing overlap despite 1 or 2 missing 10-mer probes. For example, one can allow the 18-mers to be assembled from, e.g., 7 or 8 non-consecutive 10-mers.

i. Exemplary Filtering Algorithm for Format 3

Use of pooling methods in Format 3 SBH provides special advantages in the sequence assembly process. False positive results can be eliminated, or at least greatly reduced, by using a filtering algorithm to build a set of "clean" positive K-mers, based on combining the information from the fixed probes and the labeled solution probes. The filtering algorithm may be performed by hand, but the quantity of data generated makes it preferable to perform it using a computer as described in detail below. The basic thrust of the filtering algorithm is to use the information from all overlapping probes to verify each position in a given K-mer. When each position is verified, the resulting K-mer is declared "clean" and is then stored in a table for subsequent input into the K-mer sequence assembly algorithm. The method for eliminating false positives is here exemplified in terms of using the information from 5-mer fixed probes and 5-mer labeled solution probes to generate clean 10-mers; however, one of skill in the art will recognize immediately that the same filtering algorithm can be easily adapted for fixed and solution probes of other lengths to generate clean K-mers of any desired length.

In the following illustration, the filtering algorithm is applied to data generated from 16 oligonucleotide pools, each containing 64 labeled 5-mer probes, that are used in conjunction with 16 different oligonucleotide arrays, each of which contains the complete set of 1024 fixed 5-mer probes in a spatially addressable array. Each possible 5-mer is present in one and only one pool, and the pools are constructed according the criteria discussed above. The labeled pools of probes are ligated to the fixed probes in the presence of a target nucleic acid, following standard SBH Format 3 protocols, and the signal of each spot on each array is read. Control arrays without the target nucleic acid are probed in parallel. A "dirty" table of positive 5-mers is built by subtracting the control signals from the target signals, and taking as positive those above a selected threshold value. This initial table of 5-mers is based solely on the fixed probes, and ignores the sequences of the pooled probes. The table potentially contains at least some false positive 5-mers, in addition to correct 5-mers.

(a) For each of the positive 5-mer probes, the standard K-mer sequence assembly algorithm is used to extend the sequence for 5 nucleotides at the 3' end. For example, if a portion of a (theoretical) nucleic acid has a sequence complementary to

TGCTT GCCAC AGGTC TCCCC AAGGC GCACT

then the probe AGGTC should be positive. Extending the sequence 5 nucleotides using K-1 = 4 nucleotide overlaps generates the 10-nucleotide sequence:

```
AGGTCtcccc
 GGTCTcccca      1
  GTCTCcccaa     2
   TCTCCccaag    3
    CTCCCcaagg   4
     TCCCCaaggc  5
```

(b) If this is the correct extension, then the labeling probe for the fixed AGGTC probe should be tcccc. The pool that gave the positive signal for AGGTC is then checked to see if it includes tcccc. If not, the generated 10-mer is discarded as presumably incorrect, and step (a) is repeated with a different alignment of the starting 5-mer or with a different 5-mer. If the pool does contain the extended sequence, the generated 10-mer is further analyzed.

Note that the analysis, here and below, discusses checking only one pool for a positive signal. However, it is possible (indeed probable) that repeated 5-mers in the sequence will cause at least a few fixed probes to be positive with more than one labeled pool. This does not affect the analysis, which checks only that the fixed 5-mer is positive with the correct pool; it does not require that it be uniquely positive with that pool. Thus, the present discussion ignores these other positive pools as irrelevant. When used on the complete data set, the filtering algorithm will analyze all of the possible extensions of each positive fixed 5-mer, and thus all correct positives will eventually be verified and placed in the clean 10-mer table.

(c) The 5-mer table is then used to align forward for a sixth nucleotide, generating:

```
AGGTCtcccc
 GGTCTcccca      1
  GTCTCcccaa     2
   TCTCCccaag    3
    CTCCCcaagg   4
     TCCCCaaggc  5
      CCCCAaggcg 6
```

Again the data are checked to see that cccca is in the pool that generated the signal for fixed probe GGTCT. If so, the filtering algorithm continues; if not, the filtering algorithm is started again at step (a) with a new alignment or a new 5-mer.

(d) The verification is then repeated for all probes up through a total of 5 nucleotides beyond the starting 5-mer (which requires reading 9 nucleotides beyond the starting 5-mer), generating

```
AGGTCtcccc
 GGTCTcccca       1
  GTCTCccca a     2
   TCTCCccaag     3
    CTCCCcaagg    4
     TCCCCaaggc   5
      CCCCAaggcg  6
       CCCAAggcgc 7
        CCAAGgcgca 8
         CAAGGcgcac 9
```

(e) Through this iterative process, each nucleotide in the second half of the 10-mer has been verified. If each step is positive, the next step is to work backwards and repeat the steps of using 4-nucleotide overlaps from the 5-mer table to determine the preceding sequence and then to verify the presence of the 5-mer in the pool generating the signal for the positive 5-mer:

```
CAGGTctccc          1
 AGGTCtcccc
  GGTCTcccca        1
   GTCTCccca a      2
    TCTCCccaag      3
     CTCCCcaagg     4
      TCCCCaaggc    5
       CCCCAaggcg   6
        CCCAAggcgc  7
         CCAAGgcgca 8
          CAAGGcgcac 9
```

Again, the data are checked to see that the labeling probe ctccc is in the pool that gave a signal with fixed probe CAGGT. If so, the filtering algorithm continues with the next step; if not, it starts again at (a) with a new alignment or a new starting probe.

(f) The verification is next repeated backward through a total of 9 steps:

```
GCTTGccaca                  9
 CTTGCcacag                 8
  TTGCCacagg                7
   TGCCAcaggt               6
    GCCACaggtc              5
     CCACAggtct             4
      CACAGgtctc            3
       ACAGGtctcc           2
        CAGGTctccc          1
         AGGTCtcccc
          GGTCTcccca        1
           GTCTCcccaa       2
            TCTCCccaag      3
             CTCCCcaagg     4
              TCCCCaaggc    5
               CCCCAaggcg   6
                CCCAAggcgc  7
                 CCAAGgcgca 8
                  CAAGGcgcac 9
```

If all labeling probes are permissible (i.e., the fixed probe is positive with the correct pool), then the 10-mer AGGTGtcccc is put into the clean 10-mer table. Preferably, if any of the predicted probes are found not to be correct, then AGGTGtcccc is not put into the table. Use of the filtering algorithm results in each position in the 10-mer being verified by 10 overlapping probes before it is added to the clean 10-mer table. In certain situations, however, such as when the data contains many false positives, it may be preferable to rely on a threshold number of permissible probes, preferably at least about 50%, more preferably at least about 75%, and especially preferably at least about 90%.

The resulting final clean 10-mer table should contain all true 10-mers, as long as the data contain no false negatives. Once the table of clean 10-mers has been generated, the K-mer sequence assembly algorithm is used to assemble the 10-mers into the complete sequence of the target DNA. As will be apparent from review of the above filtering algorithm, however, it cannot be fully extended to cover the starting and ending nucleotides. Therefore, the first 9 and the last 9 nucleotides are given special treatment. For the first 9 nucleotides, the basic filtering algorithm is repeated, working backward. As much data as possible is generated for each nucleotide in the sequence; however, each earlier nucleotide is less completely verified than the following nucleotide. Thus, the 9th nucleotide is read only 9 times, the 8th nucleotide only 8 times, and so on. The first nucleotide is read only once. It is identified by comparing the ligation signals for the four

10-mers that overlap nucleotides 2-10 and vary only at position 1. The one with the highest signal is chosen as the correct first nucleotide. For the example sequence, positions 2-10 are determined to be GCTTGCCAC. The signals for AGCTTGCCAC, TGCTTGCCAC, CGCTTGCCAC, and GGCTTGCCAC are compared, and the strongest signal (which should be TGCTTGCCAC) is used to determine the first nucleotide. A similar approach is applied to the last 9 nucleotides, proceeding in the forward direction. In addition, the last nine positions can be further checked using only the sequence of the fixed 5-mer probes (ignoring the pooled solution probes), thus strengthening the determination of these positions.

ii. Computing Device for Filtering

A diagram of a nucleic acid sequencing system including a computing device 200 capable of implementing the teachings of the present invention is illustrated in FIG. 2. The computing device 200 may be a general purpose computer programmed to implement the method and/or apparatus of the present invention, or the computing devic‹ 200 may be an application specific device designed to implement the method and/or apparatus of the present invention as is well known to persons of ordinary skill in the art. A controller 202 in the computing device 200 may include a data memory 204, such as a random-access memory and/or a disk drive, a program memory 206, which may be in the form of a read-only memory (ROM), and a microprocessor 208, all of which may be interconnected by an address/data bus 210. In one embodiment, the program memory 206 electronically stores a computer program that implements all or part of the method described below, and the program is executed by the microprocessor 208. The program memory 206 may be loaded from a fixed memory device such as a hard drive, or the program memory 206 may be preloaded with firmware as is well known to persons of ordinary skill in the art. Some of the steps described in the method below may be performed manually or without the use of the computing device 200.

A transmitter and receiver in the form of a conventional input/output (I/O) circuit 212, such as a modem for example, typically couples the controller 200 to external devices. An input device 214 such as a keyboard, mouse, and/or optical scanner may be connected to the I/O circuit 212 via a line 216 for entering data and commands into the controller 202. Further, an output device 218, such as a display or printer, may be connected to the I/O circuit 212 to receive data via a line 220 to generate visual displays of data generated during operation of the computing device 200.

A flowchart of one possible process 300 of nucleic acid sequencing by hybridization using pools of probes is illustrated in FIG. 3. The process 300 can be implemented by a human operator and/or the computing device 200 in accordance with the teachings of the present invention. In one embodiment, the programmed steps performed by the computing device 200 are executed by the controller 202. Generally, the process 300 employs a complete set of fixed probes in conjunction with a complete set of labeling probes. However, the labeling probes are combined into a relatively small number of pools. Ligation information from a reduced number of experiments is then processed to determine a target sequence.

When the process 300 is initiated, a researcher and/or an automated testing apparatus performs a sequencing by hybridization Format 3 experiment for each of the pools of labeling probes (step 302). In this example, the fixed probes are 5-mers and the labeling probes are 5-mers. Accordingly, each experiment contains 1024 fixed probes and 1024 labeling probes (i.e., 4^5). Of course, a person of ordinary skill in the art will readily appreciate that any number of fixed probes and any number of labeling probes may be used in the scope and spirit of the present invention. Further, in this example, sixteen pools of probes with sixty-four labeling probes per pool are used. However, it is understood that any number of pools may be used. Still further, a person of ordinary skill in the art will readily appreciate that SBH Format 1 and/or SBH Format 2 could also be used in the scope and spirit of the present invention.

When all sixteen experiments are completed, a certain number of fixed probes will indicate that a ligation has occurred (e.g., 286 of the 1024 fixed probes fluoresce). Certain fixed probes may "hit" (i.e., display a true signal) when the first pool of labeling probes is used. Other fixed probes may "hit" when the second pool of labeling probes is used, and so on through all sixteen pools of labeling probes. However, at this point in the process 300 it is unknown which of the sixty-four labeling probes in a particular pool actually caused the ligation to occur.

At step 304, a 5-mer table is preferably created. The 5-mer table documents the results of the experiment by placing each fixed probe with a true signal in a first column (e.g., "CTCGA") and the associated labeling probes in a second column (e.g., "pool 7" or "TCCGG, GTCTC, CGTTC, ..."). Of course, any data structure may be used for this purpose. In this example, if a particular fixed probe is associated with one pool, there will be sixty-four labeling probes in the second column. If a particular fixed probe is associated with two pools, there will be 128 labeling probes in the second column etc.

Once the 5-mer table is created, a 10-mer table is determined from the 5-mer table at step 306 (described in detail below). Of course, a person of ordinary skill in the art will readily appreciate that a table of any size oligomers (e.g., 15-mers) may be created using the teachings of the present invention. Subsequently, a primary sequence is determined form the 10-mer table at step 308 in a known manner. The sequence ends are treated separately at step 310.

One way of implementing process 306 to determine the 10-mer table from the 5-mer table (shown schematically in FIG. 3) in accordance with the teachings of the present invention is illustrated in the flowchart of FIG. 4. Many other methods of arranging similar steps may also be used to achieve the same result of determining the 10-mer table from the 5-mer table. In one embodiment, the steps are performed by an human operator and the controller 202. The process 306 begins at step 402 by retrieving the first fixed probe 5-mer from the 5-mer table created in step 304. Subsequently, at step 404, the process 306 also retrieves the first labeling probe 5-mer from the pool associated with the current fixed probe 5-mer (e.g., the first possible match out of sixty-four labeling probes in the pool). The fixed probe 5-mer and the labeling probe 5-mer are then combined into a candidate 10-mer at step 406.

The candidate 10-mer is then tested against the other data acquired to determine if should be placed in the 10-mer table or discarded. At step 408, an index variable is initialized to "2" in order to point to the 5-mer starting one base in from the end of the candidate 10-mer. For example, if the candidate 10-mer is "CTCGATCCGG", the 5-mer defined by [N->(N+4)] is "TCGAT" (i.e., C***TCGAT***CCGG). Similarly, when N=2, the 5-mer defined by [(N+5)->(N+8), X] is "CCGGX" where X is unknown at this point in the process (CTCGAT***CCGG***). The "question" for the data is whether the 9-mer "TCGATCCGG" makes sense and, if so, what is the value of X?

In order to answer these questions, the process 306 looks in the pool associated with the fixed probe 5-mer "TCGAT" for a labeling probe that starts out with "CCGG" at step 410. If a labeling probe starting with "CCGG" is not found in the pool associated with the fixed probe "TCGAT", then the process 306 discards this candidate

10-mer (i.e. the process 306 does not store this 10-mer in the 10-mer table). In such an instance, the process 306 determines if there are more labeling probes associated with the current fixed probe at step 412. If there are more labeling probes to test, the process 306 loops back to step 404 to retrieve the next labeling probe from the associated pool and the process repeats. In other words, if the process 306 just tested the first labeling probe of sixty-four possible labeling probes, then it moves on to the second labeling probe.

In this example, if a labeling probe starting with "CCGG" is found in the pool associated with the fixed probe "TCGAT" at step 410, then the process 306 extends the 10-mer by tacking X onto the end of the candidate 10-mer and steps forward one base at step 414. X is determined to be the fifth base in the labeling probe starting with "CCGG". If more than one labeling probe starting with "CCGG" is found, the process may "fork" until a failure is found, thereby eliminating all but one of the labeling probes.

The process 306 in this example preferably steps forward until the candidate 10-mer fails (see "No" branch of step 410) or nine forward steps have been successfully completed as determined by testing N at step 416. If nine forward steps are successfully completed, the process 306 stores the candidate 10-mer in the 10-mer table at step 418. Of course, a person of ordinary skill in the art will readily appreciate that any number of forward steps may be used as a threshold for storage of a candidate 10-mer in the 10-mer table. Further, any number of reverse steps may be performed in a similar manner.

When all of the labeling probes associated with a particular fixed probe have been tested, the process 306 determines if there are more fixed probes in the 5-mer table at step 420. If there are more fixed probes to test, the process 306 loops back to step 402 to retrieve the next fixed probe from the 5-mer table and the process repeats. In other words, if the process 306 just tested the first fixed probe with a true signal, then it moves on to the second fixed probe with a true signal. When all fixed probes and associated labeling probes have been combined into candidate 10-mers and tested, the 10-mer table is complete, and the process 306 exits. Candidate 10-mers can also be tested in parallel rather than sequentially.

j. Rescoring: Allowing Mismatched Probes to Vote for Full Match Probes

SBH sequence assembly may be improved by optional score recalculation methods that involve assigning a new score (or a "rescore") to each probe by analyzing scores of probes which are one or two bases different (i.e., have a single mismatch or a double mismatch compared to the probe of interest). This is especially advantageous when pools of probes are used because otherwise each of the probes in a pool are assigned the same score. For example, score recalculation using scores of single mismatch probes may be carried out in format 3 SBH utilizing 10-mer probes by determining a value "P" as follows:

1) First, for each 10-mer probe (designated, e.g., probe X), a value "S" is calculated.

(a) At each position "i" of probe X (e.g., i may be 0 through 9), scores of the four probes that could vary at this position (i.e., probe X and the three possible single mismatch probes at position i) are examined and the standard deviation S(i) of the four scores is calculated.

(b) S = max[S(i)] as i ranges from 0 to 9 (i.e., S is set to be the largest of the 10 standard deviations S(i) obtained in step 1(a)).

2) Next, for each probe X, the P value is calculated from the original score and S as determined in step 1:

(a) Slope = (the original score of X) / (S for X)

(b) P = 1.0 / abs (Slope - 2). The slope will be very close to 2 for a full match probe, so P will be very high for a full match probe.

Although S itself could be used as a rescore (S should be very high if X is a full match probe), preferably the P value is used as the rescore.

Alternatively, scores of single mismatch and double mismatch probe can be taken into account by calculating a rank R for the probe X as follows. The P values for a set consisting of: X, all of X's 30 single mismatches, and all of X's 405 double mismatches are examined. The 436 probes in this set are sorted by their P values. R (also called the slope-rank) is then set to be the rank of X among the 436 probes after this sort has taken place. If X is a full match probe, it is expected to be the highest ranked (where, e.g., number 436 is the highest ranked). R may then be used as the rescore.

Yet a third alternative for recalculating the score of probe X involves calculating the sum of the scores of the 6 overlapping 5-mers that constitute the sequence of probe X. For example:

1) For each 5-mer, a "Y" value is determined where Y is the sum of the original scores of all probes or all pools of probes in which the 5-mer was the fixed probe.

For example, if 16 pools were used, Y is the sum of all 16 pool scores in which that 5-mer was the fixed probe.

2) Each 10-mer (probe X) gets a score based on the sum of the Y values of its 6 overlapping constituent 5-mers. If X is a full match probe, each of the 6 scores should be large, since each of the constituent 5-mer should itself have hybridized to target as part of another 10-mer. The sum of the Ys can then be used as a the rescore of X.

Rescoring, although optional, is advantageous for determining sequence using pools. For example, rescoring allows removal of 50-80% of positive 10-mer probes from the initial PP without causing false negatives. Rescoring may also be used as an improvement to other known SBH methods, including conventional format 1, 2 or 3 SBH, to provide better discrimination between full match probes and mismatch probes than is provided by the original scores based on hybridization intensity signals.

k. Determining Likelihood Scores for Probes or Assembled Sequences

SBH sequence assembly methods can also be improved by employing a method for determining the likelihood, or probability, that a putative (or candidate) nucleotide sequence, consisting of overlapping sequences of informative regions of probes, is the correct nucleotide sequence of the target nucleic acid.

For each putative nucleotide sequence, the probes (or pools of probes) are divided into two or more categories, e.g., full match probes, single mismatch probes, single mismatch probes where the mismatch occurs at the labeling end, G/T mismatched probes, etc. The probes or pools of probes are placed in a category by assuming that the putative nucleotide sequence is correct and comparing the putative sequence to the probe sequence. When only two categories are used, probes are placed in either the full match or the mismatch category.

For each category of probes, the hybridization signal intensity for each probe (or pool of probes) in that category is plotted as a distribution density, e.g., the x-axis is intensity value and the y-axis is the relative frequency (density) of that intensity value within that category. In a rough approximation, for example, the intensity values could be divided into small intervals and the frequency for each interval could be calculated. Each probe (or pool of probes) within the category is then assigned a probability value that is equal to the density value corresponding to the probe's intensity value. For example, if a probe is in the full match category and had an intensity value of

10,000, and if 10 probes in the full match category had an intensity of 10,000-11,000, then the probability value for this probe is 10 divided by the total number of probes in that category.

After each probe (or pool of probes) has been assigned a probability value, the multiplication product of probability values for all of the probes (or pools of probes) is determined (i.e., the values are all multiplied together). This multiplication product is now the "likelihood" of the putative nucleotide sequence, and the sequence with the maximum likelihood has the highest probability of being the correct sequence.

I. Sequencing Applications Using Pools of Probes

Although discussed in terms of de novo sequencing, one of skill in the art will recognize that the pooling method can be used for sequencing even longer targets, if they are similar (preferably >95% similar) to known reference sequences. Specific pools may be used to generate clones or DNA fragment signatures, recognize sequences, score known POLYMORPHISMS and perform others types of DNA sequence analyses.

The pooling method is also advantageous in sequencing 50-150 kb bacterial artificial chromosomes (BACs) and other long clones, using 16-17-mer probes. Such a method would preferably utilize about one million pools with 4000-16,000 probes per pool. Pools with these high numbers of probes would require higher sensitivity of detection and more efficient high density arrays. With Format 3, the one million pools can be prepared by synthesis of 1000 fixed pools of 8-mers or 9-mers and 1000 labeled pools of 8-mers, each having about 100 probes. Alternatively, other combinations, such as 10,000 fixed and only 100 labeled pools, may be used. For preparing the large pools, smaller pools may be synthesized and then pools of pools prepared. Additionally, the availability of a reference sequence from a similar species may allow sequencing of long clones with shorter probes, such as 12-mers, using 250,000-500,000 pools containing 32-64 probes, without the need for PCR. This example demonstrates particular advantages of pools in dealing with large number of long probes and long targets.

The sequence information obtained may be applied to the efficient identification and sequencing, including resequencing, of one or more nucleic acid samples. The procedure has many applications in nucleic acid diagnostics, forensics, and gene mapping. It also may be used to discover mutations and POLYMORPHISMS including single nucleotide POLYMORPHISMS (SNP) in a selected portion of a gene, the full gene, the entire genome, or a subset of the genome, to identify mutations responsible for genetic disorders and other traits, to verify the identity of nucleic acid fragments, to identify infectious agents, specific strains thereof, or mutants thereof (including viruses, bacteria, fungi, and parasites), to identify nucleic acid in samples for forensic purposes or for parental identification, to assess biodiversity and to produce many other types of data dependent on nucleic acid sequence. See, e.g., Examples 19 through 27 of Int'l Publication No. WO 98/31836 published July 23, 1998 and WO 99/09217 published February 28, 1999, both of which are incorporated herein by reference.

In addition, obtaining information about the degree of hybridization exhibited for a set of only about 200 oligonucleotides probes (about 5% of the effort required for complete sequencing) defines a unique signature of each gene and may be used for sorting the cDNAs from a library to determine if the library contains multiple copies of the same gene. By such signatures, identical, similar and different cDNAs can be distinguished and inventoried. See, e.g., Example 34 of WO 99/09217 published February 28, 1999, incorporated herein by reference.

With improved engineering of miniaturized devices, appropriate resolution and sensitivity for detecting hybridization signals, appropriate specificity in discriminating full match probes from mismatched probes, and use of pools with multiplex labeling, whole bacterial artificial chromosomes (or even bacterial genomes using 15-mers and providing mapping information for 1kb subfragments) may be routinely *de novo* sequenced in one reaction.

A specific hybridization scoring method may be employed to define the presence of mutants in a genomic segment to be sequenced from a diploid chromosomal set. Two variations are where: i) the sequence from one chromosome represents a known allele and the sequence from the other represents a new mutant; or, ii) both chromosomes contain new, but different mutants. In both cases, the scanning step designed to map changes gives a maximal signal difference of two-fold at the mutant position. Further, the method can be used to identify which alleles of a gene are carried by an individual and whether the individual is homozygous or heterozygous for that gene.

Scoring two-fold signal differences required in the first case may be achieved efficiently by comparing corresponding signals with homozygous and heterozygous controls. This approach allows determination of a relative reduction in the hybridization signal for each particular probe in a given sample. This is significant because hybridization efficiency may vary more than two-fold for a particular probe hybridized with different nucleic acid fragments having the same full match target. In addition, different mutant sites may affect more than one probe depending upon the number of probes. Decrease of the signal for two to four consecutive probes produces a more significant indication of a mutant site. Results may be checked by testing with small sets of selected probes among which one or few probes selected to give a full match signal which is on average eight-fold stronger than the signals coming from mismatch-containing duplexes.

i. Exemplary Mutation Identification Algorithm Using Likelihood Scores

The likelihood score provided by the algorithm described above, which determines the probability that a putative nucleotide sequence is correct, may also be utilized as follows to identify mutations in a gene. For each base at position i within a reference gene sequence, there are 7 possible mutations (total of 8 sequence variants including the reference sequence) : 3 possible substitutes, 3 possible insertions, and a possible deletion. For example, if the reference sequence is CGT, at the second position the "G" may be substituted with an A, C or T (giving rise to the sequences CAT, CCT or CTT, respectively), there may be an insertion of A, C, T or G before the G (giving rise to the sequences CAGT, CCGT, CTGT or CGGT, respectively), or the G may be deleted entirely (giving rise to the sequence CT).

In one exemplary embodiment, the mutation identification algorithm may be carried out using the following preset parameters which were empirically determined: threshold 1 (typically 0.995) and threshold 2 (typically 0.999).

1) For each position "i", the likelihood of the reference sequence and the likelihood of each of the 7 possible mutations at the i position are examined and the target sequence is determined to be (or "called") the reference sequence if the likelihood of the reference sequence is significantly higher than the sum of the likelihoods of the 8 sequence variants. This may be carried out as follows:

(a) The i position of the reference sequence is replaced with each of the 7 possible mutations and the likelihood of each mutation is calculated. The likelihood of the reference sequence itself is also calculated.

(b) The likelihood for the reference sequence is divided by the sum of the likelihoods of all 8 sequence variants (which includes the reference sequence). If this ratio (reference to sum of mutations) value is greater than Threshold 1, and preferably if the ratio of the second largest likelihood to the third largest likelihood is less than 10^7, then the i-th position is called the reference sequence.

2) If the ratio obtained in step 1(b) is not greater than Threshold 1, then the largest and second largest likelihoods are compared as follows:

(a) S1 = the largest likelihood (among the likelihoods of the reference sequence and likelihoods of the 8 possible sequence variations). S2 = the second largest likelihood among these 10 values. SH12 = the likelihood that there is a heterozygote mutation with the two sequences that have the likelihoods S1 and S2.

(b) If $S1 / (S1+S2+SH12) >$ Threshold 2, then the target sequence is called the S1 sequence. (If S1 is not the reference sequence, S1 is considered to be a homozygous mutated sequence.)

(c) If $SH12 / (S1+S2+SH12) >$ Threshold 2, then the target sequence is called a heterozygote of S1 and S2 sequences.

3) If step 2 does not provide enough information to call the target sequence (i.e., neither $S1/(S1+S2+SH12)$ nor $SH12/(S1+S2+SH12)$ is greater than Threshold 2), then the i and i+1 positions (where i+1 is denoted "j") are examined together as follows:

(a) The i and j positions of the reference sequence are replaced with each of the possible combinations of 8 sequence variants and the likelihood of each double mutation is calculated, as well as the likelihood of the reference sequence.

(b) The likelihood for the reference sequence is divided by the sum of the likelihoods of all possible double mutations. If this ratio (reference to sum of double mutations) value is greater than Threshold 1, the target sequence at positions i and j is called the reference sequence (because the likelihood of the reference sequence is significantly higher than the sum of the likelihoods of the double mutations).

(c) If the ratio obtained in step 3(b) is not greater than Threshold 1, then the largest and second largest likelihoods are compared as follows:

(i) D1 = the largest likelihood (among the likelihoods of the reference sequence and likelihoods of all possible double mutations). D2 = the second largest likelihood among these values. DH12 = the likelihood that there is a heterozygote double mutation with the two sequences that have the likelihoods D1 and D2.

(ii) If D1 / (D1+D2+DH12) > Threshold 2, then the target sequence is called the D1 sequence. (If D1 is not the reference sequence, D1 is considered to be a homozygous mutated sequence.)

(c) If DH12 / (D1+D2+DH12) > Threshold 2, then the target sequence is called a heterozygote of D1 and D2 sequences.

4) If step 3 does not provide enough information to call the target sequence (i.e., neither D1/(D1+D2+DH12) nor DH12/(D1+D2+DH12) is greater than Threshold 2), then the i+2 position may be examined, if desired (and then i+3 and so on), or there may be a "no call" result because of insufficient information to identify clearly the sequence at position i or j.

5) Steps 1-4 may be repeated until the end of the target sequence is reached.

EXAMPLES

EXAMPLE 1: COMPUTER SIMULATION USING POOLS OF 10-MER PROBES IN FORMAT 3 SBH

Computer simulations were used to test the described methods. The simulations used Format 3 SBH with 16 pools of 64 5-mers or 32 pools of 32 labeled 5-mers and 16 or 32 full arrays of 1024 fixed 5-mers, respectively. The pools were generated to satisfy various constrains: no reverse complements, no shifts where the first four nucleotides of one probe match the last four nucleotides of the other probe, no pairs of probes in a pool with single nucleotide difference, and minimized number of pairs with only two differences. The simulations tested different randomly generated 300 nucleotide and 1000 nucleotide targets. To simulate actual conditions, some simulations contained randomly generated 30-100% false positive scores.

The simulated raw hybridization data was then used to generate sequence information, according to the following algorithm: For each fixed 5-mer that is positive with a given pool, all possible 10-mer combinations of that fixed 5-mer and all labeled probes from that pool were generated. Each 10-mer so generated was then extended through all 9-nucleotide overlaps with the other generated 10-mers. The 11-mers thus formed were then extended further if any 10-mers matched with the growing end. The design of the experiment was such that false assembles usually grow only for few cycles. Where necessary, some ambiguities in the labeled probes were resolved based on the presence or absence of a corresponding positive results with the fixed probes. In almost all tested cases, all wrong sequences could be eliminated as short, cyclical, or inconsistent with positive/negative results on the fixed probes.

More specifically, two different computer programs were used: BuildFalse (for data sets with no errors or false positives only) and BuildMMult (for data sets with both false positives and false negatives). The code for both programs is found in Appendix 1. The programs were tested with a 300 nucleotide target sequence, r300 (Appendix 2), "probed" with two different sets of pools of 5-mer probes, D16 and DN16 (Appendix 3). The computer programs generated the expected hybridization data, including false positives and false negatives where applicable, then attempted to regenerate the sequence from the simulated hybridization data. Six files starting with r300 and finish with .out are simulation results.

All simulations were carried out on the same r300 sequence. Six simulations generated six output files, presented in Appendix 4. The simulation output files are named according to the pattern "r300.x.out", where x describes nature of the simulation: 0.0 represents no errors; 100.0 represents 33% false positives and 0% false negatives; 300.0 represents 100% false positives and 0% false negatives; and 100.15 represents 33% false positives and 5% false negatives. Additionally, the two files with "DN16" in the name used the DN16 set of pools; all others used the D16 set of pools (see Appendix 3). The output files contain (1) a listing of the positive combinations of fixed probes and labeled pools; (2) the stepwise creation of overlaps of increasing lengths; and (3) all solutions of the expected length. In each simulation, one solution was correct and all the other solutions had differences only at the ends, which can be recognized by known primer sequences.

Part (2) of the output files demonstrates the assembly of the sequence by creating all possible 10-mers from all combinations of positive fixed probes with each member of the pool of labeled probes. These 10-mers were then combined into all possible 11-mers by overlapping the positive 10-mers. Next, all the 11-mer blocks were combined over 10 nucleotides, effectively creating 12-mers, in what was equivalent to two steps of adding 10-mers with overlap 9. The process was then repeated with successively longer blocks and overlaps until all 300 nucleotide sequences were generated.

In the case of simulations containing false negatives, 11-mers were also created using 8-nucleotide overlaps in the 10-mer probes in the first step, and then all 11-mers were further combined as in the cases with no false negatives. However, other more efficient options exist for handling false negatives. One option is to build all sequences from 11-mers created by overlaps of 9 or more nucleotides (as in the case with no false negatives), and then to combine these sequences (all or only sufficiently long ones) using end sequences of length 8 and shorter (note that 5% false negatives means an average assembly of 20 nucleotides before a missing probe is hit).

EXAMPLE 2: ANALYSIS OF SINGLE NUCLEOTIDE POLYMORPHISMS USING POOLS OF PROBES

Format 3 SBH was carried out using a complete set of 1,048,576 10-mer probes scored for full match hybrids in a DNA sample, by hybridizing and ligating 16 pools of 64 labeled 5-mers on 16 replica arrays containing the full complement of 1024 attached 5-mers. Different DNA targets 100-220 bases in length were successfully sequenced using this procedure.

A 135 bp fragment of the human cytochrome P 450 (CP450) gene with 67.4% G+C content (corresponding to positions 3358-3492 of the sequence deposited under Genbank Accession No. CP450 CYP206) was prepared from genomic DNA using PCR. The CP450 gene has an A/G polymorphism at position 109 of the fragment, and the DNA sample used was heterozygous for this polymorphism, so that the DNA obtained from the sample had both possible sequences. One primer was phosphorylated, to allow degradation of that strand by lambda exonuclease (GIBCO BRL, used according to supplier's instructions) after the PCR product was obtained. DNAse I (GIBCO BRL, used according to supplier's instructions) was used to fragment the resulting single stranded DNA into fragments of about 20-50 bases. A separate hybridization reaction was carried out for each of 16 pools that each contained 64 different probes. For each hybridization reaction, 1 pmol of target DNA, 5 pmols of each of the 64 labeled probes in the pool, and 25 μl of ligation reaction containing 100 units of ligase (New England Biolabs) were added to the hybridization chamber with a complete array of 1024 fixed probes. The hybridization/ligation reactions were carried out for 30 minutes at room temperature in ligase buffer with 10% PEG. The unincorporated labeled probes were removed by a thorough wash with 2X SSPE, 1% sarcosine at 80°C for one hour. The 16 pools were scored in sixteen hybridization chambers using total of 16 picomols of target (about 100 μl of PCR product). Images were obtained using a General Scanning reader and a hybridization intensity signal was determined for each of the 16,000 spots (each representing a pool of 64 10-mers). Absolute hybridization signals were normalized (by dividing the score by the median score of its unit array) to have the same median value in all array units to avoid experimental differences.

The normalized signals were sorted by descending value, and based on past experience, the top 1200 signals were declared to be putative positive pools. That number of signals represented over nine-fold more positive pools than the expected 126 positive pools and thus included many more false positive pools than expected. However, due to variations in the signal of full match probes and mismatch probes, that number of false positive pools had to be included to assure that about 95% or more of the true positive pools were included in the assembly process. This was especially important because the target DNA contained a heterozygote site, and the 20 10-mers that bound to that heterozygote position would be expected to hybridize to only half of the targets and thus provide only half of the hybridization signal of other positive probes. The P/T ratio in this case was 1200/16384, about 1/14. About 76,800 10-mers were used in the assembly process.

The algorithms for sequence assembly may be further optimized to properly balance between false positive and false negative scores. Even with significant improvements in discrimination the distributions of hybridization signals from full match probes and from mismatch probes may still overlap to some extent. In the current Format 3 protocol, the average signal ratio of a full match hybrid to a single mismatch hybrid obtained using a short synthetic target is over 20-fold. For the CP450 experiment described in this example, in the top 200 signals there were only 96 out of the predicted 120 full match probes (80%). In order to select 90% of the predicted full match probes (12 additional scores), pooled probes with the top 1000 scores had to be used in the assembly process. In order to select 95% of the predicted full match probes (6 additional scores), pooled probes with the top 1200 scores had to be used in the assembly process.

After the pooled probes with the top 1200 pool hybridization scores were selected as the initial positive probe set, sequence assembly was conducted generally as described above. First, a "clean" positive probe set was selected by overlapping each probe with 8 other probes (4 in each 5' or 3' direction) using the filtering algorithm described above. Only one K-2 overlap (i.e., one missed probe in the 9 probe overlap) was allowed. This optional filtering step was used to simplify full length sequence assembly computations and resulted in an 8.6-fold reduction in the number of probes in the positive probe set.

The "clean" positive probe set was used to assemble overlapping sequences while allowing for 2 consecutive missed probes (i.e. K-2 or K-3 overlaps). The sequence assembly program found 30,313 candidate sequences that were the exact target nucleic acid length of 135 bp and that started and ended with the known primer sequences. The program found 138,698 candidate sequences that were within ± 4% of the expected length (ranging from 130-140 bp) and that started and ended with the known primer sequences.

The rescoring procedure described above was used to improve the determination of the correct target nucleic acid sequence. The P values of the probes were determined as described above and the candidate sequences were ranked by summing the rescores for all of the overlapping probes constituting the candidate sequence. After rescoring, the two correct solutions (including the heterozygote site) ranked first and second. The errors in the incorrect sequences were predominantly deletions and exchanges on K-2 overlap sites.

Thus, the use of pools of probes in Format3 SBH demonstrated that the methods of the present invention could be successfully used for de novo sequencing of a difficult 135 bp fragment of CP450 gene characterized by 67.4% C+G content. SBH was able to correctly identify an A/G heterozygote site, even though the 20 "positive" probes that hybridize to this position should have on average a 2-fold lower signal than other positive probes. Using the same approach, sequences of other targets including a 100 bp fragment of human p53 gene (characterized by G+C content of 62.0%) and a 198 bp fragment of the human apolipoprotein B gene (characterized by G+C content of 48.0%) were also correctly determined.

EXAMPLE 3: COMPUTER SIMULATION OF SBH USING 16,384 POOLS OF 1024 12-MERS TO ASSEMBLE 3.2 KB TARGET SEQUENCES

Simulation experiments with pools of 12-mers demonstrated the potential of this SBH approach to determine the sequence of very long target nucleic acid sequences of more than 10kb in length. (See Table 3 above). Several sequence assembly experiments were conducted with simulated data and pools of 10-mer or 12-mers to measure the minimal redundancy factor for different probe lengths and target nucleic acid lengths.

First, by using 16,384 pools of 64 10-mers, an Fp of 1/5 or less was observed to be sufficient to assemble 800 bases with a success rate similar to that using individual, unpooled probes (in about 90% cases, see Table 1). In addition, there were a large number of false positive pools (pools whose hybridization scores were considered positive despite the fact that they contained no full match probes), e.g., due to random experimental error. Since both true positive and false positive pools are included in the P/T ratio, the high number of false positive pools played a significant role in determining the P/T ratio.

In a second simulation, 16,384 pools of 1024 12-mers were used to test sequence assembly for 3.2 kb sequences that are expected to be uniquely assembled in >=90% cases (see Table 3). The selected parameters define a very restrictive Fp of 3200/16,384=0.196, slightly less than 1/5. One hundred different sequences were tested with the pools and with individual 12-mer probes. Of these 100 test targets, a unique correct sequence was produced in 87 cases, while 10 cases had 2 candidate sequences, 1 case had 3 candidate sequences, 1 case had 4 candidate sequences and 1 case had 12 candidate sequences. When individual, unpooled 12-mer probes were tested against the same 100 test targets, a unique correct sequence was produced in and 91 cases. The use of pools of probes provided less successful results compared to the use of individual probes in only 4 cases (4%). These results indicate that large pools (of 1024 probes) can efficiently determine the sequence of long target nucleic acids with a very low score redundancy (about 5 measurements per base), and demonstrate the computational feasibility of assembling target sequences as long as 3.2 kb.

The present invention is not to be limited in scope by the exemplified embodiments which are intended as illustrations of single aspects of the invention The foregoing specification and accompanying drawings is considered to be sufficient to enable one skilled in the art to broadly practice the invention. Indeed, various modifications of the above-described means for carrying out the invention which are obvious to those skilled in the relevant arts are intended to be within the scope of the following claims. All patents, patents applications, and publications cited herein are hereby incorporated by reference in their entireties for all purposes.

APPENDIX 1
Computer Programs

BUILD FALSE

```
#!/usr/leo/bin/perl

if (scalar @ARGV <4) { die "Need Pool, Seq, #False positives, #False negatives\n"; }

$FalsePos=$ARGV[2];
$FalseNeg=$ARGV[3];
open(POOL,$ARGV[0]);
print "Using pool $ARGV[0]\n";
$pools=0;
while(<POOL>)
{
        last if (/TotCost/);
        chop $_;
        @Probes=split(/[: ]/,$_);
        shift @Probes;
        shift @Probes;
        shift @Probes;
        if (scalar @Probes > 0)
        {
                @{$Pool[$pools]}=@Probes;
                foreach $probe (@Probes)
                {
                        $PoolInd{$probe}=$pools;
                }
                $pools++;
        }
}

print "Using sequence $ARGV[1]\n";
open(SEQ,$ARGV[1]);
$Seq="";
while (<SEQ>)
{
        chop $_;
        $Seq .= uc($_);
}

$Found=0;
```

```
undef(%Mers);
undef(@Solutions);
undef(%On);
foreach $i(0..length($Seq)-10)
{
        $fprobe=substr($Seq,$i,5);
        $lprobe=substr($Seq,$i+5,5);
        $pool=$PoolInd{$lprobe};
        $On{$fprobe}{$pool}=1;
}
foreach $prb (keys %On)
{
        foreach $pool (keys %{$On{$prb}})
        {
                print "True Signal: fp=$prb pool=$pool\n";
                push @Signals, new_signal($prb,$pool);
        }
}
$NumOn=scalar @Signals;

@char = qw( A C G T );
foreach $1(@char) {
foreach $2(@char) {
foreach $3(@char) {
foreach $4(@char) {
foreach $5(@char) {
        push @Probes, $1.$2.$3.$4.$5;
}}}}}
foreach $i (1..$FalsePos)
{
        $pool = int(rand($pools));
        $fixed = $Probes[rand(1024)];
        $On{$fixed}{$pool}=1;
        print "False positive Signal: fp=$fixed pool=$pool\n";
}

foreach $i (0..$FalseNeg-1)
{
        $tmpSignal=$Signals[$i];
        $randPos = $i + int($NumOn);
        $Signal=$Signals[$randPos];
        $Signals[$i]=$Signal;
        $Signals[$randPos]=$tmpSignal;
        $On{$Signal->[0]}{$Signal->[1]}=0;
        print "False negative : fp=$Signal->[0] pool=$Signal->[1]\n";
        $NumOn--;
    }
    foreach $prb (keys %On)
    {
        foreach $pool (keys %{$On{$prb}})
        {
            if ($On{$prb}{$pool}==1)
            {
                foreach $probeInPool (@{$Pool[$pool]})
                {
                    $Mers{$prb.$probeInPool}=1;
                }
            }
        }
    }
    print STDERR "10mers:", scalar (keys %Mers),"\n";
    print "10mers:", scalar (keys %Mers),"\n";
    $overlap=2;
    foreach $mer (keys %Mers)
    {
        foreach $o (1..$overlap)
        {
            $Prefix[$o]{substr($mer,0,length($mer)-$overlap)}.=
                substr($mer,length($mer)-$overlap,$o)." ";
            $Postfix[$o]{substr($mer,$overlap,length($mer)-$overlap)}.=
                substr($mer,$o-1,$overlap+1-$o)." ";
        }
    }
    undef(%Pre);
    undef(%Post);
    foreach $mer (keys %Mers)
    {
        $Pre{substr($mer,0,length($mer)-1)}.=substr($mer,length($mer)-1,1);
        $Post{substr($mer,1,length($mer)-1)}.=substr($mer,0,1);
    }
    undef(%Mers);
    foreach $submer (keys %Post)
    {
        @chars=split(//,$Pre{$submer});
        @Chars=split(//,$Post{$submer});
        foreach $ch (@chars)
```

```
            {
                    foreach $Ch (@Chars)
                    {
                            $Mers{$Ch.$submer.$ch}=1;
                    }
            }
    }
    foreach $o (1..$overlap)
    {
            foreach $submer (keys %{$Postfix[$o]})
            {
                    @chars=split(/ /,$Prefix[$o]{$submer});
                    @Chars=split(/ /,$Postfix[$o]{$submer});
                    foreach $ch (@chars)
                    {
                            foreach $Ch (@Chars)
                            {
                                    $Mers{$Ch.$submer.$ch}=1;
                            }
                    }
            }
    }
    foreach $i (0..length($Seq)-11)
    {
            $mer = substr($Seq,$i,11);
            if (!$Mers{$mer})
            {
                    print STDERR $mer, " not found!\n";
                    exit(1);
            }
    }

    print STDERR "11mers:", scalar (keys %Mers),"\n";
    print "11mers:", scalar (keys %Mers),"\n";
    foreach $lenMer (12..length($Seq))
    {
            undef(%Prefix);
            undef(%Postfix);
            foreach $mer (keys %Mers)
            {
                    $Prefix{substr($mer,0,length($mer)-1)}.=substr($mer,length($mer)-1,1);
                    $Postfix{substr($mer,1,length($mer)-1)}.=substr($mer,0,1);
            }
```

```
        undef(%Mers);
        foreach $submer (keys %Postfix)
        {
                @chars=split(//,$Prefix{$submer});
                @Chars=split(//,$Postfix{$submer});
                foreach $ch (@chars)
                {
                        foreach $Ch (@Chars)
                        {
                                $Mers{$Ch.$submer.$ch}=1;
                        }
                }
        }
        print STDERR $lenMer,"mers:", scalar (keys %Mers),"\n";
        print $lenMer,"mers:", scalar (keys %Mers),"\n";
        if (($lenMer%50 == 0) && (scalar (keys %Mers) > 4000))
        {
                print STDERR "Cleaning...";
                $Cleaned=0;
                foreach $seq (keys %Mers)
                {
                        undef(%testOn);
                        foreach $i(0..length($seq)-10)
                        {
                                $fprobe=substr($seq,$i,5);
                                $pool=$PoolInd{substr($seq,$i+5,5)};
                                $testOn{$fprobe}{$pool}=1; #To see if all are fully represented
                        }
                        $NumtestOn=0;
                        foreach $prb (keys %testOn) { $NumtestOn += scalar (keys
%{$testOn{$prb}}); }
                        if ($NumtestOn<($lenMer-15))
                        {
                                $Cleaned++;
                                delete $Mers{$seq};
                        }
                }
                print STDERR "$Cleaned cleaned out.\n";
        }
}
print STDERR "Checking all ",scalar (keys %Mers), " solutions for full dot-representation...";
print OUT "#Growths: ", scalar (keys %Mers)," ";
```

```
NEXT:foreach $seq (keys %Mers)
{
        undef(%testOn);
        foreach $i(0..length($seq)-10)
        {
                $fprobe=substr($seq,$i,5);
                $pool=$PoolInd{substr($seq,$i+5,5)};
                $testOn{$fprobe}{$pool}=1; #To see if all are fully represented
        }
        $NumtestOn=0;
        foreach $prb (keys %testOn) { $NumtestOn += scalar (keys %{$testOn{$prb}}); }
        if ($seq eq $Seq)
        {
                $Found=1;
                $seq .= " True solution ";
        }
        if ($NumtestOn>=$NumOn)
        {
                push @Solutions, $seq;
                print "$seq DotsOn=$NumtestOn\n\n";
        }
}
print STDERR "done.\n",scalar @Solutions, " consistent solutions found";
if ($Found)
{
        print STDERR " including the true one.";
}
else {
        print STDERR " - TRUE not FOUND!!";
}
print "Solutions: ",scalar @Solutions," ";

sub new_signal
{
        my ($fp,$pool)=@_;
        my @Signal = ($fp,$pool);
        return \@Signal;
}
```

BuildMMult

```
#!/usr/leo/bin/perl

if (scalar @ARGV <4) { die "Need Pool, Seq, #False positives, #False negatives\n"; }

$FalsePos=$ARGV[2];
$FalseNeg=$ARGV[3];
open(POOL,$ARGV[0]);
print "Using pool $ARGV[0]\n";
$pools=0;
while(<POOL>)
{
        last if (/TotCost/);
        chop $_;
        @Probes=split(/[: ]/,$_);
        shift @Probes;
        shift @Probes;
        shift @Probes;
        if (scalar @Probes > 0)
        {
                @{$Pool[$pools]}=@Probes;
                foreach $probe (@Probes)
                {
                        $PoolInd{$probe}=$pools;
                }
                $pools++;
        }
}

print "Using sequence $ARGV[1]\n";
open(SEQ,$ARGV[1]);
$Seq="";
while (<SEQ>)
{
        chop $_;
        $Seq .= uc($_);
}

$Found=0;
undef(%Mers);
undef(@Solutions);
undef(%On);
```

```
foreach $i(0..length($Seq)-10)
{
        $fprobe=substr($Seq,$i,5);
        $lprobe=substr($Seq,$i+5,5);
        $pool=$PoolInd{$lprobe};
        $On{$fprobe}{$pool}=1;
}
foreach $prb (keys %On)
{
        foreach $pool (keys %{$On{$prb}})
        {
                print "True Signal: fp=$prb pool=$pool\n";
                push @Signals, new_signal($prb,$pool);
        }
}
$NumOn=scalar @Signals;

@char = qw( A C G T );
foreach $1(@char) {
foreach $2(@char) {
foreach $3(@char) {
foreach $4(@char) {
foreach $5(@char) {
        push @Probes, $1.$2.$3.$4.$5;
}}}}}
foreach $i (1..$FalsePos)
{
        $pool = int(rand($pools));
        $fixed = $Probes[rand(1024)];
        $On{$fixed}{$pool}=1;
        print "False positive Signal: fp=$fixed pool=$pool\n";
}

foreach $i (0..$FalseNeg-1)
{
        $tmpSignal=$Signals[$i];
        $randPos = $i + int($NumOn);
        $Signal=$Signals[$randPos];
        $Signals[$i]=$Signal;
        $Signals[$randPos]=$tmpSignal;
        $On{$Signal->[0]}{$Signal->[1]}=0;
        print "False negative : fp=$Signal->[0] pool=$Signal->[1]\n";
        $NumOn--;
```

```
}
foreach $prb (keys %On)
{
        foreach $pool (keys %{$On{$prb}})
        {
                if ($On{$prb}{$pool}==1)
                {
                        foreach $probeInPool (@{$Pool[$pool]})
                        {
                                $Mers{$prb.$probeInPool}=1;
                        }
                }
        }
}
print STDERR "10mers:", scalar (keys %Mers),"\n";
print "10mers:", scalar (keys %Mers),"\n";
#$overlap=2;
#foreach $mer (keys %Mers)
#{
#       foreach $o (1..$overlap)
#       {
#               $Prefix[$o]{substr($mer,0,length($mer)-$overlap)}.=
#                       substr($mer,length($mer)-$overlap,$o)." ";
#               $Postfix[$o]{substr($mer,$overlap,length($mer)-$overlap)}.=
#                       substr($mer,$o-1,$overlap+1-$o)." ";
#       }
#}
undef(%Pre);
undef(%Post);
foreach $mer (keys %Mers)
{
        $Pre{substr($mer,0,length($mer)-1)}.=substr($mer,length($mer)-1,1);
        $Post{substr($mer,1,length($mer)-1)}.=substr($mer,0,1);
}
undef(%Mers);
foreach $submer (keys %Post)
{
        @chars=split(//,$Pre{$submer});
        @Chars=split(//,$Post{$submer});
        foreach $ch (@chars)
        {
                foreach $Ch (@Chars)
                {
                        $Mers{$Ch.$submer.$ch}=1;
                }
        }
}
#foreach $o (1..$overlap)
#{
#       foreach $submer (keys %{$Postfix[$o]})
#       {
#               @chars=split(/ /,$Prefix[$o]{$submer});
#               @Chars=split(/ /,$Postfix[$o]{$submer});
#               foreach $ch (@chars)
#               {
#                       foreach $Ch (@Chars)
#                       {
#                               $Mers{$Ch.$submer.$ch}=1;
#                       }
#               }
#       }
#}
foreach $i (0..length($Seq)-11)
{
        $mer = substr($Seq,$i,11);
        if (!$Mers{$mer})
        {
                print STDERR $mer, " not found!\n";
                exit(1);
        }
}

print STDERR "11mers:", scalar (keys %Mers),"\n";
print "11mers:", scalar (keys %Mers),"\n";
foreach $lenMer (12..length($Seq))
{
        undef(%Prefix);
        undef(%Postfix);
        foreach $mer (keys %Mers)
        {
                $Prefix{substr($mer,0,length($mer)-1)}.=substr($mer,length($mer)-1,1);
                $Postfix{substr($mer,1,length($mer)-1)}.=substr($mer,0,1);
        }
        undef(%Mers);
        foreach $submer (keys %Postfix)
        {
```

```
                    @chars=split(//,$Prefix{$submer});
                    @Chars=split(//,$Postfix{$submer});
                    foreach $ch (@chars)
                    {
                            foreach $Ch (@Chars)
                            {
                                    $Mers{$Ch.$submer.$ch}=1;
                            }
                    }
            }
            print STDERR $lenMer,"mers:", scalar (keys %Mers),"\n";
            print $lenMer,"mers:", scalar (keys %Mers),"\n";
            if (($lenMer%50 == 0) && (scalar (keys %Mers) > 4000))
            {
                    print STDERR "Cleaning...";
                    $Cleaned=0;
                    foreach $seq (keys %Mers)
                    {
                            undef(%testOn);
                            foreach $i(0..length($seq)-10)
                            {
                                    $fprobe=substr($seq,$i,5);
                                    $pool=$PoolInd{substr($seq,$i+5,5)};
                                    $testOn{$fprobe}{$pool}=1; #To see if all are fully represented
                            }
                            $NumtestOn=0;
                            foreach $prb (keys %testOn) { $NumtestOn += scalar (keys
    %{$testOn{$prb}}); }
                            if ($NumtestOn<($lenMer-15))
                            {
                                    $Cleaned++;
                                    delete $Mers{$seq};
                            }
                    }
                    print STDERR "$Cleaned cleaned out.\n";
            }
    }
    print STDERR "Checking all ",scalar (keys %Mers), " solutions for full dot-representation...";
    print OUT "#Growths: ", scalar (keys %Mers)," ";

    NEXT:foreach $seq (keys %Mers)
    {
            undef(%testOn);
```

```
        foreach $i(0..length($seq)-10)
        {
                $fprobe=substr($seq,$i,5);
                $pool=$PoolInd{substr($seq,$i+5,5)};
                $testOn{$fprobe}{$pool}=1; #To see if all are fully represented
        }
        $NumtestOn=0;
        foreach $prb (keys %testOn) { $NumtestOn += scalar (keys %{$testOn{$prb}}); }
        if ($seq eq $Seq)
        {
                $Found=1;
                $seq .= " True solution ";
        }
        if ($NumtestOn>=$NumOn)
        {
                push @Solutions, $seq;
                print "$seq DotsOn=$NumtestOn\n\n";
        }
}
print STDERR "done.\n",scalar @Solutions, " consistent solutions found";
if ($Found)
{
        print STDERR " including the true one.";
}
else {
        print STDERR " - TRUE not FOUND!!";
}
print "Solutions: ",scalar @Solutions," ";

sub new_signal
{
        my ($fp,$pool)=@_;
        my @Signal = ($fp,$pool);
        return \@Signal;
}
```

APPENDIX 2
Experimental Target Sequence r300

```
GTAGGGGTAG ACATCGCGTA AAAGGGGCGT ACCCAGGACC CCCCTTGGCT CAATAAGTAG
CGCTGGGGTG CTACTACGGG TCTCGACACG CATTCAACTA AAAGCTTCCA TTCGCACGGG
CTTATTTAAC GAAGGTCGCG ATAAGGTGCC GAATAGGCTG CAGAGCGGCA GCCTGTCCAG
TGAATGCTGT GAGGCCTCCA GCTGACTCAT GAGAGAAGCC CAGTATTCAA ACTACGATTC
CACTCGACAA TTTAGGATGT CTTCCCGAAA GCTATCGGGT AGAATATCAG ATTCGTTTAA
```

APPENDIX 3
D16 and DN16 Pools of Probes

D16

Group 0:64:

GATTT	CAGCT	GAAAA	TGGTT	AAAGT	CGCTC
AAGAT	CAAGC	TAACG	GCCTC	TGCAA	CAATG
AGAAC	TCAAA	ACTAT	TCAGT	GGGAA	TTCTA
TTGCT	GTAAG	GGTAC	TTAGA	TAGTC	CCACA
CTCTT	ATGAA	TCTGA	ACCGC	TACAC	CCTTA
AACAG	TGGGG	GCACC	GTGGC	GGCTG	GTCCA
TATGT	GGACT	AGCGA	TGATG	GATCA	TCCCG
TCTCC	GCGGG	GTCGT	CGCCG	ATTGG	GTATC
AGTTA	ATACT	CTTCC	CCCAG	GCCAT	TGTTC
GTTTG	CGTAG	TTTAT	AACCC	CCGAC	CAGGA
CTGCG	CGATA	ACGTG	AGGCA		

Group 1:64:

GTAAA	TCAGG	ACTCC	ATTAC	CCTGT	GCCCG
GGATG	CAACG	AATGG	TATCG	CTCAA	TGCCG
GAGGA	TAATC	CAAGT	TGCTA	ACCAA	TAGAA
GGCTC	TACGC	CGGGG	TTATA	CTCGG	CTACC
ATCTG	TTTAG	CATAC	CCCCT	GACAG	AGAGA
CCAGA	ACTTA	GCACA	GCTTG	TCCAC	CTGGC
CCGCA	AAACA	ATGTC	TGGGT	TCTTC	ATAGT
TGACC	TTGAT	AGCAT	GTTCA	CGTCA	ACATT
AGGTA	AAGCC	CGAAT	CGGAC	TCGTG	GGCGT
TGTGA	GGTGC	CAGTT	GCGGC	CACTA	GGTCT
GTATT	ATCCT	GCGAT	CTTTT		

Group 2:64:

TAGGG	GCGTC	GTTTC	AGATT	TGTGT	TTCGA
TAATT	CTGAC	GGGCG	CCAAT	CTAGT	ATTCC
TGCTG	GTCCG	TACCC	AGCAG	GTTCT	ACTCG
TTAAG	CCCGG	CGCCT	GATAG	TACAA	TCATA
CAGAG	TCCCT	CCTAA	GGAGA	CCACC	AGACC
CCGTT	TCTGC	CGATG	AGCGC	CGGTA	ACAGG
ATGGG	AAGTG	CATGT	GCATT	CAGTC	CTTTG
TTGTT	TGAAC	GAAGG	GGCAA	GACTA	ACGAC
ACCAT	CAACA	AATAC	GACGT	ATAAA	GTAGC
GAGGC	AAACT	CTCTC	GCTCA	CGTGC	ATCTA
GGGAT	TTGCA	ACGGA	AGTAA		

Group 3:64:

ACGGT	CGGCA	ATACG	CCTTC	AACGC	CGCGC
CGACG	CGGGT	GTGAA	AGCAA	CTAGA	TCGTA
GAACT	TGTGC	GCCCC	TTCTT	TGCCT	TCAAG
CCTGG	TAAGG	TCCTC	ATCAC	CACTT	ACGAG
GTTGT	TTAGC	CACAG	GCATA	AAATC	CTAAT
GCAGC	GAGAC	GGTAG	TGACA	AATAT	TATTC
TTGCG	GCTAT	TGGAT	GATGG	ATGTT	TACCA
ATTTG	TCTCG	CTTTA	ATTGA	CTCCC	AGGCC
GTCGG	GGTTA	AAGCG	ACTCA	TCCGT	AGCTG
CATCC	AGAGG	GAGTG	GTGTC	AAAAA	CCCAA
CGATT	CCGCT	TAGGA	TGGTG		

Group 4:64:

TAAAT	CGTAT	AAAAG	CAAGA	ACGAT	GAACA
TTCGC	AACGG	TATGA	ATCAA	TCCAA	CGCGT
CAGAA	AACCT	GCGCA	GAGGG	AATGT	ATGCC
CTGGT	GGGCT	GAGTT	AAGTA	CAGCC	TGTAC
CAATC	AGCTC	GCAAC	ATGGA	TAGCA	TCTCT
GCCGT	CTACG	CTTAC	GTTAG	CCTGC	CGTCG
TTCCG	TCGGG	GACAC	ACATG	GGCGA	AGACT
GGATA	GTCCT	ATAAC	CCATT	CACTG	GATTC
TACTA	CCCCA	ATTTC	GGGAG	CCGAG	CATCT
TGAGC	GGTCA	GTAGG	AGTGG	TCACC	CGCCC
GCTTT	TTATT	TGTTG	TTGTG		

Group 5:64:

GGGCC	TCTAG	ACCGA	GAAAT	CATAG	CCTGA
GTTAC	AACAA	TGCGC	CGGAA	AACTT	TAAAA
ATAAG	CGCTT	GCTCT	ACTGT	TCCAT	ATGAT
CTTCG	CTGGG	AATCG	CAGCA	TTCTG	GCTAA
TGGTA	CCCCG	CTATA	AGCGG	GAACG	CACGG
TGTAT	GCCGC	TACCT	TCGCC	TAGAC	GTGTT
AGATA	GACCC	TAGGT	AAGGC	ACACC	TTATC
TCATG	CCAAC	CTCAC	GCGTG	GTACA	GACGA
CCGTC	ATTTT	GATTA	CATTC	CTGCT	CGAGC
TGACT	AGTTC	GGAGT	CCAGT	AGGAC	GTCAG
GGTGG	ACGCG	ATCCC	TTTGA		

Group 6:64:

AAGGG	CATAT	GCCTA	GAAAC	TGATT	ACAGC
TCGTC	CAGAC	CCAAG	CGCAA	CAATA	CTCTG
CGGCG	AGGAG	AACGA	GGCCC	CTACT	ACGCC
CTCGC	GCGCG	TACCG	AGCAC	TGTAA	TGCGT
TTGAA	ATAGA	TTAGG	TAAAG	ACTAA	TTTAC
ATTGT	TATGC	AACTC	ACCAG	TAGCT	AGATG
TATTA	CATGG	CGTGT	GTCAT	AGTTT	AAAAT
CCGGT	TCGAT	ATTCG	GGACA	GGGTA	GAGGT
CGAAC	GCATC	GCTGA	GTGAG	CCTCA	TCACT
GACTT	GATCC	GTTGG	TTCCA	CGGGA	ACCCT
CCTTG	CTTTC	CTGTT	TGGGC		

Group 7:64:

TCCGG	CGGGC	AAAAC	GGAAG	GACAA	TGCAC
GTCTC	CAATT	AGTGT	GCAGT	GGCCG	GCGAA
CGTTC	TGGCA	AGTAG	TTAAT	CGACT	CTTCA
AACTG	ACTTT	ACGGC	CTAAC	CGTAA	GACCT
ACCCC	ATAGG	CTTAG	TACAG	TCGGT	CACGC
GATAC	CTCGT	CATTA	TGAGA	AGACG	GTGGG
CCGTG	GTATA	GATTG	GGTGA	TAGTT	GGGTC
ATGTA	TCTAC	CAGCG	TCACA	TAAGC	AAGGA
ATCAT	TTGCC	ATTCT	GTACC	TCCTA	TGTTT
CGGAT	TTATG	TATCT	AGCCA	ACATA	TCGAG
GCGCT	TTGGA	CCAGG	CCTCC		

Group 8:64:

TTTTC	GCCCA	ATATC	GGAAA	GTTGA	CAGTA
AGTCC	TGTCG	TCACG	ACCAC	CGATC	TGAAT
AAAGA	CGTTA	CACAT	CCTCT	GATGC	CTAAG
CTACA	TATAA	GCGAC	GGGGG	CTCGA	GGCGC
GTCTT	ATTAT	TAATG	AGGAT	AATCA	TCTGT
CACCC	AACCG	AATAG	CGGCT	GAGAG	GTGTA
CCGAA	GTTCG	ACGCT	CCGCG	GGTTT	AAATT
AGGTG	TCCTT	ATCGT	TTCAA	ACTTG	ATGGC
TCGGA	CCCTA	TAGCC	TGCCA	TTGGG	TATTT
GCATG	AGAAG	CGCAG	CAAGG	TGGAC	GAGCT
CCAGC	ACAAA	GCCGG	GTACT		

Group 9:64:

TAGTA	TATCC	GTAAC	ACGTT	TTTAA	CTTGT
ATACA	CACAA	AGGAA	ATAGC	TTACT	CACCT
CATCG	TTCGT	GCTTA	AACCA	CGCTG	TAGAG
CCCTT	GCCGA	AATTT	AGTCT	GCGCC	TCGCA
ATAAT	TAAAC	CTGTA	CGACC	TACGG	GGCGG
CCGGC	GATAT	CAGGT	CCAAA	TGGTC	TTCCC
ATCTC	AGGGC	GAATG	TGTCA	GTGAT	GGCCT
TTGGC	ATGCG	ACTGA	TCATT	GAAGT	GGGTT
GCCAC	ACTAC	TCTGG	ACCCG	CGTAC	CTCAG
AAAGG	CATGA	TTTTG	GACTC	GTTGC	TGCAT
GCTAG	GGACG	TGAGG	GTGGA		

Group 10:64:

CCACG	GCAGA	AACGT	GAGAT	GGGCA	CGTTG
CCGTA	TGATA	GCCTG	TCCGA	AAGCA	GAATA
ATTCA	CGAGG	TGGGA	GGCTT	TGAAG	TGTCC
CCCTC	GCTGT	TAACT	ATCAG	TTCCT	CTCTA
TTACC	GGAAC	TCGCG	GTTAA	ATGAC	GATCG
GCAAT	GTGCT	ACTAG	AGGGT	AGCCG	CGCCA
TTTGG	ATGTG	CTGCC	TCTTT	CATGC	AAATG
TCTCA	ACACA	AGTGA	AGTAT	GTCAC	ACATC
GACGC	GGTTC	ACCTT	TACTG	CTGGA	GACCA
AGAGC	ACGGG	CAAAA	GTACG	CGCAT	TCGAC
TATAT	CTATT	GAGTC	CCTAC		

Group 11:64:

GAACC	AGCCT	AACAT	CACGT	GGGGC	CTCCA
GCTCG	ACAAG	GTCTG	GGCTA	TGTAG	ATATT
TCGTT	CCACT	ACCTA	CTTAT	CGCAC	TAATA
TGGCG	TTCGG	TACTC	TTGTC	AGGTC	TCCCA
CCTTT	GTTCC	ATTTA	GCGGA	GGAAT	TTTCT
CAAAC	ATCGC	GAGCG	CTTGA	TCAAC	TTCAT
TGATC	AGAGT	ACCGG	CGAGA	GCTTC	CAGTG
TTAAA	TACGA	CCGCC	AAAGC	AGACA	GGTAA
CGTGG	GCCAG	CGGTT	GATGT	GCAGG	AATTG
TAGAT	AAGAA	CATCA	ATGCA	CTGAG	ATGGT
GTGAC	ACTGC	AGTAC	CTATG		

Group 12:64:

TGCTT	GAGCA	ATATG	TTACA	GGATC	ACACG
CTCAT	CTGTC	AGAAT	TATTG	CTGCA	GCAAA
ATTAA	TACGT	GCCCT	AAGAC	ACCCA	GTCTA
CTAAA	AGTCG	ACCTC	CGCGA	GGTAT	CGAGT
CCTAG	GTCCC	TTCAC	GTGCG	TGGCC	TCGCT
TTGAG	TCAAT	GATGA	AGCTA	GGCAC	CTTCT
GCGGT	ACGAA	ATCGG	CCGGG	TGGAA	TGTGG
CGTTT	AGGCT	GAAAG	GAAGC	AAATA	GGGTG
CACCG	TCAGC	CCATG	GCTCC	CTTGC	CACTC
AATCC	TCCAG	AAGTT	CATAA	CAACC	TCTTA
TGACG	CGGAG	GTAGT	ACAGA		

Group 13:64:

TTTGT	ATTAG	TAAGA	TCGAA	CGACA	ACTTC
AGAAA	GTAGA	AAGTC	CCTAT	GCGTA	CGTCC
ACCGT	TCTTG	GAATT	TCCCC	ATCCG	GCTGC
GTCAA	GATAA	GGTCG	TTCTC	TGGCT	AGGGG
GGGAC	CCATC	GTGGT	GTTTT	AACTA	TCGGC
AAACC	GGCCA	TGAGT	AATGA	CTCCT	GTGCC
CAGGC	TATAC	GACGG	AGGTT	AGCCC	TACAT
CAGAT	GCACG	GTGTG	GGCAT	CGCGG	TTTCA
CCCGA	AATCT	TGCAG	CTGAA	CGGTG	ACACT
CCCAC	TAGCG	CTAGG	CAAAG	TTAAC	ATTGC
GGAGC	ACCTG	TGTTA	ACGCA		

Group 14:64:

CGGTC	GCTGG	GTCGC	TTTCC	TTGTA	CACAC
GCGTT	ACAAC	CGTCT	ACTCT	CGAAA	AGTTG
CTTGG	AGCTT	ACGTA	AGTGC	TGGAG	AGTCA
CTCCG	TTAGT	GTAAT	TTACG	GGGGA	ACAGT
TCATC	ATCGA	CCCAT	CCCGC	GCAAG	TGCCC
TTCAG	GAAGA	AAACG	TAACA	CAAAT	ATGAG
AATAA	ATATA	TGCGA	GGCAG	GCTAC	CTATC
CCGGA	CACCA	GAGAA	TTTTT	CAGGG	GATCT
TACTT	GGTGT	CATTG	GGACC	GACTG	ATGCT
GAGCC	TATGG	TCCTG	TAGGC	AAGGT	AATTC
GTTTA	CTGAT	GGATT	TCTAA		

Group 15:64:

AATTA	TAGTG	TATAG	GGGGT	GGTCC	TGAAA
CTTAA	AAGCT	CCCTG	CTGTG	GCCTT	CGAAG
CCTCG	TATCA	TAACC	TTGGT	CATTT	CCATA
TAAGT	CGTGA	AGGGA	GTCGA	GGTTG	AGATC
TGTCT	ATCTT	GACAT	TCAGA	GGAGG	AAGAG
AGGCG	GTTAT	TGCGG	CCCGT	TTTCG	CACGA
GAATC	ATACC	CAACT	GCACT	TTGAC	ACTGG
GCCAA	CCGAT	TGCTC	GTGCA	GCGAG	GACCG
GAGTA	TTTTA	AGCGT	CGCTA	TCCGC	TCTAT
CGGCC	CTAGC	GTATG	ATCCA	AACAC	ACAAT
TTTGC	CCCCC	ACGTC	AATGC		

DN16

Group 0:64:

GATTT	CCTTT	GAAAA	TGGTT	AAAGT	CGCTC
AAGAT	CAAGC	TAACG	GCCTC	TGCAA	CAATG
AGAAC	TCAAA	ACTAT	TCAGT	GGGAA	TTCTA
TTGCT	GTAAG	GGTAC	TTAGA	TAGTC	CCACA
CTCTT	AACTT	TCTGA	ACCGC	TACAC	GACGA
AACAG	TGGGG	GCACC	GTGGC	GGCTG	GTCCA
TATGT	GGACT	AGCGA	TGATG	GATCA	TCCCG
TCTCC	GCGGG	GTCGT	CGCCG	ATTGG	GTATC
AGTTA	ATACT	CTTCC	CCCAG	CTTAT	TTGAG
GTTTG	CGTAG	CATGG	AACCC	CCGAC	CAGGA
CTGCG	CGATA	ACGTG	AGGCA		

Group 1:64:

GTAAA	TCAGG	ACTCC	AAAGC	CCTGT	GCCCG
GGATG	CAACG	AATGG	TATCG	CTCAA	TGCCG
GAGGA	TAATC	CAAGT	TGCTA	ACCAA	TAGAA
GGCTC	TACGC	CGGGG	TTATA	CTCGG	CTACC
ATCTG	TTTAG	CATAC	CCCCT	GACAG	AGAGA
CCAGA	ACTTA	GCACA	GCTTG	TCCAC	CTGGC
CCGCA	AAACA	ATGTC	TGGGT	TCTTC	ACCTT
TGACC	TTGAT	AGCAT	GTTCA	CGTCA	ACGAG
AGGTA	ATTGT	CGAAT	CGGAC	TCGTG	GGCGT
TGTGA	GGTGC	CAGTT	GCGGC	CACTA	GGTCT
GTATT	ATCCT	GCGAT	CTTTT		

Group 2:64:

TAGGG	GCGTC	GTTTC	TTGGC	CCCCA	TTCGA
TAATT	AGGTG	GGGCG	CCAAT	CTAGT	ATTCC
TGCTG	GTCCG	TACCC	AGCAG	GTTCT	ACTCG
TTAAG	CCCGG	CGCCT	GATAG	TACAA	TCATA
CAGAG	TCCCT	CCTAA	GGAGA	CCACC	AGACC
CCGTT	TCTGC	CGATG	AGCGC	CGGTA	ACAGG
TATCA	ATGAT	CATGT	GCATT	CAGTC	CTTTG
TTGTT	TGAAC	TTTAC	GGCAA	GACTA	ACGAC
ACCAT	CAACA	AATAC	GACGT	ATAAA	GTAGC
GAGGC	AAACT	CTCTC	GCTCA	CGTGC	ATCTA
GGGAT	TTGCA	ACGGA	AGTAA		

Group 3:64:

ACGGT	CGGCA	ATACG	CCTTC	AACGC	CGCGC
CGACG	CGGGT	GTGAA	AGCAA	CTAGA	TCGTA
GAACT	TGTGC	GCCCC	TTCTT	TGCCT	TCAAG
CCTGG	TAAGG	TCCTC	ATCAC	CACTT	ACATT
GTTGT	TTAGC	CACAG	GCATA	AAATC	CTAAT
GCAGC	GAGAC	GGTAG	TGACA	AATAT	TATTC
TTGCG	GCTAT	TGGAT	GATGG	ATGTT	TACCA
ATTTG	TCTCG	CTTTA	ATTGA	CTCCC	AGGCC
GTCGG	GGTTA	AAGCG	GGGCT	TCCGT	AGCTG
CATCC	AGAGG	GAGTG	GTGTC	AAAAA	CCCAA
CGATT	AGTAC	TAGGA	TGGTG		

Group 4:64:

TAAAT	CGTAT	AAAAG	CAAGA	ACGAT	GAACA
TTCGC	AACGG	TATGA	ATCAA	TCCAA	CGCGT
CAGAA	AACCT	GCGCA	GAGGG	AATGT	ATGCC
CTGGT	ACTCA	GAGTT	AAGTA	CAGCC	TGTAC
CAATC	AGCTC	GCAAC	ATGGA	TAGCA	TCTCT
GCCGT	CTACG	CTTAC	GTTAG	CCTGC	CGTCG
TTCCG	TCGGG	GACAC	ACATG	GGCGA	AGACT
GGATA	GTCCT	ATAAC	CCATT	CACTG	GATTC
CGCTA	CTTGA	ATTTC	GGGAG	CCGAG	CATCT
TGAGC	GGTCA	GTAGG	AGTGG	TCACC	CGCCC
GCTTT	TTATT	TGTTG	TTGTG		

Group 5:64:

GGGCC	TCTAG	ACCGA	GAAAT	CATAG	CCTCC
GTTAC	AACAA	TGCGC	CGGAA	ATGAA	TAAAA
ATAAG	CGCTT	ACGCA	ACTGT	TCCAT	AAGTG
CTTCG	CTGGG	AATCG	CAGCA	TTCTG	GCTAA
TGGTA	CCCCG	CTATA	AGCGG	GAACG	CACGG
TGTAT	GCCGC	TACCT	TACTA	TAGAC	GTGTT
AGATA	GACCC	TAGGT	AAGGC	ACACC	TTATC
TCATG	CCAAC	GTCGA	GCGTG	GTACA	CCTTA
CCGTC	ATTTT	GATTA	CATTC	CTGCT	CGAGC
TGACT	AGTTC	GGAGT	CCAGT	AGGAC	GTCAG
GGTGG	GATCT	ATCCC	TTTGA		

Group 6:64:

AGGCT	CATAT	GCCTA	GAAAC	TGATT	ACAGC
TCGTC	CAGAC	CCAAG	CGCAA	CAATA	CTCTG
CGGCG	GCCAG	AACGA	GGCCC	CTACT	ACTTC
CTCGC	GCGCG	TACCG	AGCAC	TGTAA	TGCGT
TTGAA	ATAGA	TTAGG	TAAAG	ACTAA	GAAGG
AAGCC	TATGC	TGTCG	CCCCC	TAGCT	AGATG
TATTA	TTTAT	CGTGT	GTCAT	AGTTT	AAAAT
CCGGT	TCGAT	ATTCG	GGACA	GGGTA	GAGGT
CGAAC	GCATC	GCTGA	GTGAG	CCTCA	TCACT
AGGAG	GATCC	GTTGG	TTCCA	CGGGA	ACCCT
CCTTG	CTTTC	CTGTT	TGGGC		

Group 7:64:

TCCGG	CGGGC	AAAAC	GGAAG	GACAA	TGCAC
GTCTC	CAATT	AGTGT	GCAGT	GGCCG	GCGAA
CGTTC	TGGCA	AGTAG	TTAAT	CGACT	CTTCA
AACTG	ACTTT	ACGGC	CTAAC	CGTAA	GACCT
ACCCC	ATAGG	CTTAG	TACAG	TCGGT	CACGC
GATAC	CTCGT	CATTA	TGAGA	AGACG	GTGGG
CCGTG	GTATA	GATTG	GGTGA	TAGTT	GGGTC
ATGTA	TCTAC	CAGCG	TCACA	TAAGC	AAGGA
ATCAT	TTGCC	ATTCT	GTACC	TCCTA	TGTTT
CGGAT	TTATG	TATCT	CCTGA	ACATA	TCGAG
GCGCT	TTGGA	CCAGG	AGCCA		

Group 8:64:

TTTTC	GCCCA	ATATC	GGAAA	GTTGA	CAGTA
AGTCC	CTGAC	TCACG	ACCAC	CGATC	TGAAT
AAAGA	CGTTA	CACAT	CCTCT	GATGC	CTAAG
CTACA	TATAA	GCGAC	GGGGG	CTCGA	GGCGC
GTCTT	ATTAT	TAATG	AGGAT	AATCA	TCTGT
CACCC	AACCG	AATAG	CGGCT	GAGAG	GTGTA
CCGAA	GTTCG	ACGCT	CCGCG	GGTTT	AAATT
AACTC	TCCTT	ATCGT	TTCAA	ACTTG	ATGGC
TCGGA	CCCTA	TAGCC	TGCCA	TTGGG	TATTT
GCATG	AGAAG	CGCAG	CAAGG	TGGAC	GAGCT
CCAGC	ACAAA	GCCGG	GTACT		

Group 9:64:

TAGTA	TATCC	GTAAC	ACGTT	TTTAA	CTTGT
ATACA	CACAA	AGGAA	ATAGC	TTACT	CACCT
CATCG	TTCGT	GCTTA	AACCA	CGCTG	TAGAG
CCCTT	GCCGA	AATTT	AGTCT	GCGCC	TCGCA
ATAAT	TAAAC	CTGTA	CGACC	TACGG	GGCGG
CCGGC	GATAT	CAGGT	CCAAA	TGGTC	TTCCC
ATCTC	AGGGC	GAATG	TGTCA	GTGAT	GGCCT
AGATT	ATGCG	ACTGA	TCATT	GAAGT	GGGTT
GCCAC	ACTAC	TCTGG	ACCCG	CGTAC	CTCAG
AAAGG	CATGA	TTTTG	GACTC	GTTGC	TGCAT
GCTAG	GGACG	TGAGG	GTGGA		

Group 10:64:

CCACG	GCAGA	AACGT	GAGAT	GGGCA	CGTTG
CCGTA	TGATA	GCCTG	TCCGA	AAGCA	GAATA
ATTCA	CGAGG	TGGGA	GGCTT	TGAAG	TGTCC
CCGCT	GCTGT	TAACT	ATCAG	TTCCT	CTCTA
TTACC	GGAAC	TCGCG	GTTAA	ATGAC	GATCG
GCAAT	GTGCT	ACTAG	AGGGT	AGCCG	CGCCA
TTTGG	ATGTG	CTGCC	TCTTT	CATGC	AAATG
TCTCA	ACACA	AGTGA	AGTAT	GTCAC	ACATC
GACGC	GGTTC	ATAGT	TACTG	CTGGA	GACCA
AGAGC	ACGGG	CAAAA	GTACG	CGCAT	TCGAC
TATAT	CTATT	GAGTC	CCTAC		

Group 11:64:

GAACC	AGCCT	AACAT	CACGT	GGGGC	CTCCA
GCTCG	ACAAG	GTCTG	GGCTA	TGTAG	ATATT
TCGTT	CCACT	GCCAT	ACCTA	CGCAC	TAATA
TGGCG	TTCGG	TACTC	TTGTC	AGGTC	TCCCA
CAGCT	GTTCC	ATTTA	GCGGA	GGAAT	TTTCT
CAAAC	ATCGC	GAGCG	TGTGT	TCAAC	TTCAT
TGATC	AGAGT	ACCGG	CGAGA	GCTTC	CAGTG
TTAAA	TACGA	CCGCC	ATTAC	AGACA	GGTAA
CGTGG	GACTT	CGGTT	GATGT	GCAGG	AATTG
TAGAT	AAGAA	CATCA	ATGCA	CCCTC	ATGGT
GTGAC	ACTGC	CTGAG	CTATG		

Group 12:64:

TGCTT	GAGCA	ATATG	TTACA	GGATC	ACACG
CTCAT	CTGTC	AGAAT	TATTG	TTTTT	GCAAA
ATTAA	TACGT	GCCCT	AAGAC	ACCCA	GTCTA
CTAAA	AGTCG	ACCTC	CGCGA	GGTAT	CGAGT
CCTAG	GTCCC	TTCAC	GTGCG	TGGCC	TCGCT
TGTTC	TCAAT	GATGA	AGCTA	GGCAC	CTTCT
GCGGT	ACGAA	ATCGG	CCGGG	TGGAA	TGTGG
CGTTT	AAGGG	GAAAG	GAAGC	AAATA	GGGTG
CACCG	TCAGC	CCATG	GCTCC	CTTGC	CACTC
AATCC	TCCAG	AAGTT	CATAA	CAACC	TCTTA
TGACG	CGGAG	GTAGT	ACAGA		

Group 13:64:

TTTGT	ATTAG	TAAGA	TCGAA	CGACA	ACGCC
AGAAA	GTAGA	AAGTC	CCTAT	GCGTA	CGTCC
ACCGT	TCTTG	GAATT	TCCCC	ATCCG	GCTGC
GTCAA	GATAA	GGTCG	TTCTC	TGGCT	AGGGG
GGGAC	CCATC	GTGGT	GTTTT	AACTA	TCGGC
AAACC	GGCCA	TGAGT	AATGA	CTCCT	GTGCC
CAGGC	TATAC	GACGG	AGGTT	AGCCC	TACAT
CAGAT	GCACG	GTGTG	GGCAT	CGCGG	TTTCA
CCCGA	AATCT	TGCAG	CTGAA	CGGTG	ACACT
CCCAC	TAGCG	CTAGG	CAAAG	TTAAC	ATTGC
GGAGC	ACCTG	TGTTA	GCTCT		

Group 14:64:

CGGTC	GCTGG	GTCGC	TTTCC	TTGTA	CACAC
GCGTT	ACAAC	CGTCT	ACTCT	CGAAA	AGTTG
CTTGG	AGCTT	ACGTA	AGTGC	TGGAG	AGTCA
CTCCG	TTAGT	GTAAT	TTACG	ACGCG	ACAGT
TCATC	ATCGA	CCCAT	CCCGC	GCAAG	TGCCC
TTCAG	GAAGA	AAACG	TAACA	CAAAT	ATGAG
AATAA	ATATA	TGCGA	GGCAG	GCTAC	CTATC
CCGGA	CACCA	GAGAA	CTGCA	CAGGG	GGGGA
TACTT	GGTGT	CATTG	GGACC	GACTG	ATGCT
GAGCC	TATGG	TCCTG	TAGGC	AAGGT	AATTC
GTTTA	CTGAT	GGATT	TCTAA		

Group 15:64:

AATTA	TAGTG	TATAG	GGGGT	GGTCC	TGAAA
CTTAA	AAGCT	CCCTG	CTGTG	GCCTT	CGAAG
CCTCG	ATGGG	TAACC	TTGGT	CATTT	CCATA
TAAGT	CGTGA	AGGGA	CTCAC	GGTTG	AGATC
TGTCT	ATCTT	GACAT	TCAGA	GGAGG	AAGAG
AGGCG	GTTAT	TGCGG	CCCGT	TTTCG	CACGA
GAATC	ATACC	CAACT	GCACT	TTGAC	ACTGG
GCCAA	CCGAT	TGCTC	GTGCA	GCGAG	GACCG
GAGTA	TCGCC	AGCGT	TTTTA	TCCGC	TCTAT
CGGCC	CTAGC	GTATG	ATCCA	AACAC	ACAAT
TTTGC	ACCAG	ACGTC	AATGC		

APPENDIX 4
Simulation Results

r300.0.0.out

```
Using pool D16
Using sequence r300

True Signal: fp=CTCGA pool=7
True Signal: fp=CTACG pool=1
True Signal: fp=CTACG pool=2
True Signal: fp=GTACC pool=0
True Signal: fp=ATCGC pool=1
True Signal: fp=GAATG pool=15
True Signal: fp=ATCGG pool=13
True Signal: fp=GTCGC pool=13
True Signal: fp=ACCCA pool=14
True Signal: fp=CTGGG pool=10
True Signal: fp=CAATT pool=3
True Signal: fp=GACAA pool=1
True Signal: fp=TACTA pool=3
True Signal: fp=ACCCC pool=6
True Signal: fp=AGACA pool=10
True Signal: fp=TTCCA pool=8
True Signal: fp=TTCCA pool=4
True Signal: fp=ACGCA pool=8
True Signal: fp=GACAC pool=2
True Signal: fp=CGACA pool=10
True Signal: fp=CGACA pool=11
True Signal: fp=CTACT pool=10
True Signal: fp=CCCCC pool=9
True Signal: fp=CCCCC pool=14
True Signal: fp=TTCCC pool=12
True Signal: fp=GCCCA pool=1
True Signal: fp=GAGAA pool=8
True Signal: fp=CCAGC pool=5
True Signal: fp=CAGAG pool=3
True Signal: fp=GCAGA pool=1
True Signal: fp=GCAGC pool=12
True Signal: fp=CGCGA pool=3
True Signal: fp=AGCGC pool=0
True Signal: fp=GGACC pool=1
True Signal: fp=CCAGG pool=7
True Signal: fp=TTAGG pool=1
True Signal: fp=GAGAG pool=1
True Signal: fp=TAAAA pool=11
True Signal: fp=AGCGG pool=4
True Signal: fp=ACTAA pool=15
True Signal: fp=CGGGC pool=4
True Signal: fp=ACTAC pool=4
True Signal: fp=ACTAC pool=7
True Signal: fp=AGGGG pool=9
True Signal: fp=AGGGG pool=5
True Signal: fp=TTTAA pool=15
True Signal: fp=GGGGC pool=7
True Signal: fp=CAGAT pool=11
True Signal: fp=CATGA pool=14
True Signal: fp=AATGC pool=1
True Signal: fp=CCCCT pool=13
True Signal: fp=GACAT pool=4
True Signal: fp=TCTTC pool=8
True Signal: fp=CCAGT pool=10
True Signal: fp=CCAGT pool=9
True Signal: fp=GCTAC pool=9
True Signal: fp=TTTAG pool=11
True Signal: fp=TGAGA pool=12
True Signal: fp=TGCCG pool=8
True Signal: fp=GCGCT pool=15
True Signal: fp=CGCGT pool=4
True Signal: fp=TGAGG pool=7
True Signal: fp=TCGGG pool=1
True Signal: fp=CGGGT pool=8
True Signal: fp=CGGGT pool=12
True Signal: fp=GGCGT pool=12
True Signal: fp=TATCA pool=4
True Signal: fp=ATATC pool=2
True Signal: fp=CTATC pool=6
True Signal: fp=GGGGT pool=11
True Signal: fp=GGGGT pool=14
True Signal: fp=TATCG pool=3
True Signal: fp=GCTAT pool=3
True Signal: fp=GATGT pool=0
True Signal: fp=TGGCT pool=6
True Signal: fp=CTCAA pool=15
True Signal: fp=ATCAG pool=6
True Signal: fp=CGATA pool=8
True Signal: fp=CTGAC pool=5
True Signal: fp=GTATT pool=11
True Signal: fp=ATGAG pool=8
True Signal: fp=GCCTC pool=0
True Signal: fp=GTGAA pool=2
True Signal: fp=GCGTA pool=0
True Signal: fp=GCGTA pool=9
True Signal: fp=GCCTG pool=12
True Signal: fp=GGATG pool=1
True Signal: fp=GTGAG pool=0
True Signal: fp=TTAAC pool=2
True Signal: fp=AAAGC pool=1
True Signal: fp=AAAGC pool=6
True Signal: fp=AAGCC pool=8
True Signal: fp=CTCAT pool=8
True Signal: fp=AGATT pool=12
True Signal: fp=CAGCC pool=10
True Signal: fp=CGCAC pool=4
True Signal: fp=AAAGG pool=1
True Signal: fp=GACCC pool=9
True Signal: fp=CCCTT pool=1
True Signal: fp=CGATT pool=11
True Signal: fp=GAAGC pool=5
True Signal: fp=TCATG pool=1
True Signal: fp=AGGAC pool=15
True Signal: fp=TGCTA pool=4
True Signal: fp=GAAGG pool=10
True Signal: fp=AATAA pool=2
True Signal: fp=TGCTG pool=9
True Signal: fp=GGCAG pool=1
True Signal: fp=GAGCG pool=3
True Signal: fp=CTTGG pool=1
True Signal: fp=ACAAT pool=6
True Signal: fp=ACTCA pool=7
True Signal: fp=TCCAC pool=10
True Signal: fp=AATAG pool=13
True Signal: fp=GATAA pool=1
True Signal: fp=TACGA pool=6
True Signal: fp=TATTC pool=2
True Signal: fp=CCTCC pool=3
True Signal: fp=TAACG pool=14
True Signal: fp=AAGCT pool=12
True Signal: fp=AAGCT pool=5
True Signal: fp=ACTCG pool=15
True Signal: fp=CAGCT pool=9
True Signal: fp=TCCAG pool=8
True Signal: fp=TCCAG pool=2
True Signal: fp=CGCAT pool=11
True Signal: fp=TCGAC pool=9
True Signal: fp=TCGAC pool=13
True Signal: fp=GCTCA pool=5
True Signal: fp=AGGAT pool=8
True Signal: fp=TAGGA pool=15
True Signal: fp=AGTGA pool=14
True Signal: fp=TAGGC pool=13
True Signal: fp=TACGG pool=7
True Signal: fp=TAGGG pool=13
True Signal: fp=AATAT pool=13
True Signal: fp=GGTGC pool=1
True Signal: fp=GGTGC pool=4
True Signal: fp=TCCAT pool=9
True Signal: fp=TGAAT pool=10
True Signal: fp=TATTT pool=6
True Signal: fp=TGTCC pool=10
True Signal: fp=AACTA pool=11
True Signal: fp=AACTA pool=3
True Signal: fp=CACTC pool=7
True Signal: fp=CTCCA pool=6
True Signal: fp=AAGTA pool=7
True Signal: fp=CAGTA pool=8
True Signal: fp=GACTC pool=14
True Signal: fp=GTCCA pool=3
True Signal: fp=CTGCA pool=11
True Signal: fp=ATAGG pool=12
True Signal: fp=GTAGA pool=8
True Signal: fp=GTAGA pool=9
True Signal: fp=TGTCT pool=0
True Signal: fp=CAGTG pool=15
True Signal: fp=GTAGC pool=14
True Signal: fp=GTGCC pool=10
True Signal: fp=CAAAC pool=11
True Signal: fp=GTAGG pool=3
True Signal: fp=AAAAG pool=0
True Signal: fp=AAAAG pool=2
True Signal: fp=ACACG pool=5
True Signal: fp=GAAAG pool=14
True Signal: fp=CCCGA pool=15
True Signal: fp=AGCCC pool=10
True Signal: fp=AGAGA pool=13
True Signal: fp=ATGCT pool=6
True Signal: fp=AGAGC pool=14
True Signal: fp=GCTTA pool=9
True Signal: fp=AGGCC pool=12
True Signal: fp=CGGCA pool=10
True Signal: fp=GCCGA pool=7
True Signal: fp=CCTTG pool=2
True Signal: fp=GCTTC pool=5
True Signal: fp=TTCGC pool=10
True Signal: fp=GCACG pool=10
True Signal: fp=TTGGC pool=12
True Signal: fp=GTGCT pool=9
True Signal: fp=ACGGG pool=11
True Signal: fp=ACGGG pool=3
True Signal: fp=GCGGC pool=11
True Signal: fp=TAGAA pool=15
True Signal: fp=CCACT pool=13
True Signal: fp=GGGCG pool=2
True Signal: fp=TCAGA pool=9
True Signal: fp=CGTAA pool=6
True Signal: fp=TAGAC pool=11
True Signal: fp=CTTAT pool=13
True Signal: fp=AGCCT pool=0
True Signal: fp=CGTAC pool=7
True Signal: fp=CATCG pool=7
True Signal: fp=TCGCA pool=7
True Signal: fp=TCCCG pool=11
True Signal: fp=AGTAG pool=9
True Signal: fp=AGGCT pool=10
True Signal: fp=GGCCT pool=8
True Signal: fp=TCGCG pool=5
True Signal: fp=GGTAG pool=10
True Signal: fp=GGTAG pool=3
True Signal: fp=GGGCT pool=8
True Signal: fp=TGGGG pool=1
True Signal: fp=AGTAT pool=0
True Signal: fp=ATGTC pool=9
True Signal: fp=TGACT pool=9
True Signal: fp=CTGTC pool=11
True Signal: fp=GTCTC pool=4
True Signal: fp=CTGTG pool=3
True Signal: fp=CTAAA pool=14
True Signal: fp=ACATC pool=13
True Signal: fp=GTAAA pool=13
True Signal: fp=ATAAG pool=13
True Signal: fp=AGCTA pool=4
True Signal: fp=GTCTT pool=13
True Signal: fp=AGCTG pool=3
True Signal: fp=AGGTC pool=1
True Signal: fp=CGCTG pool=12
True Signal: fp=GGCTC pool=14
True Signal: fp=AGGTG pool=8
True Signal: fp=GGGTA pool=10
True Signal: fp=GGGTA pool=15
True Signal: fp=GGCTG pool=2
True Signal: fp=GGGTC pool=10
True Signal: fp=CGAAA pool=3
True Signal: fp=ATTCA pool=13
True Signal: fp=ATTCA pool=6
True Signal: fp=TTCAA pool=9
True Signal: fp=TTCAA pool=12
True Signal: fp=AACGA pool=11
True Signal: fp=ACGAA pool=13
True Signal: fp=ATTCC pool=2
True Signal: fp=CCGAA pool=12
True Signal: fp=CCGAA pool=14
True Signal: fp=CATTC pool=13
True Signal: fp=CCATT pool=11
True Signal: fp=GGGTG pool=6
True Signal: fp=AGAAG pool=0
True Signal: fp=CCCAG pool=3
True Signal: fp=CCCAG pool=5
True Signal: fp=CACGC pool=10
True Signal: fp=CTTCC pool=14
True Signal: fp=CTTCC pool=6
True Signal: fp=TTATT pool=0
True Signal: fp=GATTC pool=12
True Signal: fp=GATTC pool=14
True Signal: fp=CAGGA pool=15
True Signal: fp=GCATT pool=15
True Signal: fp=AGCTT pool=4
True Signal: fp=ATTCG pool=9
True Signal: fp=ATTCG pool=5
True Signal: fp=CGAAG pool=14
True Signal: fp=CACGG pool=9
True Signal: fp=AAGGG pool=13
True Signal: fp=GAGGC pool=11
True Signal: fp=GGCTT pool=11
True Signal: fp=AAACT pool=4
True Signal: fp=TCAAA pool=4
True Signal: fp=TCAAC pool=5
True Signal: fp=CAACT pool=4
True Signal: fp=AGAAT pool=10
True Signal: fp=AATTT pool=8
True Signal: fp=TACCC pool=5
True Signal: fp=ACGAT pool=1
True Signal: fp=CGAAT pool=12
True Signal: fp=TAAGG pool=1
True Signal: fp=AAGGT pool=9
True Signal: fp=AAGGT pool=12
True Signal: fp=GCTGA pool=12
True Signal: fp=TGCAG pool=5
True Signal: fp=TAGCG pool=5
True Signal: fp=GCGAT pool=14
True Signal: fp=GCTGC pool=10
True Signal: fp=GCTGG pool=1
True Signal: fp=GGTCG pool=0
True Signal: fp=TCAAT pool=4
True Signal: fp=TAAGT pool=2
True Signal: fp=CCTGT pool=5
True Signal: fp=TCTCG pool=12
True Signal: fp=TGTGA pool=9
True Signal: fp=GCTGT pool=2
True Signal: fp=GGTCT pool=13
True Signal: fp=CAATA pool=7
True Signal: fp=GAATA pool=0
True Signal: fp=GAATA pool=15
True Signal: fp=ATTTA pool=1
True Signal: fp=ATTTA pool=12

GGTAGGGGTA GACATCGCGT AAAAGGGGCG TACCCAGGAC CCCCCTTGGC TCAATAAGTA
GCGCTGGGGT GCTACTACGG GTCTCGACAC GCATTCAACT AAAAGCTTCC ATTCGCACGG
GCTTATTTAA CGAAGGTCGC GATAAGGTGC CGAATAGGCT GCAGAGCGGC AGCCTGTCCA
GTGAATGCTG TGAGGCCTCC AGCTGACTCA TGAGAGAAGC CCAGTATTCA AACTACGATT
CCACTCGACA ATTTAGGATG TCTTCCCGAA AGCTATCGGG TAGAATATCA GATTCGTTTA

DotsOn=286

GGTAGGGGTA GACATCGCGT AAAAGGGGCG TACCCAGGAC CCCCCTTGGC TCAATAAGTA
GCGCTGGGGT GCTACTACGG GTCTCGACAC GCATTCAACT AAAAGCTTCC ATTCGCACGG
GCTTATTTAA CGAAGGTCGC GATAAGGTGC CGAATAGGCT GCAGAGCGGC AGCCTGTCCA
GTGAATGCTG TGAGGCCTCC AGCTGACTCA TGAGAGAAGC CCAGTATTCA AACTACGATT
CCACTCGACA ATTTAGGATG TCTTCCCGAA AGCTATCGGG TAGAATATCA GATTCGTTTG

DotsOn=286

GGTAGGGGTA GACATCGCGT AAAAGGGGCG TACCCAGGAC CCCCCTTGGC TCAATAAGTA
GCGCTGGGGT GCTACTACGG GTCTCGACAC GCATTCAACT AAAAGCTTCC ATTCGCACGG
GCTTATTTAA CGAAGGTCGC GATAAGGTGC CGAATAGGCT GCAGAGCGGC AGCCTGTCCA
GTGAATGCTG TGAGGCCTCC AGCTGACTCA TGAGAGAAGC CCAGTATTCA AACTACGATT
CCACTCGACA ATTTAGGATG TCTTCCCGAA AGCTATCGGG TAGAATATCA GATTCGTTTT

DotsOn=286

GTAGGGGTAG ACATCGCGTA AAAGGGGCGT ACCCAGGACC CCCCTTGGCT CAATAAGTAG
CGCTGGGGTG CTACTACGGG TCTCGACACG CATTCAACTA AAAGCTTCCA TTCGCACGGG
CTTATTTAAC GAAGGTCGCG ATAAGGTGCC GAATAGGCTG CAGAGCGGCA GCCTGTCCAG
TGAATGCTGT GAGGCCTCCA GCTGACTCAT GAGAGAAGCC CAGTATTCAA ACTACGATTC
CACTCGACAA TTTAGGATGT CTTCCCGAAA GCTATCGGGT AGAATATCAG ATTCGTTTAA

 True solution  DotsOn=286

GTAGGGGTAG ACATCGCGTA AAAGGGGCGT ACCCAGGACC CCCCTTGGCT CAATAAGTAG
CGCTGGGGTG CTACTACGGG TCTCGACACG CATTCAACTA AAAGCTTCCA TTCGCACGGG
CTTATTTAAC GAAGGTCGCG ATAAGGTGCC GAATAGGCTG CAGAGCGGCA GCCTGTCCAG
TGAATGCTGT GAGGCCTCCA GCTGACTCAT GAGAGAAGCC CAGTATTCAA ACTACGATTC
CACTCGACAA TTTAGGATGT CTTCCCGAAA GCTATCGGGT AGAATATCAG ATTCGTTTTG

DotsOn=286

Solutions: 5
```

r300.100.0.out

```
Using pool D16
Using sequence r300

True Signal: fp=CTCGA pool=7
True Signal: fp=CTACG pool=1
True Signal: fp=CTACG pool=2
True Signal: fp=GTACC pool=0
True Signal: fp=ATCGC pool=1
True Signal: fp=GAATG pool=15
True Signal: fp=ATCGG pool=13
True Signal: fp=GTCGC pool=13
True Signal: fp=ACCCA pool=14
True Signal: fp=CTGGG pool=10
True Signal: fp=CAATT pool=3
True Signal: fp=GACAA pool=1
True Signal: fp=TACTA pool=3
True Signal: fp=ACCCC pool=6
True Signal: fp=AGACA pool=10
True Signal: fp=TTCCA pool=8
True Signal: fp=TTCCA pool=4
True Signal: fp=ACGCA pool=8
True Signal: fp=GACAC pool=2
True Signal: fp=CGACA pool=10
True Signal: fp=CGACA pool=11
True Signal: fp=CTACT pool=10
True Signal: fp=CCCCC pool=9
True Signal: fp=CCCCC pool=14
True Signal: fp=TTCCC pool=12
True Signal: fp=GCCCA pool=1
True Signal: fp=GAGAA pool=8
True Signal: fp=CCAGC pool=5
True Signal: fp=CAGAG pool=3
True Signal: fp=GCAGA pool=1
True Signal: fp=GCAGC pool=12
True Signal: fp=CGCGA pool=3
True Signal: fp=AGCGC pool=0
True Signal: fp=GGACC pool=1
True Signal: fp=CCAGG pool=7
True Signal: fp=TTAGG pool=1
True Signal: fp=GAGAG pool=1
True Signal: fp=TAAAA pool=11
True Signal: fp=AGCGG pool=4
True Signal: fp=ACTAA pool=15
True Signal: fp=CGGGC pool=4
True Signal: fp=ACTAC pool=4
True Signal: fp=ACTAC pool=7
True Signal: fp=AGGGG pool=9
True Signal: fp=AGGGG pool=5
True Signal: fp=TTTAA pool=15
True Signal: fp=GGGGC pool=7
True Signal: fp=CAGAT pool=11
True Signal: fp=CATGA pool=14
True Signal: fp=AATGC pool=1
True Signal: fp=CCCCT pool=13
True Signal: fp=GACAT pool=4
True Signal: fp=TCTTC pool=8
True Signal: fp=CCAGT pool=10
True Signal: fp=CCAGT pool=9
True Signal: fp=GCTAC pool=9
True Signal: fp=TTTAG pool=11
True Signal: fp=TGAGA pool=12
True Signal: fp=TGCCG pool=8
True Signal: fp=GCGCT pool=15
True Signal: fp=CGCGT pool=4
True Signal: fp=TGAGG pool=7
True Signal: fp=TCGGG pool=1
True Signal: fp=CGGGT pool=8
True Signal: fp=CGGGT pool=12
True Signal: fp=GGCGT pool=12
True Signal: fp=TATCA pool=4
True Signal: fp=ATATC pool=2
True Signal: fp=CTATC pool=6
True Signal: fp=GGGGT pool=11
True Signal: fp=GGGGT pool=14
True Signal: fp=TATCG pool=3
True Signal: fp=GCTAT pool=3
True Signal: fp=GATGT pool=0
True Signal: fp=TGGCT pool=6
True Signal: fp=CTCAA pool=15
True Signal: fp=ATCAG pool=6
True Signal: fp=CGATA pool=8
True Signal: fp=CTGAC pool=5
True Signal: fp=GTATT pool=11
True Signal: fp=ATGAG pool=8
True Signal: fp=GCCTC pool=0
True Signal: fp=GTGAA pool=2
True Signal: fp=GCGTA pool=0
True Signal: fp=GCGTA pool=9
True Signal: fp=GCCTG pool=12
True Signal: fp=GGATG pool=1
True Signal: fp=GTGAG pool=0
True Signal: fp=TTAAC pool=2
True Signal: fp=AAAGC pool=1
True Signal: fp=AAAGC pool=6
True Signal: fp=AAGCC pool=8
True Signal: fp=CTCAT pool=8
True Signal: fp=AGATT pool=12
True Signal: fp=CAGCC pool=10
True Signal: fp=CGCAC pool=4
True Signal: fp=AAAGG pool=1
True Signal: fp=GACCC pool=9
True Signal: fp=CCCTT pool=1
True Signal: fp=CGATT pool=11
True Signal: fp=GAAGC pool=5
True Signal: fp=TCATG pool=1
True Signal: fp=AGGAC pool=15
True Signal: fp=TGCTA pool=4
True Signal: fp=GAAGG pool=10
True Signal: fp=AATAA pool=2
True Signal: fp=TGCTG pool=9
True Signal: fp=GGCAG pool=1
True Signal: fp=GAGCG pool=3
True Signal: fp=CTTGG pool=1
True Signal: fp=ACAAT pool=6
True Signal: fp=ACTCA pool=7
True Signal: fp=TCCAC pool=10
True Signal: fp=AATAG pool=13
True Signal: fp=GATAA pool=1
True Signal: fp=TACGA pool=6
True Signal: fp=TATTC pool=2
True Signal: fp=CCTCC pool=3
True Signal: fp=TAACG pool=14
True Signal: fp=AAGCT pool=12
True Signal: fp=AAGCT pool=5
True Signal: fp=ACTCG pool=15
True Signal: fp=CAGCT pool=9
True Signal: fp=TCCAG pool=8
True Signal: fp=TCCAG pool=2
True Signal: fp=CGCAT pool=11
True Signal: fp=TCGAC pool=9
True Signal: fp=TCGAC pool=13
True Signal: fp=GCTCA pool=5
True Signal: fp=AGGAT pool=8
True Signal: fp=TAGGA pool=15
True Signal: fp=AGTGA pool=14
True Signal: fp=TAGGC pool=13
True Signal: fp=TACGG pool=7
True Signal: fp=TAGGG pool=13
True Signal: fp=AATAT pool=13
True Signal: fp=GGTGC pool=1
True Signal: fp=GGTGC pool=4
True Signal: fp=TCCAT pool=9
True Signal: fp=TGAAT pool=10
True Signal: fp=TATTT pool=6
True Signal: fp=TGTCC pool=10
True Signal: fp=AACTA pool=11
True Signal: fp=AACTA pool=3
True Signal: fp=CACTC pool=7
True Signal: fp=CTCCA pool=6
True Signal: fp=AAGTA pool=7
True Signal: fp=CAGTA pool=8
True Signal: fp=GACTC pool=14
True Signal: fp=GTCCA pool=3
True Signal: fp=CTGCA pool=11
True Signal: fp=ATAGG pool=12
True Signal: fp=GTAGA pool=8
True Signal: fp=GTAGA pool=9
True Signal: fp=TGTCT pool=0
True Signal: fp=CAGTG pool=15
True Signal: fp=GTAGC pool=14
True Signal: fp=GTGCC pool=10
True Signal: fp=CAAAC pool=11
True Signal: fp=GTAGG pool=3
True Signal: fp=AAAAG pool=0
True Signal: fp=AAAAG pool=2
True Signal: fp=ACACG pool=5
True Signal: fp=GAAAG pool=14
True Signal: fp=CCCGA pool=15
True Signal: fp=AGCCC pool=10
True Signal: fp=AGAGA pool=13
True Signal: fp=ATGCT pool=6
True Signal: fp=AGAGC pool=14
True Signal: fp=GCTTA pool=9
True Signal: fp=AGGCC pool=12
True Signal: fp=CGGCA pool=10
True Signal: fp=GCCGA pool=7
True Signal: fp=CCTTG pool=2
True Signal: fp=GCTTC pool=5
True Signal: fp=TTCGC pool=10
True Signal: fp=GCACG pool=10
True Signal: fp=TTGGC pool=12
True Signal: fp=GTGCT pool=9
True Signal: fp=ACGGG pool=11
True Signal: fp=ACGGG pool=3
True Signal: fp=GCGGC pool=11
True Signal: fp=TAGAA pool=15
True Signal: fp=CCACT pool=13
True Signal: fp=GGGCG pool=2
True Signal: fp=TCAGA pool=9
True Signal: fp=CGTAA pool=6
True Signal: fp=TAGAC pool=11
True Signal: fp=CTTAT pool=13
True Signal: fp=AGCCT pool=0
True Signal: fp=CGTAC pool=7
True Signal: fp=CATCG pool=7
True Signal: fp=TCGCA pool=7
True Signal: fp=TCCCG pool=11
True Signal: fp=AGTAG pool=9
True Signal: fp=AGGCT pool=10
True Signal: fp=GGCCT pool=8
True Signal: fp=TCGCG pool=5
True Signal: fp=GGTAG pool=10
True Signal: fp=GGTAG pool=3
True Signal: fp=GGGCT pool=8
True Signal: fp=TGGGG pool=1
True Signal: fp=AGTAT pool=0
True Signal: fp=ATGTC pool=9
True Signal: fp=TGACT pool=9
True Signal: fp=CTGTC pool=11
True Signal: fp=GTCTC pool=4
True Signal: fp=CTGTG pool=3
True Signal: fp=CTAAA pool=14
True Signal: fp=ACATC pool=13
True Signal: fp=GTAAA pool=13
True Signal: fp=ATAAG pool=13
True Signal: fp=AGCTA pool=4
True Signal: fp=GTCTT pool=13
True Signal: fp=AGCTG pool=3
True Signal: fp=AGGTC pool=1
True Signal: fp=CGCTG pool=12
True Signal: fp=GGCTC pool=14
True Signal: fp=AGGTG pool=8
True Signal: fp=GGGTA pool=10
True Signal: fp=GGGTA pool=15
True Signal: fp=GGCTG pool=2
True Signal: fp=GGGTC pool=10
True Signal: fp=CGAAA pool=3
True Signal: fp=ATTCA pool=13
True Signal: fp=ATTCA pool=6
True Signal: fp=TTCAA pool=9
True Signal: fp=TTCAA pool=12
True Signal: fp=AACGA pool=11
True Signal: fp=ACGAA pool=13
True Signal: fp=ATTCC pool=2
True Signal: fp=CCGAA pool=12
True Signal: fp=CCGAA pool=14
True Signal: fp=CATTC pool=13
True Signal: fp=CCATT pool=11
True Signal: fp=GGGTG pool=6
True Signal: fp=AGAAG pool=0
True Signal: fp=CCCAG pool=3
True Signal: fp=CCCAG pool=5
True Signal: fp=CACGC pool=10
True Signal: fp=CTTCC pool=14
True Signal: fp=CTTCC pool=6
True Signal: fp=TTATT pool=0
True Signal: fp=GATTC pool=12
True Signal: fp=GATTC pool=14
True Signal: fp=CAGGA pool=15
True Signal: fp=GCATT pool=15
True Signal: fp=AGCTT pool=4
True Signal: fp=ATTCG pool=9
True Signal: fp=ATTCG pool=5
True Signal: fp=CGAAG pool=14
True Signal: fp=CACGG pool=9
True Signal: fp=AAGGG pool=13
True Signal: fp=GAGGC pool=11
True Signal: fp=GGCTT pool=11
True Signal: fp=AAACT pool=4
True Signal: fp=TCAAA pool=4
True Signal: fp=TCAAC pool=5
True Signal: fp=CAACT pool=4
True Signal: fp=AGAAT pool=10
True Signal: fp=AATTT pool=8
True Signal: fp=TACCC pool=5
True Signal: fp=ACGAT pool=1
True Signal: fp=CGAAT pool=12
True Signal: fp=TAAGG pool=1
True Signal: fp=AAGGT pool=9
True Signal: fp=AAGGT pool=12
True Signal: fp=GCTGA pool=12
True Signal: fp=TGCAG pool=5
True Signal: fp=TAGCG pool=5
True Signal: fp=GCGAT pool=14
True Signal: fp=GCTGC pool=10
True Signal: fp=GCTGG pool=1
True Signal: fp=GGTCG pool=0
True Signal: fp=TCAAT pool=4
True Signal: fp=TAAGT pool=2
True Signal: fp=CCTGT pool=5
True Signal: fp=TCTCG pool=12
True Signal: fp=TGTGA pool=9
True Signal: fp=GCTGT pool=2
True Signal: fp=GGTCT pool=13
True Signal: fp=CAATA pool=7
True Signal: fp=GAATA pool=0
True Signal: fp=GAATA pool=15
True Signal: fp=ATTTA pool=1
True Signal: fp=ATTTA pool=12
False positive Signal: fp=CTCTG pool=11
False positive Signal: fp=AACAT pool=6
False positive Signal: fp=GTGTC pool=0
False positive Signal: fp=GTACT pool=0
False positive Signal: fp=GAGAT pool=14
False positive Signal: fp=GGTTG pool=9
False positive Signal: fp=CTTTT pool=6
False positive Signal: fp=AGTAA pool=8
False positive Signal: fp=GCGGC pool=11
False positive Signal: fp=ATATA pool=11
False positive Signal: fp=CAAGA pool=9
False positive Signal: fp=GGGTT pool=10
False positive Signal: fp=CACCT pool=1
False positive Signal: fp=AAATA pool=0
False positive Signal: fp=AGCAT pool=6
False positive Signal: fp=GTGAT pool=11
False positive Signal: fp=GGTAG pool=6
False positive Signal: fp=GACTT pool=3
False positive Signal: fp=CCGGA pool=14
False positive Signal: fp=CGATC pool=15
False positive Signal: fp=CTTGT pool=0
False positive Signal: fp=CGGCC pool=6
False positive Signal: fp=GCGGA pool=5
False positive Signal: fp=ACATA pool=9
False positive Signal: fp=TGATA pool=9
False positive Signal: fp=ATAGC pool=10
False positive Signal: fp=CTGGT pool=10
False positive Signal: fp=ATCCC pool=8
False positive Signal: fp=ATTAG pool=6
False positive Signal: fp=AGCTA pool=5
False positive Signal: fp=GGCGG pool=12
False positive Signal: fp=TATCA pool=1
False positive Signal: fp=TCAGG pool=4
False positive Signal: fp=GATAG pool=9
False positive Signal: fp=TTGGT pool=2
False positive Signal: fp=TGACG pool=9
False positive Signal: fp=CCCTC pool=0
False positive Signal: fp=AGATG pool=10
False positive Signal: fp=CCGGC pool=14
False positive Signal: fp=TATAT pool=11
False positive Signal: fp=CATTA pool=14
False positive Signal: fp=GAGTA pool=10
False positive Signal: fp=TATAA pool=11
False positive Signal: fp=CGGTG pool=11
False positive Signal: fp=CCCTA pool=10
False positive Signal: fp=GCATA pool=14
False positive Signal: fp=TGGTC pool=0
False positive Signal: fp=AGGTT pool=11
False positive Signal: fp=CATAC pool=15
False positive Signal: fp=TCAGC pool=10
False positive Signal: fp=GGACT pool=12
False positive Signal: fp=TGCTC pool=13
False positive Signal: fp=CCATA pool=1
False positive Signal: fp=AATTA pool=13
False positive Signal: fp=GCGAA pool=15
False positive Signal: fp=ACCGG pool=11
False positive Signal: fp=GTTCA pool=2
False positive Signal: fp=AGTAC pool=7
False positive Signal: fp=GAGTC pool=6
False positive Signal: fp=GTGCT pool=12
False positive Signal: fp=TCACT pool=9
False positive Signal: fp=CTACA pool=8
False positive Signal: fp=GACGA pool=2
False positive Signal: fp=GGTCG pool=9
False positive Signal: fp=CTCAA pool=15
False positive Signal: fp=TCACT pool=15
False positive Signal: fp=AGATC pool=12
False positive Signal: fp=GTCGG pool=10
False positive Signal: fp=GGGGA pool=5
False positive Signal: fp=TGGAG pool=1
False positive Signal: fp=GGAGT pool=9
False positive Signal: fp=TGCCA pool=7
False positive Signal: fp=AAATC pool=13
False positive Signal: fp=ACCGT pool=9
False positive Signal: fp=GACGC pool=8
False positive Signal: fp=TAAGT pool=4
False positive Signal: fp=TGACC pool=10
False positive Signal: fp=GGATC pool=11
False positive Signal: fp=GAAGG pool=7
False positive Signal: fp=CGATT pool=10
False positive Signal: fp=GCTAG pool=10
False positive Signal: fp=GTGGC pool=12
False positive Signal: fp=GAATC pool=13
False positive Signal: fp=CCATG pool=4
False positive Signal: fp=GATCA pool=10
False positive Signal: fp=CAGTA pool=3
False positive Signal: fp=CAACT pool=4
False positive Signal: fp=CGCCA pool=2
False positive Signal: fp=TATAG pool=1
False positive Signal: fp=TACTG pool=1
False positive Signal: fp=AAAGC pool=4
False positive Signal: fp=CGACG pool=14
False positive Signal: fp=GTACT pool=3
False positive Signal: fp=TAATG pool=7
False positive Signal: fp=CGCAC pool=10
False positive Signal: fp=GCCTC pool=0
False positive Signal: fp=AATTT pool=1
False positive Signal: fp=CTCAC pool=14
False positive Signal: fp=AGTCA pool=12
False positive Signal: fp=CAGAT pool=14
10mers:24448
11mers:3459
12mers:744
13mers:386
14mers:344
15mers:337
16mers:336
17mers:333
18mers:330
19mers:327
20mers:325
21mers:324
22mers:326
23mers:322
24mers:322
25mers:320
26mers:319
27mers:319
28mers:320
29mers:316
30mers:314
31mers:313
32mers:310
33mers:309
34mers:307
35mers:306
36mers:305
37mers:303
38mers:302
39mers:304
40mers:302
41mers:302
42mers:300
43mers:299
44mers:298
45mers:297
46mers:295
47mers:295
48mers:293
49mers:291
50mers:289
```

```
51mers:289
52mers:285
53mers:284
54mers:285
55mers:283
56mers:282
57mers:282
58mers:280
59mers:278
60mers:279
61mers:276
62mers:276
63mers:275
64mers:274
65mers:272
66mers:274
67mers:271
68mers:269
69mers:268
70mers:267
71mers:266
72mers:265
73mers:264
74mers:261
75mers:260
76mers:259
77mers:260
78mers:259
79mers:257
80mers:255
81mers:255
82mers:253
83mers:253
84mers:253
85mers:251
86mers:249
87mers:248
88mers:247
89mers:248
90mers:250
91mers:247
92mers:246
93mers:244
94mers:243
95mers:241
96mers:238
97mers:237
98mers:237
```

```
99mers:236
100mers:234
101mers:234
102mers:236
103mers:234
104mers:230
105mers:230
106mers:229
107mers:227
108mers:225
109mers:226
110mers:224
111mers:223
112mers:221
113mers:219
114mers:219
115mers:217
116mers:215
117mers:215
118mers:216
119mers:213
120mers:212
121mers:210
122mers:208
123mers:207
124mers:207
125mers:204
126mers:203
127mers:202
128mers:201
129mers:201
130mers:199
131mers:198
132mers:197
133mers:197
134mers:195
135mers:195
136mers:194
137mers:192
138mers:191
139mers:190
140mers:190
141mers:190
142mers:188
143mers:186
144mers:186
145mers:185
146mers:184
```

```
147mers:182
148mers:181
149mers:180
150mers:181
151mers:178
152mers:177
153mers:176
154mers:174
155mers:173
156mers:172
157mers:172
158mers:171
159mers:170
160mers:167
161mers:167
162mers:165
163mers:165
164mers:164
165mers:166
166mers:164
167mers:161
168mers:159
169mers:159
170mers:157
171mers:156
172mers:156
173mers:156
174mers:153
175mers:152
176mers:154
177mers:152
178mers:150
179mers:148
180mers:148
181mers:146
182mers:145
183mers:144
184mers:144
185mers:143
186mers:141
187mers:141
188mers:139
189mers:136
190mers:136
191mers:137
192mers:135
193mers:132
194mers:131
```

```
195mers:130
196mers:130
197mers:129
198mers:127
199mers:127
200mers:126
201mers:125
202mers:125
203mers:125
204mers:121
205mers:120
206mers:120
207mers:120
208mers:117
209mers:115
210mers:114
211mers:114
212mers:112
213mers:113
214mers:113
215mers:111
216mers:108
217mers:109
218mers:107
219mers:106
220mers:106
221mers:102
222mers:101
223mers:102
224mers:102
225mers:98
226mers:100
227mers:96
228mers:95
229mers:94
230mers:93
231mers:91
232mers:92
233mers:89
234mers:86
235mers:85
236mers:85
237mers:83
238mers:82
239mers:83
240mers:79
241mers:80
242mers:78
```

```
243mers:77
244mers:74
245mers:73
246mers:72
247mers:72
248mers:69
249mers:69
250mers:69
251mers:67
252mers:66
253mers:66
254mers:65
255mers:62
256mers:61
257mers:59
258mers:61
259mers:58
260mers:56
261mers:55
262mers:54
263mers:52
264mers:53
265mers:53
266mers:52
267mers:48
268mers:46
269mers:46
270mers:45
271mers:45
272mers:42
273mers:41
274mers:38
275mers:37
276mers:36
277mers:35
278mers:34
279mers:32
280mers:30
281mers:27
282mers:26
283mers:26
284mers:25
285mers:24
286mers:22
287mers:21
288mers:19
289mers:17
290mers:17
```

```
291mers:15
292mers:14
293mers:12
294mers:10
295mers:9
296mers:8
297mers:7
298mers:6
299mers:5
300mers:3

GTAGGGGTAG ACATCGCGTA AAAGGGGCGT ACCCAGGACC CCCCTTGGCT CAATAAGTAG
CGCTGGGGTG CTACTACGGG TCTCGACACG CATTCAACTA AAAGCTTCCA TTCGCACGGG
CTTATTTAAC GAAGGTCGCG ATAAGGTGCC GAATAGGCTG CAGAGCGGCA GCCTGTCCAG
TGAATGCTGT GAGGCCTCCA GCTGACTCAT GAGAGAAGCC CAGTATTCAA ACTACGATTC
CACTCGACAA TTTAGGATGT CTTCCCGAAA GCTATCGGGT AGAATATCAG ATTCGTTTAA

 True solution  DotsOn=286

GGTAGGGGTA GACATCGCGT AAAAGGGGCG TACCCAGGAC CCCCCTTGGC TCAATAAGTA
GCGCTGGGGT GCTACTACGG GTCTCGACAC GCATTCAACT AAAAGCTTCC ATTCGCACGG
GCTTATTTAA CGAAGGTCGC GATAAGGTGC CGAATAGGCT GCAGAGCGGC AGCCTGTCCA
GTGAATGCTG TGAGGCCTCC AGCTGACTCA TGAGAGAAGC CCAGTATTCA AACTACGATT
CCACTCGACA ATTTAGGATG TCTTCCCGAA AGCTATCGGG TAGAATATCA GATTCGTTTA

 DotsOn=286

Solutions: 2
```

r300.300.0.out

```
Using pool D16
Using sequence r300

True Signal: fp=CTCGA pool=7
True Signal: fp=CTACG pool=1
True Signal: fp=CTACG pool=2
True Signal: fp=GTACC pool=0
True Signal: fp=ATCGC pool=1
True Signal: fp=GAATG pool=15
True Signal: fp=ATCGG pool=13
True Signal: fp=GTCGC pool=13
True Signal: fp=ACCCA pool=14
True Signal: fp=CTGGG pool=10
True Signal: fp=CAATT pool=3
True Signal: fp=GACAA pool=1
True Signal: fp=TACTA pool=3
True Signal: fp=ACCCC pool=6
True Signal: fp=AGACA pool=10
True Signal: fp=TTCCA pool=8
True Signal: fp=TTCCA pool=4
True Signal: fp=ACGCA pool=8
True Signal: fp=GACAC pool=2
True Signal: fp=CGACA pool=10
True Signal: fp=CGACA pool=11
True Signal: fp=CTACT pool=10
True Signal: fp=CCCCC pool=9
True Signal: fp=CCCCC pool=14
True Signal: fp=TTCCC pool=12
True Signal: fp=GCCCA pool=1
True Signal: fp=GAGAA pool=8
True Signal: fp=CCAGC pool=5
True Signal: fp=CAGAG pool=3
True Signal: fp=GCAGA pool=1
True Signal: fp=GCAGC pool=12
True Signal: fp=CGCGA pool=3
True Signal: fp=AGCGC pool=0
True Signal: fp=GGACC pool=1
True Signal: fp=CCAGG pool=7
True Signal: fp=TTAGG pool=1
True Signal: fp=GAGAG pool=1
True Signal: fp=TAAAA pool=11
True Signal: fp=AGCGG pool=4
True Signal: fp=ACTAA pool=15
True Signal: fp=CGGGC pool=4
True Signal: fp=ACTAC pool=4
True Signal: fp=ACTAC pool=7
True Signal: fp=AGGGG pool=9
True Signal: fp=AGGGG pool=5
True Signal: fp=TTTAA pool=15
True Signal: fp=GGGGC pool=7
True Signal: fp=CAGAT pool=11
True Signal: fp=CATGA pool=14
True Signal: fp=AATGC pool=1
True Signal: fp=CCCCT pool=13
True Signal: fp=GACAT pool=4
True Signal: fp=TCTTC pool=8
True Signal: fp=CCAGT pool=10
True Signal: fp=CCAGT pool=9
True Signal: fp=GCTAC pool=9
True Signal: fp=TTTAG pool=11
True Signal: fp=TGAGA pool=12
True Signal: fp=TGCCG pool=8
True Signal: fp=GCGCT pool=15
True Signal: fp=CGCGT pool=4
True Signal: fp=TGAGG pool=7
True Signal: fp=TCGGG pool=1
True Signal: fp=CGGGT pool=8
True Signal: fp=CGGGT pool=12
True Signal: fp=GGCGT pool=12
True Signal: fp=TATCA pool=4
True Signal: fp=ATATC pool=2
True Signal: fp=CTATC pool=6
True Signal: fp=GGGGT pool=11
True Signal: fp=GGGGT pool=14
True Signal: fp=TATCG pool=3
True Signal: fp=GCTAT pool=3
True Signal: fp=GATGT pool=0
True Signal: fp=TGGCT pool=6
True Signal: fp=CTCAA pool=15
True Signal: fp=ATCAG pool=6
True Signal: fp=CGATA pool=8
True Signal: fp=CTGAC pool=5
True Signal: fp=GTATT pool=11
True Signal: fp=ATGAG pool=8
True Signal: fp=GCCTC pool=0
True Signal: fp=GTGAA pool=2
True Signal: fp=GCGTA pool=0
True Signal: fp=GCGTA pool=9
True Signal: fp=GCCTG pool=12
True Signal: fp=GGATG pool=1
True Signal: fp=GTGAG pool=0
True Signal: fp=TTAAC pool=2
True Signal: fp=AAAGC pool=1
True Signal: fp=AAAGC pool=6
True Signal: fp=AAGCC pool=8
True Signal: fp=CTCAT pool=8
True Signal: fp=AGATT pool=12
True Signal: fp=CAGCC pool=10
True Signal: fp=CGCAC pool=4
True Signal: fp=AAAGG pool=1
True Signal: fp=GACCC pool=9
True Signal: fp=CCCTT pool=1
True Signal: fp=CGATT pool=11
True Signal: fp=GAAGC pool=5
True Signal: fp=TCATG pool=1
True Signal: fp=AGGAC pool=15
True Signal: fp=TGCTA pool=4
True Signal: fp=GAAGG pool=10
True Signal: fp=AATAA pool=2
True Signal: fp=TGCTG pool=9
True Signal: fp=GGCAG pool=1
True Signal: fp=GAGCG pool=3
True Signal: fp=CTTGG pool=1
True Signal: fp=ACAAT pool=6
True Signal: fp=ACTCA pool=7
True Signal: fp=TCCAC pool=10
True Signal: fp=AATAG pool=13
True Signal: fp=GATAA pool=1
True Signal: fp=TACGA pool=6
True Signal: fp=TATTC pool=2
True Signal: fp=CCTCC pool=3
True Signal: fp=TAACG pool=14
True Signal: fp=AAGCT pool=12
True Signal: fp=AAGCT pool=5
True Signal: fp=ACTCG pool=15
True Signal: fp=CAGCT pool=9
True Signal: fp=TCCAG pool=8
True Signal: fp=TCCAG pool=2
True Signal: fp=CGCAT pool=11
True Signal: fp=TCGAC pool=9
True Signal: fp=TCGAC pool=13
True Signal: fp=GCTCA pool=5
True Signal: fp=AGGAT pool=8
True Signal: fp=TAGGA pool=15
True Signal: fp=AGTGA pool=14
True Signal: fp=TAGGC pool=13
True Signal: fp=TACGG pool=7
True Signal: fp=TAGGG pool=13
True Signal: fp=AATAT pool=13
True Signal: fp=GGTGC pool=1
True Signal: fp=GGTGC pool=4
True Signal: fp=TCCAT pool=9
True Signal: fp=TGAAT pool=10
True Signal: fp=TATTT pool=6
True Signal: fp=TGTCC pool=10
True Signal: fp=AACTA pool=11
True Signal: fp=AACTA pool=3
True Signal: fp=CACTC pool=7
True Signal: fp=CTCCA pool=6
True Signal: fp=AAGTA pool=7
True Signal: fp=CAGTA pool=8
True Signal: fp=GACTC pool=14
True Signal: fp=GTCCA pool=3
True Signal: fp=CTGCA pool=11
True Signal: fp=ATAGG pool=12
True Signal: fp=GTAGA pool=8
True Signal: fp=GTAGA pool=9
True Signal: fp=TGTCT pool=0
True Signal: fp=CAGTG pool=15
True Signal: fp=GTAGC pool=14
True Signal: fp=GTGCC pool=10
True Signal: fp=CAAAC pool=11
True Signal: fp=GTAGG pool=3
True Signal: fp=AAAAG pool=0
True Signal: fp=AAAAG pool=2
True Signal: fp=ACACG pool=5
True Signal: fp=GAAAG pool=14
True Signal: fp=CCCGA pool=15
True Signal: fp=AGCCC pool=10
True Signal: fp=AGAGA pool=13
True Signal: fp=ATGCT pool=6
True Signal: fp=AGAGC pool=14
True Signal: fp=GCTTA pool=9
True Signal: fp=AGGCC pool=12
True Signal: fp=CGGCA pool=10
True Signal: fp=GCCGA pool=7
True Signal: fp=CCTTG pool=2
True Signal: fp=GCTTC pool=5
True Signal: fp=TTCGC pool=10
True Signal: fp=GCACG pool=10
True Signal: fp=TTGGC pool=12
True Signal: fp=GTGCT pool=9
True Signal: fp=ACGGG pool=11
True Signal: fp=ACGGG pool=3
True Signal: fp=GCGGC pool=11
True Signal: fp=TAGAA pool=15
True Signal: fp=CCACT pool=13
True Signal: fp=GGGCG pool=2
True Signal: fp=TCAGA pool=9
True Signal: fp=CGTAA pool=6
True Signal: fp=TAGAC pool=11
True Signal: fp=CTTAT pool=13
True Signal: fp=AGCCT pool=0
True Signal: fp=CGTAC pool=7
True Signal: fp=CATCG pool=7
True Signal: fp=TCGCA pool=7
True Signal: fp=TCCCG pool=11
True Signal: fp=AGTAG pool=9
True Signal: fp=AGGCT pool=10
True Signal: fp=GGCCT pool=8
True Signal: fp=TCGCG pool=5
True Signal: fp=GGTAG pool=10
True Signal: fp=GGTAG pool=3
True Signal: fp=GGGCT pool=8
True Signal: fp=TGGGG pool=1
True Signal: fp=AGTAT pool=0
True Signal: fp=ATGTC pool=9
True Signal: fp=TGACT pool=9
True Signal: fp=CTGTC pool=11
True Signal: fp=GTCTC pool=4
True Signal: fp=CTGTG pool=3
True Signal: fp=CTAAA pool=14
True Signal: fp=ACATC pool=13
True Signal: fp=GTAAA pool=13
True Signal: fp=ATAAG pool=13
True Signal: fp=AGCTA pool=4
True Signal: fp=GTCTT pool=13
True Signal: fp=AGCTG pool=3
True Signal: fp=AGGTC pool=1
True Signal: fp=CGCTG pool=12
True Signal: fp=GGCTC pool=14
True Signal: fp=AGGTG pool=8
True Signal: fp=GGGTA pool=10
True Signal: fp=GGGTA pool=15
True Signal: fp=GGCTG pool=2
True Signal: fp=GGGTC pool=10
True Signal: fp=CGAAA pool=3
True Signal: fp=ATTCA pool=13
True Signal: fp=ATTCA pool=6
True Signal: fp=TTCAA pool=9
True Signal: fp=TTCAA pool=12
True Signal: fp=AACGA pool=11
True Signal: fp=ACGAA pool=13
True Signal: fp=ATTCC pool=2
True Signal: fp=CCGAA pool=12
True Signal: fp=CCGAA pool=14
True Signal: fp=CATTC pool=13
True Signal: fp=CCATT pool=11
True Signal: fp=GGGTG pool=6
True Signal: fp=AGAAG pool=0
True Signal: fp=CCCAG pool=3
True Signal: fp=CCCAG pool=5
True Signal: fp=CACGC pool=10
True Signal: fp=CTTCC pool=14
True Signal: fp=CTTCC pool=6
True Signal: fp=TTATT pool=0
True Signal: fp=GATTC pool=12
True Signal: fp=GATTC pool=14
True Signal: fp=CAGGA pool=15
True Signal: fp=GCATT pool=15
True Signal: fp=AGCTT pool=4
True Signal: fp=ATTCG pool=9
True Signal: fp=ATTCG pool=5
True Signal: fp=CGAAG pool=14
True Signal: fp=CACGG pool=9
True Signal: fp=AAGGG pool=13
True Signal: fp=GAGGC pool=11
True Signal: fp=GGCTT pool=11
True Signal: fp=AAACT pool=4
True Signal: fp=TCAAA pool=4
True Signal: fp=TCAAC pool=5
True Signal: fp=CAACT pool=4
True Signal: fp=AGAAT pool=10
True Signal: fp=AATTT pool=8
True Signal: fp=TACCC pool=5
True Signal: fp=ACGAT pool=1
True Signal: fp=CGAAT pool=12
True Signal: fp=TAAGG pool=1
True Signal: fp=AAGGT pool=9
True Signal: fp=AAGGT pool=12
True Signal: fp=GCTGA pool=12
True Signal: fp=TGCAG pool=5
True Signal: fp=TAGCG pool=5
True Signal: fp=GCGAT pool=14
True Signal: fp=GCTGC pool=10
True Signal: fp=GCTGG pool=1
True Signal: fp=GGTCG pool=0
True Signal: fp=TCAAT pool=4
True Signal: fp=TAAGT pool=2
True Signal: fp=CCTGT pool=5
True Signal: fp=TCTCG pool=12
True Signal: fp=TGTGA pool=9
True Signal: fp=GCTGT pool=2
True Signal: fp=GGTCT pool=13
True Signal: fp=CAATA pool=7
True Signal: fp=GAATA pool=0
True Signal: fp=GAATA pool=15
True Signal: fp=ATTTA pool=1
True Signal: fp=ATTTA pool=12
False positive Signal: fp=AAACT pool=2
False positive Signal: fp=CCAGG pool=0
False positive Signal: fp=TAGTA pool=4
False positive Signal: fp=TCCCT pool=13
False positive Signal: fp=CTGTG pool=7
False positive Signal: fp=GCGTA pool=13
False positive Signal: fp=TCTAG pool=0
False positive Signal: fp=ACCTA pool=0
False positive Signal: fp=CACTT pool=10
False positive Signal: fp=GGAAG pool=12
False positive Signal: fp=CCGAC pool=3
False positive Signal: fp=TAGGG pool=12
False positive Signal: fp=TAGCG pool=4
False positive Signal: fp=TCTCC pool=15
False positive Signal: fp=CAGAA pool=9
False positive Signal: fp=TGCGC pool=9
False positive Signal: fp=CGAAT pool=2
False positive Signal: fp=CCGAG pool=9
False positive Signal: fp=CATGC pool=4
False positive Signal: fp=GTATC pool=1
False positive Signal: fp=TCGCT pool=2
False positive Signal: fp=AGGTA pool=14
False positive Signal: fp=AACCC pool=13
False positive Signal: fp=TACCC pool=6
False positive Signal: fp=GTTAA pool=8
False positive Signal: fp=TGGAG pool=12
False positive Signal: fp=ATTCC pool=9
False positive Signal: fp=TCACA pool=15
False positive Signal: fp=CTGCT pool=3
False positive Signal: fp=TGCCG pool=2
False positive Signal: fp=ACTCG pool=4
False positive Signal: fp=CGCAC pool=14
False positive Signal: fp=CTTCG pool=15
False positive Signal: fp=CCTGG pool=0
False positive Signal: fp=AGAAG pool=2
False positive Signal: fp=CTTAA pool=3
False positive Signal: fp=ACGGT pool=9
False positive Signal: fp=CTTGG pool=3
False positive Signal: fp=AGATC pool=12
False positive Signal: fp=GACCG pool=5
False positive Signal: fp=CCGTT pool=8
False positive Signal: fp=CACTC pool=12
False positive Signal: fp=ATTGG pool=5
False positive Signal: fp=AACAC pool=14
False positive Signal: fp=GTACC pool=14
False positive Signal: fp=CCCGT pool=4
False positive Signal: fp=AGTGG pool=6
False positive Signal: fp=AGGTC pool=9
False positive Signal: fp=GAACC pool=1
False positive Signal: fp=GATTC pool=12
False positive Signal: fp=AAGCT pool=1
False positive Signal: fp=GCACC pool=7
False positive Signal: fp=GCCCT pool=5
False positive Signal: fp=GCTGC pool=0
False positive Signal: fp=GACAA pool=7
False positive Signal: fp=TCGCT pool=0
False positive Signal: fp=CGTAA pool=2
False positive Signal: fp=CGAGT pool=3
False positive Signal: fp=AATGC pool=7
False positive Signal: fp=AAACT pool=5
False positive Signal: fp=CGATG pool=7
False positive Signal: fp=ATCCA pool=14
False positive Signal: fp=GGTCG pool=1
False positive Signal: fp=ACCGC pool=2
False positive Signal: fp=TATCA pool=0
False positive Signal: fp=AATCC pool=4
False positive Signal: fp=GAGGA pool=14
False positive Signal: fp=TATAC pool=5
False positive Signal: fp=TCGCG pool=2
False positive Signal: fp=GAGGG pool=5
False positive Signal: fp=ATTGA pool=5
False positive Signal: fp=TCAGA pool=15
False positive Signal: fp=CGGCC pool=1
False positive Signal: fp=TCGCT pool=7
False positive Signal: fp=TCTCA pool=10
False positive Signal: fp=TCTGT pool=11
False positive Signal: fp=GTGGT pool=4
False positive Signal: fp=CTTCC pool=5
False positive Signal: fp=GACAA pool=14
False positive Signal: fp=CTGCC pool=5
False positive Signal: fp=CAACT pool=6
False positive Signal: fp=CGAAG pool=13
False positive Signal: fp=TCGCA pool=15
False positive Signal: fp=CTTGT pool=13
False positive Signal: fp=GGTCC pool=13
False positive Signal: fp=ATGTT pool=14
False positive Signal: fp=CGGCG pool=3
False positive Signal: fp=CGAGC pool=2
False positive Signal: fp=AAGCA pool=14
False positive Signal: fp=CAAGG pool=9
False positive Signal: fp=TGGCT pool=15
False positive Signal: fp=AGGAT pool=8
False positive Signal: fp=ACGGG pool=9
False positive Signal: fp=AGATG pool=15
False positive Signal: fp=CCCAA pool=0
False positive Signal: fp=ACTTC pool=1
False positive Signal: fp=TCCTT pool=15
False positive Signal: fp=CCAGG pool=6
False positive Signal: fp=TGCGT pool=4
False positive Signal: fp=CTACT pool=4
False positive Signal: fp=AATTG pool=3
False positive Signal: fp=GGAGC pool=6
False positive Signal: fp=AACAG pool=9
False positive Signal: fp=GGATT pool=12
False positive Signal: fp=ATGAA pool=8
False positive Signal: fp=AGGTT pool=11
False positive Signal: fp=GCCTT pool=2
False positive Signal: fp=TGCCG pool=12
False positive Signal: fp=ACTCC pool=13
False positive Signal: fp=ACCAG pool=13
False positive Signal: fp=CTCTG pool=4
False positive Signal: fp=CAGTT pool=15
False positive Signal: fp=CTAAG pool=10
False positive Signal: fp=ATCGG pool=0
False positive Signal: fp=CCGTC pool=5
False positive Signal: fp=TGCTC pool=4
False positive Signal: fp=ATCTG pool=4
False positive Signal: fp=GGCGT pool=6
False positive Signal: fp=TACCA pool=9
False positive Signal: fp=GTGGG pool=6
False positive Signal: fp=ACGTA pool=12
False positive Signal: fp=ACGTG pool=9
False positive Signal: fp=CTGTA pool=11
False positive Signal: fp=GCAGA pool=12
False positive Signal: fp=GCCGC pool=9
False positive Signal: fp=ATCAG pool=14
False positive Signal: fp=AAAAG pool=0
False positive Signal: fp=GTGGG pool=10
False positive Signal: fp=AACCA pool=5
False positive Signal: fp=GGACG pool=7
False positive Signal: fp=GCCGG pool=6
False positive Signal: fp=GCGAC pool=11
False positive Signal: fp=GCCAC pool=3
False positive Signal: fp=AGGCC pool=4
False positive Signal: fp=ACGCA pool=15
False positive Signal: fp=ACTGA pool=15
False positive Signal: fp=AATTC pool=10
False positive Signal: fp=GCAAC pool=0
False positive Signal: fp=GTTTA pool=7
False positive Signal: fp=AGCAA pool=2
False positive Signal: fp=GCAAC pool=7
False positive Signal: fp=CGAAA pool=14
False positive Signal: fp=GTGCA pool=4
False positive Signal: fp=GCTGT pool=5
False positive Signal: fp=AATGA pool=15
False positive Signal: fp=GATGA pool=4
False positive Signal: fp=GTAAG pool=2
False positive Signal: fp=GTCGG pool=1
False positive Signal: fp=TATAC pool=1
False positive Signal: fp=AAAGT pool=2
False positive Signal: fp=AGCGC pool=13
False positive Signal: fp=GTTCT pool=13
False positive Signal: fp=GGGCG pool=3
False positive Signal: fp=AAAAT pool=7
False positive Signal: fp=GTAGG pool=1
False positive Signal: fp=AAGAT pool=14
False positive Signal: fp=CATGC pool=3
False positive Signal: fp=CGGTG pool=7
False positive Signal: fp=AGAGT pool=9
False positive Signal: fp=GGATT pool=5
False positive Signal: fp=ATTAT pool=12
False positive Signal: fp=TGTGA pool=0
False positive Signal: fp=CTGAT pool=15
False positive Signal: fp=TGGTC pool=13
False positive Signal: fp=GTTTA pool=2
False positive Signal: fp=AAATC pool=1
False positive Signal: fp=TAGTA pool=3
False positive Signal: fp=AAACA pool=9
False positive Signal: fp=GTCGT pool=10
False positive Signal: fp=TCGTC pool=4
False positive Signal: fp=AAACT pool=10
False positive Signal: fp=AGCCT pool=5
False positive Signal: fp=CAGTC pool=9
False positive Signal: fp=AGATC pool=1
False positive Signal: fp=CTCTG pool=3
False positive Signal: fp=TGTCC pool=9
False positive Signal: fp=CTGCT pool=15
False positive Signal: fp=GGTAG pool=14
False positive Signal: fp=CTCTT pool=11
False positive Signal: fp=CCCTT pool=2
False positive Signal: fp=GAATA pool=14
False positive Signal: fp=TAACC pool=0
False positive Signal: fp=GCTAT pool=8
False positive Signal: fp=TACTG pool=2
False positive Signal: fp=ATGTT pool=3
False positive Signal: fp=GACGA pool=12
False positive Signal: fp=ACAAC pool=14
False positive Signal: fp=TCGAC pool=2
False positive Signal: fp=ATGGA pool=9
False positive Signal: fp=CAGTT pool=1
False positive Signal: fp=GGGCT pool=12
False positive Signal: fp=ACCGG pool=1
False positive Signal: fp=TGCGA pool=12
False positive Signal: fp=GGGTG pool=1
False positive Signal: fp=TGTCA pool=1
False positive Signal: fp=GCCCT pool=5
False positive Signal: fp=CGCTG pool=10
False positive Signal: fp=GCATG pool=11
False positive Signal: fp=TGGCT pool=12
False positive Signal: fp=CGGAG pool=13
False positive Signal: fp=CTCCG pool=3
False positive Signal: fp=CGAAA pool=0
False positive Signal: fp=ACTGG pool=2
False positive Signal: fp=ATCTT pool=6
False positive Signal: fp=AACCT pool=1
False positive Signal: fp=GGACG pool=10
False positive Signal: fp=CGATA pool=11
False positive Signal: fp=ATATA pool=7
False positive Signal: fp=TCGGT pool=10
False positive Signal: fp=TACCT pool=9
False positive Signal: fp=TCAAG pool=1
False positive Signal: fp=GTCGT pool=0
False positive Signal: fp=TATCA pool=1
False positive Signal: fp=GCTAC pool=10
False positive Signal: fp=GTCTT pool=11
False positive Signal: fp=GTATC pool=5
False positive Signal: fp=TCGCC pool=1
False positive Signal: fp=GTTTA pool=14
False positive Signal: fp=GCATT pool=6
False positive Signal: fp=TATAG pool=5
False positive Signal: fp=TCACC pool=5
False positive Signal: fp=TCGCA pool=11
False positive Signal: fp=AACCC pool=15
False positive Signal: fp=TATGC pool=6
False positive Signal: fp=TGGAT pool=0
False positive Signal: fp=TATCC pool=4
False positive Signal: fp=TCAGG pool=8
False positive Signal: fp=CACAA pool=4
False positive Signal: fp=TGCCC pool=11
False positive Signal: fp=GTTCT pool=5
False positive Signal: fp=TACAT pool=8
False positive Signal: fp=TGTTT pool=9
False positive Signal: fp=ACATT pool=7
False positive Signal: fp=AAGCT pool=1
False positive Signal: fp=CGGAC pool=2
False positive Signal: fp=AGAAT pool=13
False positive Signal: fp=AGGCG pool=6
False positive Signal: fp=GCTGT pool=1
False positive Signal: fp=GGGGT pool=1
False positive Signal: fp=TGGTG pool=2
False positive Signal: fp=TCGAT pool=9
False positive Signal: fp=GATCA pool=13
False positive Signal: fp=CCGGT pool=10
False positive Signal: fp=ATTGT pool=8
False positive Signal: fp=ATCAC pool=5
False positive Signal: fp=GGAAG pool=15
False positive Signal: fp=GACTA pool=0
False positive Signal: fp=TCTAT pool=0
False positive Signal: fp=AAGCT pool=15
False positive Signal: fp=ATTTA pool=5
False positive Signal: fp=GTTAA pool=7
False positive Signal: fp=ATAAT pool=12
False positive Signal: fp=AAGTC pool=9
False positive Signal: fp=GCCTA pool=9
False positive Signal: fp=AGCCA pool=4
False positive Signal: fp=AACGC pool=3
False positive Signal: fp=GGTAA pool=15
False positive Signal: fp=TACTA pool=11
False positive Signal: fp=GAGCC pool=6
False positive Signal: fp=AGAAT pool=6
False positive Signal: fp=AATTG pool=12
False positive Signal: fp=TGCCC pool=11
False positive Signal: fp=AGTAA pool=12
False positive Signal: fp=GTAGC pool=4
False positive Signal: fp=TCGAG pool=4
False positive Signal: fp=TGCAG pool=0
False positive Signal: fp=GAGTA pool=1
False positive Signal: fp=GTACC pool=11
False positive Signal: fp=TCCTG pool=5
False positive Signal: fp=CCTGA pool=10
False positive Signal: fp=GTATG pool=1
False positive Signal: fp=ACAGA pool=7
False positive Signal: fp=GCGTC pool=15
False positive Signal: fp=ATCGA pool=4
False positive Signal: fp=ATCCT pool=5
False positive Signal: fp=TCGTG pool=0
False positive Signal: fp=TCTCT pool=15
False positive Signal: fp=AGCAA pool=8
False positive Signal: fp=GCGCT pool=10
False positive Signal: fp=ACTTC pool=5
False positive Signal: fp=TCCAG pool=3
False positive Signal: fp=ACGCG pool=7
False positive Signal: fp=GAGCA pool=5
False positive Signal: fp=TCAAC pool=4
False positive Signal: fp=CCTTG pool=1
False positive Signal: fp=GAGAT pool=11
False positive Signal: fp=CTGAA pool=0
False positive Signal: fp=CTGGC pool=0
False positive Signal: fp=ACCTG pool=6
False positive Signal: fp=GATAC pool=13
False positive Signal: fp=TAGTG pool=7
False positive Signal: fp=TCGAC pool=13
False positive Signal: fp=ATTGA pool=15
False positive Signal: fp=TGTCG pool=2
False positive Signal: fp=CGTGC pool=6
False positive Signal: fp=CAGTG pool=10
False positive Signal: fp=GAGTC pool=11
False positive Signal: fp=AAGTT pool=11
False positive Signal: fp=AGAGA pool=2
False positive Signal: fp=ATATA pool=8
10mers:37056
11mers:6330
12mers:1360
13mers:536
14mers:412
15mers:395
16mers:390
17mers:382
18mers:379
19mers:376
20mers:372
21mers:372
22mers:377
23mers:371
24mers:369
25mers:367
26mers:363
27mers:365
28mers:371
29mers:366
30mers:359
31mers:360
32mers:356
33mers:358
34mers:359
35mers:359
36mers:352
37mers:346
38mers:343
39mers:340
40mers:342
41mers:344
42mers:343
```

```
43mers:337
44mers:335
45mers:333
46mers:334
47mers:335
48mers:334
49mers:333
50mers:325
51mers:323
52mers:321
53mers:322
54mers:324
55mers:323
56mers:319
57mers:319
58mers:319
59mers:318
60mers:319
61mers:315
62mers:315
63mers:312
64mers:309
65mers:312
66mers:312
67mers:309
68mers:308
69mers:304
70mers:302
71mers:301
72mers:297
73mers:297
74mers:298
75mers:295
76mers:290
77mers:288
78mers:290
79mers:287
80mers:284
81mers:284
82mers:284
83mers:285
84mers:283
85mers:284
86mers:282
87mers:278
88mers:276
89mers:276
90mers:278
```

```
91mers:282
92mers:277
93mers:270
94mers:270
95mers:269
96mers:268
97mers:270
98mers:269
99mers:267
100mers:265
101mers:266
102mers:265
103mers:265
104mers:261
105mers:258
106mers:258
107mers:260
108mers:254
109mers:250
110mers:250
111mers:248
112mers:246
113mers:244
114mers:245
115mers:247
116mers:248
117mers:245
118mers:245
119mers:241
120mers:239
121mers:235
122mers:234
123mers:236
124mers:235
125mers:235
126mers:232
127mers:230
128mers:232
129mers:232
130mers:226
131mers:224
132mers:220
133mers:221
134mers:219
135mers:219
136mers:220
137mers:217
138mers:213
```

```
139mers:213
140mers:213
141mers:211
142mers:211
143mers:208
144mers:211
145mers:210
146mers:207
147mers:205
148mers:209
149mers:208
150mers:203
151mers:198
152mers:196
153mers:196
154mers:194
155mers:197
156mers:194
157mers:190
158mers:188
159mers:187
160mers:186
161mers:188
162mers:187
163mers:184
164mers:184
165mers:186
166mers:184
167mers:183
168mers:182
169mers:178
170mers:174
171mers:174
172mers:174
173mers:169
174mers:168
175mers:170
176mers:170
177mers:166
178mers:166
179mers:164
180mers:165
181mers:167
182mers:161
183mers:159
184mers:159
185mers:159
186mers:155
```

```
187mers:156
188mers:154
189mers:151
190mers:150
191mers:154
192mers:152
193mers:150
194mers:150
195mers:144
196mers:143
197mers:144
198mers:140
199mers:141
200mers:142
201mers:137
202mers:136
203mers:136
204mers:135
205mers:134
206mers:132
207mers:129
208mers:128
209mers:124
210mers:123
211mers:123
212mers:122
213mers:122
214mers:123
215mers:121
216mers:119
217mers:121
218mers:121
219mers:121
220mers:120
221mers:115
222mers:111
223mers:112
224mers:112
225mers:109
226mers:111
227mers:107
228mers:104
229mers:104
230mers:103
231mers:101
232mers:102
233mers:99
234mers:96
```

```
235mers:94
236mers:91
237mers:92
238mers:92
239mers:90
240mers:85
241mers:84
242mers:82
243mers:80
244mers:79
245mers:80
246mers:78
247mers:77
248mers:75
249mers:74
250mers:75
251mers:74
252mers:72
253mers:71
254mers:74
255mers:72
256mers:68
257mers:65
258mers:66
259mers:63
260mers:62
261mers:61
262mers:59
263mers:58
264mers:57
265mers:59
266mers:60
267mers:60
268mers:56
269mers:52
270mers:50
271mers:51
272mers:48
273mers:48
274mers:49
275mers:43
276mers:40
277mers:41
278mers:40
279mers:39
280mers:38
281mers:32
282mers:29
```

```
283mers:29
284mers:29
285mers:27
286mers:26
287mers:23
288mers:19
289mers:17
290mers:17
291mers:15
292mers:13
293mers:12
294mers:9
295mers:7
296mers:6
297mers:5
298mers:4
299mers:3
300mers:1

GTAGGGGTAG ACATCGCGTA AAAGGGGCGT ACCCAGGACC CCCCTTGGCT CAATAAGTAG
CGCTGGGGTG CTACTACGGG TCTCGACACG CATTCAACTA AAAGCTTCCA TTCGCACGGG
CTTATTTAAC GAAGGTCGCG ATAAGGTGCC GAATAGGCTG CAGAGCGGCA GCCTGTCCAG
TGAATGCTGT GAGGCCTCCA GCTGACTCAT GAGAGAAGCC CAGTATTCAA ACTACGATTC
CACTCGACAA TTTAGGATGT CTTCCCGAAA GCTATCGGGT AGAATATCAG ATTCGTTTAA

True solution  DotsOn=286

Solutions: 1
```

r300.100.15.out

```
Using pool D16
Using sequence r300

True Signal: fp=CTCGA pool=7
True Signal: fp=CTACG pool=1
True Signal: fp=CTACG pool=2
True Signal: fp=GTACC pool=0
True Signal: fp=ATCGC pool=1
True Signal: fp=GAATG pool=15
True Signal: fp=ATCGG pool=13
True Signal: fp=GTCGC pool=13
True Signal: fp=ACCCA pool=14
True Signal: fp=CTGGG pool=10
True Signal: fp=CAATT pool=3
True Signal: fp=GACAA pool=1
True Signal: fp=TACTA pool=3
True Signal: fp=ACCCC pool=6
True Signal: fp=AGACA pool=10
True Signal: fp=TTCCA pool=8
True Signal: fp=TTCCA pool=4
True Signal: fp=ACGCA pool=8
True Signal: fp=GACAC pool=2
True Signal: fp=CGACA pool=10
True Signal: fp=CGACA pool=11
True Signal: fp=CTACT pool=10
True Signal: fp=CCCCC pool=9
True Signal: fp=CCCCC pool=14
True Signal: fp=TTCCC pool=12
True Signal: fp=GCCCA pool=1
True Signal: fp=GAGAA pool=8
True Signal: fp=CCAGC pool=5
True Signal: fp=CAGAG pool=3
True Signal: fp=GCAGA pool=1
True Signal: fp=GCAGC pool=12
True Signal: fp=CGCGA pool=3
True Signal: fp=AGCGC pool=0
True Signal: fp=GGACC pool=1
True Signal: fp=CCAGG pool=7
True Signal: fp=TTAGG pool=1
True Signal: fp=GAGAG pool=1
True Signal: fp=TAAAA pool=11
True Signal: fp=AGCGG pool=4
True Signal: fp=ACTAA pool=15
True Signal: fp=CGGGC pool=4
True Signal: fp=ACTAC pool=4
True Signal: fp=ACTAC pool=7
True Signal: fp=AGGGG pool=9
True Signal: fp=AGGGG pool=5
True Signal: fp=TTTAA pool=15
True Signal: fp=GGGGC pool=7
True Signal: fp=CAGAT pool=11
True Signal: fp=CATGA pool=14
True Signal: fp=AATGC pool=1
True Signal: fp=CCCCT pool=13
True Signal: fp=GACAT pool=4
True Signal: fp=TCTTC pool=8
True Signal: fp=CCAGT pool=10
True Signal: fp=CCAGT pool=9
True Signal: fp=GCTAC pool=9
True Signal: fp=TTTAG pool=11
True Signal: fp=TGAGA pool=12
True Signal: fp=TGCCG pool=8
True Signal: fp=GCGCT pool=15
True Signal: fp=CGCGT pool=4
True Signal: fp=TGAGG pool=7
True Signal: fp=TCGGG pool=1
True Signal: fp=CGGGT pool=8
True Signal: fp=CGGGT pool=12
True Signal: fp=GGCGT pool=12
True Signal: fp=TATCA pool=4
True Signal: fp=ATATC pool=2
True Signal: fp=CTATC pool=6
True Signal: fp=GGGGT pool=11
True Signal: fp=GGGGT pool=14
True Signal: fp=TATCG pool=3
True Signal: fp=GCTAT pool=3
True Signal: fp=GATGT pool=0
True Signal: fp=TGGCT pool=6
True Signal: fp=CTCAA pool=15
True Signal: fp=ATCAG pool=6
True Signal: fp=CGATA pool=8
True Signal: fp=CTGAC pool=5
True Signal: fp=GTATT pool=11
True Signal: fp=ATGAG pool=8
True Signal: fp=GCCTC pool=0
True Signal: fp=GTGAA pool=2
True Signal: fp=GCGTA pool=0
True Signal: fp=GCGTA pool=9
True Signal: fp=GCCTG pool=12
True Signal: fp=GGATG pool=1
True Signal: fp=GTGAG pool=0
True Signal: fp=TTAAC pool=2
True Signal: fp=AAAGC pool=1
True Signal: fp=AAAGC pool=6
True Signal: fp=AAGCC pool=8
True Signal: fp=CTCAT pool=8
True Signal: fp=AGATT pool=12
True Signal: fp=CAGCC pool=10
True Signal: fp=CGCAC pool=4
True Signal: fp=AAAGG pool=1
True Signal: fp=GACCC pool=9
True Signal: fp=CCCTT pool=1
True Signal: fp=CGATT pool=11
True Signal: fp=GAAGC pool=5
True Signal: fp=TCATG pool=1
True Signal: fp=AGGAC pool=15
True Signal: fp=TGCTA pool=4
True Signal: fp=GAAGG pool=10
True Signal: fp=AATAA pool=2
True Signal: fp=TGCTG pool=9
True Signal: fp=GGCAG pool=1
True Signal: fp=GAGCG pool=3
True Signal: fp=CTTGG pool=1
True Signal: fp=ACAAT pool=6
True Signal: fp=ACTCA pool=7
True Signal: fp=TCCAC pool=10
True Signal: fp=AATAG pool=13
True Signal: fp=GATAA pool=1
True Signal: fp=TACGA pool=6
True Signal: fp=TATTC pool=2
True Signal: fp=CCTCC pool=3
True Signal: fp=TAACG pool=14
True Signal: fp=AAGCT pool=12
True Signal: fp=AAGCT pool=5
True Signal: fp=ACTCG pool=15
True Signal: fp=CAGCT pool=9
True Signal: fp=TCCAG pool=8
True Signal: fp=TCCAG pool=2
True Signal: fp=CGCAT pool=11
True Signal: fp=TCGAC pool=9
True Signal: fp=TCGAC pool=13
True Signal: fp=GCTCA pool=5
True Signal: fp=AGGAT pool=8
True Signal: fp=TAGGA pool=15
True Signal: fp=AGTGA pool=14
True Signal: fp=TAGGC pool=13
True Signal: fp=TACGG pool=7
True Signal: fp=TAGGG pool=13
True Signal: fp=AATAT pool=13
True Signal: fp=GGTGC pool=1
True Signal: fp=GGTGC pool=4
True Signal: fp=TCCAT pool=9
True Signal: fp=TGAAT pool=10
True Signal: fp=TATTT pool=6
True Signal: fp=TGTCC pool=10
True Signal: fp=AACTA pool=11
True Signal: fp=AACTA pool=3
True Signal: fp=CACTC pool=7
True Signal: fp=CTCCA pool=6
True Signal: fp=AAGTA pool=7
True Signal: fp=CAGTA pool=8
True Signal: fp=GACTC pool=14
True Signal: fp=GTCCA pool=3
True Signal: fp=CTGCA pool=11
True Signal: fp=ATAGG pool=12
True Signal: fp=GTAGA pool=8
True Signal: fp=GTAGA pool=9
True Signal: fp=TGTCT pool=0
True Signal: fp=CAGTG pool=15
True Signal: fp=GTAGC pool=14
True Signal: fp=GTGCC pool=10
True Signal: fp=CAAAC pool=11
True Signal: fp=GTAGG pool=3
True Signal: fp=AAAAG pool=0
True Signal: fp=AAAAG pool=2
True Signal: fp=ACACG pool=5
True Signal: fp=GAAAG pool=14
True Signal: fp=CCCGA pool=15
True Signal: fp=AGCCC pool=10
True Signal: fp=AGAGA pool=13
True Signal: fp=ATGCT pool=6
True Signal: fp=AGAGC pool=14
True Signal: fp=GCTTA pool=9
True Signal: fp=AGGCC pool=12
True Signal: fp=CGGCA pool=10
True Signal: fp=GCCGA pool=7
True Signal: fp=CCTTG pool=2
True Signal: fp=GCTTC pool=5
True Signal: fp=TTCGC pool=10
True Signal: fp=GCACG pool=10
True Signal: fp=TTGGC pool=12
True Signal: fp=GTGCT pool=9
True Signal: fp=ACGGG pool=11
True Signal: fp=ACGGG pool=3
True Signal: fp=GCGGC pool=11
True Signal: fp=TAGAA pool=15
True Signal: fp=CCACT pool=13
True Signal: fp=GGGCG pool=2
True Signal: fp=TCAGA pool=9
True Signal: fp=CGTAA pool=6
True Signal: fp=TAGAC pool=11
True Signal: fp=CTTAT pool=13
True Signal: fp=AGCCT pool=0
True Signal: fp=CGTAC pool=7
True Signal: fp=CATCG pool=7
True Signal: fp=TCGCA pool=7
True Signal: fp=TCCCG pool=11
True Signal: fp=AGTAG pool=9
True Signal: fp=AGGCT pool=10
True Signal: fp=GGCCT pool=8
True Signal: fp=TCGCG pool=5
True Signal: fp=GGTAG pool=10
True Signal: fp=GGTAG pool=3
True Signal: fp=GGGCT pool=8
True Signal: fp=TGGGG pool=1
True Signal: fp=AGTAT pool=0
True Signal: fp=ATGTC pool=9
True Signal: fp=TGACT pool=9
True Signal: fp=CTGTC pool=11
True Signal: fp=GTCTC pool=4
True Signal: fp=CTGTG pool=3
True Signal: fp=CTAAA pool=14
True Signal: fp=ACATC pool=13
True Signal: fp=GTAAA pool=13
True Signal: fp=ATAAG pool=13
True Signal: fp=AGCTA pool=4
True Signal: fp=GTCTT pool=13
True Signal: fp=AGCTG pool=3
True Signal: fp=AGGTC pool=1
True Signal: fp=CGCTG pool=12
True Signal: fp=GGCTC pool=14
True Signal: fp=AGGTG pool=8
True Signal: fp=GGGTA pool=10
True Signal: fp=GGGTA pool=15
True Signal: fp=GGCTG pool=2
True Signal: fp=GGGTC pool=10
True Signal: fp=CGAAA pool=3
True Signal: fp=ATTCA pool=13
True Signal: fp=ATTCA pool=6
True Signal: fp=TTCAA pool=9
True Signal: fp=TTCAA pool=12
True Signal: fp=AACGA pool=11
True Signal: fp=ACGAA pool=13
True Signal: fp=ATTCC pool=2
True Signal: fp=CCGAA pool=12
True Signal: fp=CCGAA pool=14
True Signal: fp=CATTC pool=13
True Signal: fp=CCATT pool=11
True Signal: fp=GGGTG pool=6
True Signal: fp=AGAAG pool=0
True Signal: fp=CCCAG pool=3
True Signal: fp=CCCAG pool=5
True Signal: fp=CACGC pool=10
True Signal: fp=CTTCC pool=14
True Signal: fp=CTTCC pool=6
True Signal: fp=TTATT pool=0
True Signal: fp=GATTC pool=12
True Signal: fp=GATTC pool=14
True Signal: fp=CAGGA pool=15
True Signal: fp=GCATT pool=15
True Signal: fp=AGCTT pool=4
True Signal: fp=ATTCG pool=9
True Signal: fp=ATTCG pool=5
True Signal: fp=CGAAG pool=14
True Signal: fp=CACGG pool=9
True Signal: fp=AAGGG pool=13
True Signal: fp=GAGGC pool=11
True Signal: fp=GGCTT pool=11
True Signal: fp=AAACT pool=4
True Signal: fp=TCAAA pool=4
True Signal: fp=TCAAC pool=5
True Signal: fp=CAACT pool=4
True Signal: fp=AGAAT pool=10
True Signal: fp=AATTT pool=8
True Signal: fp=TACCC pool=5
True Signal: fp=ACGAT pool=1
True Signal: fp=CGAAT pool=12
True Signal: fp=TAAGG pool=1
True Signal: fp=AAGGT pool=9
True Signal: fp=AAGGT pool=12
True Signal: fp=GCTGA pool=12
True Signal: fp=TGCAG pool=5
True Signal: fp=TAGCG pool=5
True Signal: fp=GCGAT pool=14
True Signal: fp=GCTGC pool=10
True Signal: fp=GCTGG pool=1
True Signal: fp=GGTCG pool=0
True Signal: fp=TCAAT pool=4
True Signal: fp=TAAGT pool=2
True Signal: fp=CCTGT pool=5
True Signal: fp=TCTCG pool=12
True Signal: fp=TGTGA pool=9
True Signal: fp=GCTGT pool=2
True Signal: fp=GGTCT pool=13
True Signal: fp=CAATA pool=7
True Signal: fp=GAATA pool=0
True Signal: fp=GAATA pool=15
True Signal: fp=ATTTA pool=1
True Signal: fp=ATTTA pool=12
False positive Signal: fp=CAATT pool=6
False positive Signal: fp=AGAGT pool=4
False positive Signal: fp=TGCAC pool=15
False positive Signal: fp=CATCA pool=9
False positive Signal: fp=ACACG pool=1
False positive Signal: fp=GTTTG pool=5
False positive Signal: fp=CAGGT pool=12
False positive Signal: fp=TCACT pool=2
False positive Signal: fp=GGCAA pool=13
False positive Signal: fp=GCCTA pool=2
False positive Signal: fp=AGGAG pool=11
False positive Signal: fp=GGCCG pool=8
False positive Signal: fp=CTCGA pool=8
False positive Signal: fp=GGAGG pool=10
False positive Signal: fp=GACCT pool=7
False positive Signal: fp=CAGAG pool=14
False positive Signal: fp=ACTTC pool=11
False positive Signal: fp=AGACT pool=8
False positive Signal: fp=TGCTT pool=12
False positive Signal: fp=GGTCG pool=4
False positive Signal: fp=GATAC pool=8
False positive Signal: fp=AGGCG pool=4
False positive Signal: fp=TGCGG pool=3
False positive Signal: fp=GTCTC pool=7
False positive Signal: fp=ACCCA pool=10
False positive Signal: fp=ACATA pool=9
False positive Signal: fp=AAGGG pool=5
False positive Signal: fp=GCGAT pool=9
False positive Signal: fp=CTATT pool=11
False positive Signal: fp=TAGGT pool=8
False positive Signal: fp=GACCG pool=11
False positive Signal: fp=ACATT pool=1
False positive Signal: fp=GCTAC pool=2
False positive Signal: fp=ACAAT pool=7
False positive Signal: fp=AGGAC pool=7
False positive Signal: fp=GCCTC pool=13
False positive Signal: fp=CTAGT pool=9
False positive Signal: fp=AGTTA pool=8
False positive Signal: fp=ATAGA pool=14
False positive Signal: fp=ATTTC pool=10
False positive Signal: fp=CGATC pool=0
False positive Signal: fp=GCGTT pool=1
False positive Signal: fp=CGGAG pool=3
False positive Signal: fp=GTATG pool=8
False positive Signal: fp=TCGAA pool=4
False positive Signal: fp=ACATT pool=8
False positive Signal: fp=AAAAC pool=11
False positive Signal: fp=TGCGC pool=11
False positive Signal: fp=GCAAC pool=11
False positive Signal: fp=GGCAG pool=1
False positive Signal: fp=CGAGA pool=2
False positive Signal: fp=GTCAA pool=9
False positive Signal: fp=TCGAT pool=10
False positive Signal: fp=AGGAT pool=7
False positive Signal: fp=TCAGT pool=14
False positive Signal: fp=CGACG pool=14
False positive Signal: fp=GGAAG pool=11
False positive Signal: fp=GTCTG pool=6
False positive Signal: fp=TGCTC pool=13
False positive Signal: fp=TGCTC pool=15
False positive Signal: fp=CTAGC pool=13
False positive Signal: fp=GCCTT pool=1
False positive Signal: fp=CATAA pool=4
False positive Signal: fp=GCCAC pool=9
False positive Signal: fp=CAGCA pool=12
False positive Signal: fp=ATCGA pool=8
False positive Signal: fp=CAGCC pool=14
False positive Signal: fp=CGCGA pool=9
False positive Signal: fp=CAGCC pool=8
False positive Signal: fp=GGCTT pool=8
False positive Signal: fp=GGTCG pool=0
False positive Signal: fp=TATGA pool=14
False positive Signal: fp=CCCGC pool=10
False positive Signal: fp=AGCCG pool=0
False positive Signal: fp=CTAGC pool=10
False positive Signal: fp=AGTCT pool=1
False positive Signal: fp=GAGCT pool=7
False positive Signal: fp=ACCAA pool=10
False positive Signal: fp=GTCTT pool=3
False positive Signal: fp=GGGCG pool=5
False positive Signal: fp=GAGTT pool=1
False positive Signal: fp=AATGC pool=13
False positive Signal: fp=GAGGT pool=7
False positive Signal: fp=TACTA pool=3
False positive Signal: fp=TACTT pool=7
False positive Signal: fp=CTCCA pool=5
False positive Signal: fp=GATAA pool=0
False positive Signal: fp=TGTAT pool=0
False positive Signal: fp=GACCG pool=5
False positive Signal: fp=TCTAT pool=11
False positive Signal: fp=CTCTA pool=15
False positive Signal: fp=TAACG pool=14
False positive Signal: fp=TCTGC pool=6
False positive Signal: fp=CCTCA pool=15
False positive Signal: fp=GAGCT pool=2
False positive Signal: fp=CGGCT pool=0
False positive Signal: fp=GCCGA pool=9
False positive Signal: fp=TAAAC pool=7
False positive Signal: fp=TAGGT pool=8
False positive Signal: fp=GGGAT pool=12
False negative : fp= pool=
False negative : fp=CTCGA pool=7
False negative : fp=CTACG pool=1
False negative : fp=CTACG pool=2
False negative : fp=GTACC pool=0
False negative : fp=ATCGC pool=1
False negative : fp=GAATG pool=15
False negative : fp=ATCGG pool=13
False negative : fp=GTCGC pool=13
False negative : fp=ACCCA pool=14
False negative : fp=CTGGG pool=10
False negative : fp=CAATT pool=3
False negative : fp=GACAA pool=1
False negative : fp=TACTA pool=3
False negative : fp=ACCCC pool=6
10mers:23488
11mers:20478
12mers:15215
13mers:10346
14mers:7890
15mers:5945
16mers:5080
17mers:4433
18mers:4074
19mers:3825
20mers:3745
21mers:3700
22mers:3705
23mers:3680
24mers:3668
25mers:3676
26mers:3670
27mers:3688
28mers:3719
29mers:3742
30mers:3734
31mers:3767
32mers:3837
33mers:3855
34mers:3867
35mers:3953
```

```
36mers:3981
37mers:3995
38mers:4024
39mers:4041
40mers:4058
41mers:4039
42mers:4085
43mers:4135
44mers:4217
45mers:4386
46mers:4528
47mers:4608
48mers:4641
49mers:4644
50mers:4662
51mers:4705
52mers:4786
53mers:4845
54mers:4875
55mers:4899
56mers:4935
57mers:4925
58mers:4943
59mers:4993
60mers:5058
61mers:5142
62mers:5174
63mers:5221
64mers:5262
65mers:5295
66mers:5287
67mers:5312
68mers:5383
69mers:5483
70mers:5601
71mers:5707
72mers:5814
73mers:5885
74mers:5954
75mers:6047
76mers:6110
77mers:6127
78mers:6109
79mers:6137
80mers:6176
81mers:6186
82mers:6242
83mers:6311
```

```
84mers:6361
85mers:6382
86mers:6372
87mers:6417
88mers:6464
89mers:6507
90mers:6610
91mers:6646
92mers:6616
93mers:6595
94mers:6584
95mers:6631
96mers:6684
97mers:6771
98mers:6832
99mers:6829
100mers:6841
101mers:6887
102mers:6853
103mers:6867
104mers:6882
105mers:6897
106mers:6957
107mers:7050
108mers:7186
109mers:7307
110mers:7360
111mers:7470
112mers:7521
113mers:7502
114mers:7556
115mers:7560
116mers:7605
117mers:7619
118mers:7587
119mers:7614
120mers:7620
121mers:7630
122mers:7664
123mers:7626
124mers:7592
125mers:7575
126mers:7532
127mers:7528
128mers:7487
129mers:7419
130mers:7372
131mers:7363
```

```
132mers:7396
133mers:7453
134mers:7442
135mers:7436
136mers:7425
137mers:7365
138mers:7383
139mers:7426
140mers:7429
141mers:7487
142mers:7491
143mers:7446
144mers:7414
145mers:7405
146mers:7429
147mers:7434
148mers:7497
149mers:7558
150mers:7550
151mers:5291
152mers:5258
153mers:5165
154mers:5051
155mers:4937
156mers:4850
157mers:4858
158mers:4844
159mers:4796
160mers:4755
161mers:4666
162mers:4602
163mers:4557
164mers:4509
165mers:4503
166mers:4487
167mers:4478
168mers:4466
169mers:4432
170mers:4407
171mers:4389
172mers:4342
173mers:4332
174mers:4266
175mers:4166
176mers:4115
177mers:4031
178mers:3959
179mers:3857
```

```
180mers:3758
181mers:3718
182mers:3685
183mers:3632
184mers:3575
185mers:3498
186mers:3454
187mers:3434
188mers:3427
189mers:3424
190mers:3396
191mers:3361
192mers:3340
193mers:3271
194mers:3218
195mers:3200
196mers:3130
197mers:3091
198mers:3067
199mers:3020
200mers:3013
201mers:3011
202mers:3032
203mers:3015
204mers:2876
205mers:2800
206mers:2757
207mers:2733
208mers:2740
209mers:2680
210mers:2610
211mers:2558
212mers:2511
213mers:2513
214mers:2473
215mers:2397
216mers:2317
217mers:2208
218mers:2143
219mers:2141
220mers:2118
221mers:2114
222mers:2144
223mers:2121
224mers:2104
225mers:2077
226mers:2077
227mers:2029
```

```
228mers:1924
229mers:1870
230mers:1823
231mers:1781
232mers:1772
233mers:1731
234mers:1625
235mers:1561
236mers:1515
237mers:1493
238mers:1442
239mers:1379
240mers:1323
241mers:1246
242mers:1195
243mers:1197
244mers:1160
245mers:1137
246mers:1127
247mers:1099
248mers:1095
249mers:1076
250mers:1046
251mers:991
252mers:944
253mers:916
254mers:901
255mers:881
256mers:877
257mers:862
258mers:818
259mers:789
260mers:771
261mers:754
262mers:728
263mers:698
264mers:663
265mers:610
266mers:566
267mers:555
268mers:521
269mers:474
270mers:418
271mers:367
272mers:343
273mers:326
274mers:316
275mers:294
```

```
276mers:263
277mers:236
278mers:219
279mers:214
280mers:218
281mers:220
282mers:218
283mers:209
284mers:199
285mers:194
286mers:196
287mers:187
288mers:174
289mers:161
290mers:139
291mers:123
292mers:114
293mers:101
294mers:79
295mers:58
296mers:47
297mers:37
298mers:27
299mers:18
300mers:11
```

```
GGTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCT
GGGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAA
CGAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGG
CCTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATG
TCTTCCCGAAAGCTATCGGGTAGAATATCAGATTCGTTTT
  DotsOn=286
```

```
GTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCTG
GGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAAC
GAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGGC
CTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATGT
CTTCCCGAAAGCTATCGGGTAGAATATCAGATTCGTTTTG
  DotsOn=286
```

```
GGGTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGC
TGGGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTA
ACGAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAG
GCCTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGAT
GTCTTCCCGAAAGCTATCGGGTAGAATATCAGATTCGTTT
  DotsOn=285
```

```
GGTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCT
GGGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAA
CGAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGG
CCTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATG
TCTTCCCGAAAGCTATCGGGTAGAATATCAGATTCGTTTG
   DotsOn=286

GGGTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGC
TGGGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTA
ACGAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAG
GCCTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGAT
GTCTTCCCGAAAGCTATCGGGTAGAATATCAGATTGTAGT
   DotsOn=285

GTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCTG
GGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAAC
GAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGGC
CTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATGT
CTTCCCGAAAGCTATCGGGTAGAATATCAGATTCGTTTAA
   True solution  DotsOn=286

GTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCTG
GGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAAC
GAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGGC
CTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATGT
CTTCCCGAAAGCTATCGGGTAGAATATCAGATTCCCATGT
   DotsOn=284

GGTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCT
GGGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAA
CGAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGG
CCTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATG
TCTTCCCGAAAGCTATCGGGTAGAATATCAGATTCCCATG
   DotsOn=285

GGGTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGC
TGGGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTA
ACGAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAG
GCCTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGAT
GTCTTCCCGAAAGCTATCGGGTAGAATATCAGATTCCCAT
   DotsOn=285

GGTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCT
GGGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAA
CGAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGG
CCTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATG
TCTTCCCGAAAGCTATCGGGTAGAATATCAGATTCGTTTA
   DotsOn=286
```

```
GTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCTG
GGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAAC
GAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGGC
CTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATGT
CTTCCCGAAAGCTATCGGGTAGAATATCAGATTCGTTTGA
  DotsOn=285

Solutions: 11
```

r300.0.0.DN16.out

```
Using pool DN16
Using sequence r300

True Signal: fp=CTCGA pool=7
True Signal: fp=CTACG pool=1
True Signal: fp=CTACG pool=2
True Signal: fp=GTACC pool=0
True Signal: fp=ATCGC pool=1
True Signal: fp=GAATG pool=15
True Signal: fp=ATCGG pool=13
True Signal: fp=GTCGC pool=13
True Signal: fp=ACCCA pool=14
True Signal: fp=CTGGG pool=10
True Signal: fp=CAATT pool=3
True Signal: fp=GACAA pool=1
True Signal: fp=TACTA pool=3
True Signal: fp=ACCCC pool=6
True Signal: fp=AGACA pool=10
True Signal: fp=TTCCA pool=8
True Signal: fp=TTCCA pool=4
True Signal: fp=ACGCA pool=8
True Signal: fp=GACAC pool=2
True Signal: fp=CGACA pool=10
True Signal: fp=CGACA pool=11
True Signal: fp=CTACT pool=10
True Signal: fp=CCCCC pool=2
True Signal: fp=CCCCC pool=14
True Signal: fp=TTCCC pool=12
True Signal: fp=GCCCA pool=1
True Signal: fp=GAGAA pool=8
True Signal: fp=CCAGC pool=5
True Signal: fp=CAGAG pool=3
True Signal: fp=GCAGA pool=1
True Signal: fp=GCAGC pool=12
True Signal: fp=CGCGA pool=3
True Signal: fp=AGCGC pool=0
True Signal: fp=GGACC pool=1
True Signal: fp=CCAGG pool=7
True Signal: fp=TTAGG pool=1
True Signal: fp=GAGAG pool=6
True Signal: fp=TAAAA pool=11
True Signal: fp=AGCGG pool=4
True Signal: fp=ACTAA pool=15
True Signal: fp=CGGGC pool=4
True Signal: fp=ACTAC pool=4
True Signal: fp=ACTAC pool=7
True Signal: fp=AGGGG pool=9
True Signal: fp=AGGGG pool=5
True Signal: fp=TTTAA pool=15
True Signal: fp=GGGGC pool=7
True Signal: fp=CAGAT pool=11
True Signal: fp=CATGA pool=14
True Signal: fp=AATGC pool=1
True Signal: fp=CCCCT pool=13
True Signal: fp=GACAT pool=4
True Signal: fp=TCTTC pool=8
True Signal: fp=CCAGT pool=10
True Signal: fp=CCAGT pool=9
True Signal: fp=GCTAC pool=9
True Signal: fp=TTTAG pool=11
True Signal: fp=TGAGA pool=12
True Signal: fp=TGCCG pool=8
True Signal: fp=GCGCT pool=15
True Signal: fp=CGCGT pool=4
True Signal: fp=TGAGG pool=5
True Signal: fp=TCGGG pool=1
True Signal: fp=CGGGT pool=8
True Signal: fp=CGGGT pool=12
True Signal: fp=GGCGT pool=12
True Signal: fp=TATCA pool=4
True Signal: fp=ATATC pool=9
True Signal: fp=CTATC pool=6
True Signal: fp=GGGGT pool=11
True Signal: fp=GGGGT pool=14
True Signal: fp=TATCG pool=3
True Signal: fp=GCTAT pool=3
True Signal: fp=GATGT pool=0
True Signal: fp=TGGCT pool=6
True Signal: fp=CTCAA pool=15
True Signal: fp=ATCAG pool=6
True Signal: fp=CGATA pool=2
True Signal: fp=CTGAC pool=5
True Signal: fp=GTATT pool=11
True Signal: fp=ATGAG pool=8
True Signal: fp=GCCTC pool=11
True Signal: fp=GTGAA pool=2
True Signal: fp=GCGTA pool=0
True Signal: fp=GCGTA pool=9
True Signal: fp=GCCTG pool=12
True Signal: fp=GGATG pool=1
True Signal: fp=GTGAG pool=0
True Signal: fp=TTAAC pool=6
True Signal: fp=AAAGC pool=1
True Signal: fp=AAAGC pool=6
True Signal: fp=AAGCC pool=8
True Signal: fp=CTCAT pool=8
True Signal: fp=AGATT pool=12
True Signal: fp=CAGCC pool=10
True Signal: fp=CGCAC pool=3
True Signal: fp=AAAGG pool=1
True Signal: fp=GACCC pool=9
True Signal: fp=CCCTT pool=1
True Signal: fp=CGATT pool=11
True Signal: fp=GAAGC pool=5
True Signal: fp=TCATG pool=1
True Signal: fp=AGGAC pool=6
True Signal: fp=TGCTA pool=4
True Signal: fp=GAAGG pool=10
True Signal: fp=AATAA pool=2
True Signal: fp=TGCTG pool=9
True Signal: fp=GGCAG pool=1
True Signal: fp=GAGCG pool=3
True Signal: fp=CTTGG pool=1
True Signal: fp=ACAAT pool=6
True Signal: fp=ACTCA pool=7
True Signal: fp=TCCAC pool=10
True Signal: fp=AATAG pool=13
True Signal: fp=GATAA pool=1
True Signal: fp=TACGA pool=6
True Signal: fp=TATTC pool=2
True Signal: fp=CCTCC pool=3
True Signal: fp=TAACG pool=14
True Signal: fp=AAGCT pool=12
True Signal: fp=AAGCT pool=5
True Signal: fp=ACTCG pool=15
True Signal: fp=CAGCT pool=9
True Signal: fp=TCCAG pool=8
True Signal: fp=CGCAT pool=11
True Signal: fp=TCGAC pool=9
True Signal: fp=TCGAC pool=5
True Signal: fp=GCTCA pool=5
True Signal: fp=AGGAT pool=8
True Signal: fp=TAGGA pool=15
True Signal: fp=AGTGA pool=14
True Signal: fp=TAGGC pool=13
True Signal: fp=TACGG pool=7
True Signal: fp=TAGGG pool=13
True Signal: fp=AATAT pool=13
True Signal: fp=GGTGC pool=1
True Signal: fp=GGTGC pool=5
True Signal: fp=TCCAT pool=9
True Signal: fp=TGAAT pool=10
True Signal: fp=TATTT pool=6
True Signal: fp=TGTCC pool=10
True Signal: fp=AACTA pool=1
True Signal: fp=AACTA pool=3
True Signal: fp=CACTC pool=7
True Signal: fp=CTCCA pool=6
True Signal: fp=AAGTA pool=7
True Signal: fp=CAGTA pool=8
True Signal: fp=GACTC pool=14
True Signal: fp=GTCCA pool=3
True Signal: fp=CTGCA pool=11
True Signal: fp=ATAGG pool=14
True Signal: fp=GTAGA pool=8
True Signal: fp=GTAGA pool=9
True Signal: fp=TGTCT pool=0
True Signal: fp=CAGTG pool=15
True Signal: fp=GTAGC pool=14
True Signal: fp=GTGCC pool=10
True Signal: fp=CAAAC pool=11
True Signal: fp=GTAGG pool=3
True Signal: fp=AAAAG pool=0
True Signal: fp=AAAAG pool=2
True Signal: fp=ACACG pool=5
True Signal: fp=GAAAG pool=14
True Signal: fp=CCCGA pool=15
True Signal: fp=AGCCC pool=10
True Signal: fp=AGAGA pool=13
True Signal: fp=ATGCT pool=6
True Signal: fp=AGAGC pool=14
True Signal: fp=GCTTA pool=9
True Signal: fp=AGGCC pool=12
True Signal: fp=CGGCA pool=10
True Signal: fp=GCCGA pool=7
True Signal: fp=CCTTG pool=2
True Signal: fp=GCTTC pool=5
True Signal: fp=TTCGC pool=10
True Signal: fp=GCACG pool=10
True Signal: fp=TTGGC pool=12
True Signal: fp=GTGCT pool=9
True Signal: fp=ACGGG pool=0
True Signal: fp=ACGGG pool=3
True Signal: fp=GCGGC pool=11
True Signal: fp=TAGAA pool=2
True Signal: fp=CCACT pool=13
True Signal: fp=GGGCG pool=2
True Signal: fp=TCAGA pool=9
True Signal: fp=CGTAA pool=12
True Signal: fp=TAGAC pool=11
True Signal: fp=CTTAT pool=13
True Signal: fp=AGCCT pool=0
True Signal: fp=CGTAC pool=7
True Signal: fp=CATCG pool=7
True Signal: fp=TCGCA pool=7
True Signal: fp=TCCCG pool=1
True Signal: fp=AGTAG pool=9
True Signal: fp=AGGCT pool=10
True Signal: fp=GGCCT pool=8
True Signal: fp=TCGCG pool=5
True Signal: fp=GGTAG pool=10
True Signal: fp=GGTAG pool=3
True Signal: fp=GGGCT pool=8
True Signal: fp=TGGGG pool=1
True Signal: fp=AGTAT pool=0
True Signal: fp=ATGTC pool=9
True Signal: fp=TGACT pool=9
True Signal: fp=CTGTC pool=11
True Signal: fp=GTCTC pool=4
True Signal: fp=CTGTG pool=3
True Signal: fp=CTAAA pool=14
True Signal: fp=ACATC pool=13
True Signal: fp=GTAAA pool=13
True Signal: fp=ATAAG pool=13
True Signal: fp=AGCTA pool=4
True Signal: fp=GTCTT pool=13
True Signal: fp=AGCTG pool=4
True Signal: fp=AGGTC pool=1
True Signal: fp=CGCTG pool=12
True Signal: fp=GGCTC pool=14
True Signal: fp=AGGTG pool=8
True Signal: fp=GGGTA pool=10
True Signal: fp=GGGTA pool=15
True Signal: fp=GGCTG pool=2
True Signal: fp=GGGTC pool=10
True Signal: fp=CGAAA pool=3
True Signal: fp=ATTCA pool=13
True Signal: fp=ATTCA pool=6
True Signal: fp=TTCAA pool=9
True Signal: fp=TTCAA pool=12
True Signal: fp=AACGA pool=11
True Signal: fp=ACGAA pool=13
True Signal: fp=ATTCC pool=2
True Signal: fp=CCGAA pool=12
True Signal: fp=CCGAA pool=14
True Signal: fp=CATTC pool=13
True Signal: fp=CCATT pool=11
True Signal: fp=GGGTG pool=6
True Signal: fp=AGAAG pool=0
True Signal: fp=CCCAG pool=3
True Signal: fp=CCCAG pool=5
True Signal: fp=CACGC pool=10
True Signal: fp=CTTCC pool=14
True Signal: fp=CTTCC pool=6
True Signal: fp=TTATT pool=0
True Signal: fp=GATTC pool=12
True Signal: fp=GATTC pool=14
True Signal: fp=CAGGA pool=6
True Signal: fp=GCATT pool=15
True Signal: fp=AGCTT pool=4
True Signal: fp=ATTCG pool=9
True Signal: fp=ATTCG pool=5
True Signal: fp=CGAAG pool=14
True Signal: fp=CACGG pool=9
True Signal: fp=AAGGG pool=13
True Signal: fp=GAGGC pool=11
True Signal: fp=GGCTT pool=11
True Signal: fp=AAACT pool=4
True Signal: fp=TCAAA pool=4
True Signal: fp=TCAAC pool=5
True Signal: fp=CAACT pool=4
True Signal: fp=AGAAT pool=10
True Signal: fp=AATTT pool=8
True Signal: fp=TACCC pool=5
True Signal: fp=ACGAT pool=1
True Signal: fp=CGAAT pool=6
True Signal: fp=TAAGG pool=1
True Signal: fp=AAGGT pool=9
True Signal: fp=AAGGT pool=12
True Signal: fp=GCTGA pool=12
True Signal: fp=TGCAG pool=5
True Signal: fp=TAGCG pool=5
True Signal: fp=GCGAT pool=14
True Signal: fp=GCTGC pool=10
True Signal: fp=GCTGG pool=1
True Signal: fp=GGTCG pool=0
True Signal: fp=TCAAT pool=4
True Signal: fp=TAAGT pool=2
True Signal: fp=CCTGT pool=5
True Signal: fp=TCTCG pool=12
True Signal: fp=TGTGA pool=9
True Signal: fp=GCTGT pool=2
True Signal: fp=GGTCT pool=13
True Signal: fp=CAATA pool=7
True Signal: fp=GAATA pool=0
True Signal: fp=GAATA pool=15
True Signal: fp=ATTTA pool=1
True Signal: fp=ATTTA pool=12
10mers:18240
11mers:2483
12mers:581
13mers:357
14mers:335
15mers:325
16mers:321
17mers:322
18mers:319
19mers:317
20mers:315
21mers:313
22mers:313
23mers:310
24mers:310
25mers:310
26mers:307
27mers:305
28mers:304
29mers:302
30mers:302
31mers:301
32mers:298
33mers:297
34mers:296
35mers:295
36mers:294
37mers:293
38mers:292
39mers:292
40mers:291
41mers:290
42mers:289
43mers:288
44mers:287
45mers:288
46mers:285
47mers:283
48mers:282
49mers:281
50mers:281
51mers:279
52mers:278
53mers:277
54mers:276
55mers:275
```

```
56mers:275
57mers:275
58mers:273
59mers:271
60mers:271
61mers:271
62mers:271
63mers:268
64mers:267
65mers:268
66mers:265
67mers:264
68mers:262
69mers:261
70mers:260
71mers:259
72mers:258
73mers:257
74mers:254
75mers:253
76mers:252
77mers:252
78mers:250
79mers:250
80mers:249
81mers:247
82mers:246
83mers:245
84mers:245
85mers:244
86mers:241
87mers:240
88mers:239
89mers:238
90mers:239
91mers:239
92mers:237
93mers:234
94mers:233
95mers:232
96mers:230
97mers:229
98mers:228
99mers:228
100mers:227
101mers:225
102mers:224
103mers:225
```

```
104mers:222
105mers:222
106mers:222
107mers:220
108mers:218
109mers:217
110mers:217
111mers:216
112mers:215
113mers:214
114mers:211
115mers:211
116mers:209
117mers:208
118mers:209
119mers:207
120mers:206
121mers:203
122mers:201
123mers:200
124mers:199
125mers:199
126mers:197
127mers:196
128mers:195
129mers:195
130mers:193
131mers:192
132mers:192
133mers:191
134mers:188
135mers:187
136mers:186
137mers:185
138mers:184
139mers:183
140mers:182
141mers:181
142mers:180
143mers:179
144mers:179
145mers:178
146mers:177
147mers:176
148mers:174
149mers:173
150mers:173
151mers:171
```

```
152mers:170
153mers:169
154mers:168
155mers:167
156mers:166
157mers:166
158mers:165
159mers:163
160mers:161
161mers:160
162mers:159
163mers:158
164mers:157
165mers:158
166mers:157
167mers:154
168mers:153
169mers:153
170mers:152
171mers:150
172mers:150
173mers:150
174mers:149
175mers:147
176mers:147
177mers:144
178mers:144
179mers:142
180mers:142
181mers:140
182mers:139
183mers:138
184mers:137
185mers:137
186mers:135
187mers:134
188mers:133
189mers:131
190mers:130
191mers:132
192mers:130
193mers:128
194mers:126
195mers:125
196mers:124
197mers:124
198mers:122
199mers:122
```

```
200mers:120
201mers:119
202mers:120
203mers:120
204mers:117
205mers:115
206mers:115
207mers:113
208mers:113
209mers:110
210mers:109
211mers:108
212mers:107
213mers:106
214mers:106
215mers:105
216mers:103
217mers:103
218mers:102
219mers:102
220mers:103
221mers:99
222mers:96
223mers:96
224mers:95
225mers:94
226mers:92
227mers:91
228mers:90
229mers:89
230mers:87
231mers:86
232mers:86
233mers:84
234mers:81
235mers:79
236mers:78
237mers:77
238mers:77
239mers:78
240mers:75
241mers:72
242mers:70
243mers:69
244mers:68
245mers:67
246mers:66
247mers:65
```

```
248mers:64
249mers:64
250mers:64
251mers:62
252mers:60
253mers:60
254mers:60
255mers:57
256mers:56
257mers:55
258mers:55
259mers:53
260mers:51
261mers:50
262mers:50
263mers:48
264mers:48
265mers:49
266mers:48
267mers:44
268mers:43
269mers:43
270mers:42
271mers:41
272mers:38
273mers:38
274mers:36
275mers:34
276mers:33
277mers:33
278mers:32
279mers:30
280mers:28
281mers:25
282mers:24
283mers:24
284mers:23
285mers:22
286mers:19
287mers:17
288mers:16
289mers:15
290mers:15
291mers:13
292mers:11
293mers:9
294mers:8
295mers:7
```

```
296mers:6
297mers:5
298mers:4
299mers:3
300mers:1

GTAGGGGTAG ACATCGCGTA AAAGGGGCGT ACCCAGGACC CCCCTTGGCT CAATAAGTAG
CGCTGGGGTG CTACTACGGG TCTCGACACG CATTCAACTA AAAGCTTCCA TTCGCACGGG
CTTATTTAAC GAAGGTCGCG ATAAGGTGCC GAATAGGCTG CAGAGCGGCA GCCTGTCCAG
TGAATGCTGT GAGGCCTCCA GCTGACTCAT GAGAGAAGCC CAGTATTCAA ACTACGATTC
CACTCGACAA TTTAGGATGT CTTCCCGAAA GCTATCGGGT AGAATATCAG ATTCGTTTAA

True solution  DotsOn=285

Solutions: 1
```

r300.100.15.DN16.out

```
Using pool DN16
Using sequence r300

True Signal: fp=CTCGA pool=7
True Signal: fp=CTACG pool=1
True Signal: fp=CTACG pool=2
True Signal: fp=GTACC pool=0
True Signal: fp=ATCGC pool=1
True Signal: fp=GAATG pool=15
True Signal: fp=ATCGG pool=13
True Signal: fp=GTCGC pool=13
True Signal: fp=ACCCA pool=14
True Signal: fp=CTGGG pool=10
True Signal: fp=CAATT pool=3
True Signal: fp=GACAA pool=1
True Signal: fp=TACTA pool=3
True Signal: fp=ACCCC pool=6
True Signal: fp=AGACA pool=10
True Signal: fp=TTCCA pool=8
True Signal: fp=TTCCA pool=4
True Signal: fp=ACGCA pool=8
True Signal: fp=GACAC pool=2
True Signal: fp=CGACA pool=10
True Signal: fp=CGACA pool=11
True Signal: fp=CTACT pool=10
True Signal: fp=CCCCC pool=2
True Signal: fp=CCCCC pool=14
True Signal: fp=TTCCC pool=12
True Signal: fp=GCCCA pool=1
True Signal: fp=GAGAA pool=8
True Signal: fp=CCAGC pool=5
True Signal: fp=CAGAG pool=3
True Signal: fp=GCAGA pool=1
True Signal: fp=GCAGC pool=12
True Signal: fp=CGCGA pool=3
True Signal: fp=AGCGC pool=0
True Signal: fp=GGACC pool=1
True Signal: fp=CCAGG pool=7
True Signal: fp=TTAGG pool=1
True Signal: fp=GAGAG pool=6
True Signal: fp=TAAAA pool=11
True Signal: fp=AGCGG pool=4
True Signal: fp=ACTAA pool=15
True Signal: fp=CGGGC pool=4
True Signal: fp=ACTAC pool=4
True Signal: fp=ACTAC pool=7
True Signal: fp=AGGGG pool=9
True Signal: fp=AGGGG pool=5
True Signal: fp=TTTAA pool=15
True Signal: fp=GGGGC pool=7
True Signal: fp=CAGAT pool=11
True Signal: fp=CATGA pool=14
True Signal: fp=AATGC pool=1
True Signal: fp=CCCCT pool=13
True Signal: fp=GACAT pool=4
True Signal: fp=TCTTC pool=8
True Signal: fp=CCAGT pool=10
True Signal: fp=CCAGT pool=9
True Signal: fp=GCTAC pool=9
True Signal: fp=TTTAG pool=11
True Signal: fp=TGAGA pool=12
True Signal: fp=TGCCG pool=8
True Signal: fp=GCGCT pool=15
True Signal: fp=CGCGT pool=4
True Signal: fp=TGAGG pool=5
True Signal: fp=TCGGG pool=1
True Signal: fp=CGGGT pool=8
True Signal: fp=CGGGT pool=12
True Signal: fp=GGCGT pool=12
True Signal: fp=TATCA pool=4
True Signal: fp=ATATC pool=9
True Signal: fp=CTATC pool=6
True Signal: fp=GGGGT pool=11
True Signal: fp=GGGGT pool=14
True Signal: fp=TATCG pool=3
True Signal: fp=GCTAT pool=3
True Signal: fp=GATGT pool=0
True Signal: fp=TGGCT pool=6
True Signal: fp=CTCAA pool=15
True Signal: fp=ATCAG pool=6
True Signal: fp=CGATA pool=2
True Signal: fp=CTGAC pool=5
True Signal: fp=GTATT pool=11
True Signal: fp=ATGAG pool=8
True Signal: fp=GCCTC pool=11
True Signal: fp=GTGAA pool=2
True Signal: fp=GCGTA pool=0
True Signal: fp=GCGTA pool=9
True Signal: fp=GCCTG pool=12
True Signal: fp=GGATG pool=1
True Signal: fp=GTGAG pool=0
True Signal: fp=TTAAC pool=6
True Signal: fp=AAAGC pool=1
True Signal: fp=AAAGC pool=6
True Signal: fp=AAGCC pool=8
True Signal: fp=CTCAT pool=8
True Signal: fp=AGATT pool=12
True Signal: fp=CAGCC pool=10
True Signal: fp=CGCAC pool=3
True Signal: fp=AAAGG pool=1
True Signal: fp=GACCC pool=9
True Signal: fp=CCCTT pool=1
True Signal: fp=CGATT pool=11
True Signal: fp=GAAGC pool=5
True Signal: fp=TCATG pool=1
True Signal: fp=AGGAC pool=6
True Signal: fp=TGCTA pool=4
True Signal: fp=GAAGG pool=10
True Signal: fp=AATAA pool=2
True Signal: fp=TGCTG pool=9
True Signal: fp=GGCAG pool=1
True Signal: fp=GAGCG pool=3
True Signal: fp=CTTGG pool=1
True Signal: fp=ACAAT pool=6
True Signal: fp=ACTCA pool=7
True Signal: fp=TCCAC pool=10
True Signal: fp=AATAG pool=13
True Signal: fp=GATAA pool=1
True Signal: fp=TACGA pool=6
True Signal: fp=TATTC pool=2
True Signal: fp=CCTCC pool=3
True Signal: fp=TAACG pool=14
True Signal: fp=AAGCT pool=12
True Signal: fp=AAGCT pool=5
True Signal: fp=ACTCG pool=15
True Signal: fp=CAGCT pool=9
True Signal: fp=TCCAG pool=8
True Signal: fp=CGCAT pool=11
True Signal: fp=TCGAC pool=9
True Signal: fp=TCGAC pool=5
True Signal: fp=GCTCA pool=5
True Signal: fp=AGGAT pool=8
True Signal: fp=TAGGA pool=15
True Signal: fp=AGTGA pool=14
True Signal: fp=TAGGC pool=13
True Signal: fp=TACGG pool=7
True Signal: fp=TAGGG pool=13
True Signal: fp=AATAT pool=13
True Signal: fp=GGTGC pool=1
True Signal: fp=GGTGC pool=5
True Signal: fp=TCCAT pool=9
True Signal: fp=TGAAT pool=10
True Signal: fp=TATTT pool=6
True Signal: fp=TGTCC pool=10
True Signal: fp=AACTA pool=1
True Signal: fp=AACTA pool=3
True Signal: fp=CACTC pool=7
True Signal: fp=CTCCA pool=6
True Signal: fp=AAGTA pool=7
True Signal: fp=CAGTA pool=8
True Signal: fp=GACTC pool=14
True Signal: fp=GTCCA pool=3
True Signal: fp=CTGCA pool=11
True Signal: fp=ATAGG pool=14
True Signal: fp=GTAGA pool=8
True Signal: fp=GTAGA pool=9
True Signal: fp=TGTCT pool=0
True Signal: fp=CAGTG pool=15
True Signal: fp=GTAGC pool=14
True Signal: fp=GTGCC pool=10
True Signal: fp=CAAAC pool=11
True Signal: fp=GTAGG pool=3
True Signal: fp=AAAAG pool=0
True Signal: fp=AAAAG pool=2
True Signal: fp=ACACG pool=5
True Signal: fp=GAAAG pool=14
True Signal: fp=CCCGA pool=15
True Signal: fp=AGCCC pool=10
True Signal: fp=AGAGA pool=13
True Signal: fp=ATGCT pool=6
True Signal: fp=AGAGC pool=14
True Signal: fp=GCTTA pool=9
True Signal: fp=AGGCC pool=12
True Signal: fp=CGGCA pool=10
True Signal: fp=GCCGA pool=7
True Signal: fp=CCTTG pool=2
True Signal: fp=GCTTC pool=5
True Signal: fp=TTCGC pool=10
True Signal: fp=GCACG pool=10
True Signal: fp=TTGGC pool=12
True Signal: fp=GTGCT pool=9
True Signal: fp=ACGGG pool=0
True Signal: fp=ACGGG pool=3
True Signal: fp=GCGGC pool=11
True Signal: fp=TAGAA pool=2
True Signal: fp=CCACT pool=13
True Signal: fp=GGGCG pool=2
True Signal: fp=TCAGA pool=9
True Signal: fp=CGTAA pool=12
True Signal: fp=TAGAC pool=11
True Signal: fp=CTTAT pool=13
True Signal: fp=AGCCT pool=0
True Signal: fp=CGTAC pool=7
True Signal: fp=CATCG pool=7
True Signal: fp=TCGCA pool=7
True Signal: fp=TCCCG pool=1
True Signal: fp=AGTAG pool=9
True Signal: fp=AGGCT pool=10
True Signal: fp=GGCCT pool=8
True Signal: fp=TCGCG pool=5
True Signal: fp=GGTAG pool=10
True Signal: fp=GGTAG pool=3
True Signal: fp=GGGCT pool=8
True Signal: fp=TGGGG pool=1
True Signal: fp=AGTAT pool=0
True Signal: fp=ATGTC pool=9
True Signal: fp=TGACT pool=9
True Signal: fp=CTGTC pool=11
True Signal: fp=GTCTC pool=4
True Signal: fp=CTGTG pool=3
True Signal: fp=CTAAA pool=14
True Signal: fp=ACATC pool=13
True Signal: fp=GTAAA pool=13
True Signal: fp=ATAAG pool=13
True Signal: fp=AGCTA pool=4
True Signal: fp=GTCTT pool=13
True Signal: fp=AGCTG pool=4
True Signal: fp=AGGTC pool=1
True Signal: fp=CGCTG pool=12
True Signal: fp=GGCTC pool=14
True Signal: fp=AGGTG pool=8
True Signal: fp=GGGTA pool=10
True Signal: fp=GGGTA pool=15
True Signal: fp=GGCTG pool=2
True Signal: fp=GGGTC pool=10
True Signal: fp=CGAAA pool=3
True Signal: fp=ATTCA pool=13
True Signal: fp=ATTCA pool=6
True Signal: fp=TTCAA pool=9
True Signal: fp=TTCAA pool=12
True Signal: fp=AACGA pool=11
True Signal: fp=ACGAA pool=13
True Signal: fp=ATTCC pool=2
True Signal: fp=CCGAA pool=12
True Signal: fp=CCGAA pool=14
True Signal: fp=CATTC pool=13
True Signal: fp=CCATT pool=11
True Signal: fp=GGGTG pool=6
True Signal: fp=AGAAG pool=0
True Signal: fp=CCCAG pool=3
True Signal: fp=CCCAG pool=5
True Signal: fp=CACGC pool=10
True Signal: fp=CTTCC pool=14
True Signal: fp=CTTCC pool=6
True Signal: fp=TTATT pool=0
True Signal: fp=GATTC pool=12
True Signal: fp=GATTC pool=14
True Signal: fp=CAGGA pool=6
True Signal: fp=GCATT pool=15
True Signal: fp=AGCTT pool=4
True Signal: fp=ATTCG pool=9
True Signal: fp=ATTCG pool=5
True Signal: fp=CGAAG pool=14
True Signal: fp=CACGG pool=9
True Signal: fp=AAGGG pool=13
True Signal: fp=GAGGC pool=11
True Signal: fp=GGCTT pool=11
True Signal: fp=AAACT pool=4
True Signal: fp=TCAAA pool=4
True Signal: fp=TCAAC pool=5
True Signal: fp=CAACT pool=4
True Signal: fp=AGAAT pool=10
True Signal: fp=AATTT pool=8
True Signal: fp=TACCC pool=5
True Signal: fp=ACGAT pool=1
True Signal: fp=CGAAT pool=6
True Signal: fp=TAAGG pool=1
True Signal: fp=AAGGT pool=9
True Signal: fp=AAGGT pool=12
True Signal: fp=GCTGA pool=12
True Signal: fp=TGCAG pool=5
True Signal: fp=TAGCG pool=5
True Signal: fp=GCGAT pool=14
True Signal: fp=GCTGC pool=10
True Signal: fp=GCTGG pool=1
True Signal: fp=GGTCG pool=0
True Signal: fp=TCAAT pool=4
True Signal: fp=TAAGT pool=2
True Signal: fp=CCTGT pool=5
True Signal: fp=TCTCG pool=12
True Signal: fp=TGTGA pool=9
True Signal: fp=GCTGT pool=2
True Signal: fp=GGTCT pool=13
True Signal: fp=CAATA pool=7
True Signal: fp=GAATA pool=0
True Signal: fp=GAATA pool=15
True Signal: fp=ATTTA pool=1
True Signal: fp=ATTTA pool=12
False positive Signal: fp=AGACT pool=2
False positive Signal: fp=AACTG pool=12
False positive Signal: fp=CCACA pool=11
False positive Signal: fp=GCCGC pool=7
False positive Signal: fp=CATAC pool=2
False positive Signal: fp=GTGTA pool=0
False positive Signal: fp=AAGAG pool=9
False positive Signal: fp=GATGT pool=7
False positive Signal: fp=CAAGC pool=6
False positive Signal: fp=GGGAC pool=3
False positive Signal: fp=ATTTC pool=9
False positive Signal: fp=GATTA pool=1
False positive Signal: fp=TCCCT pool=10
False positive Signal: fp=GGTAC pool=11
False positive Signal: fp=GCAGC pool=9
False positive Signal: fp=CCGCT pool=4
False positive Signal: fp=CATTT pool=3
False positive Signal: fp=ACTGA pool=15
False positive Signal: fp=AGAGC pool=2
False positive Signal: fp=GTCCA pool=10
False positive Signal: fp=TGAGA pool=2
False positive Signal: fp=GAATC pool=10
False positive Signal: fp=ATCTC pool=1
False positive Signal: fp=CACCC pool=5
False positive Signal: fp=CTGGT pool=10
False positive Signal: fp=CGGCT pool=7
False positive Signal: fp=CAAGT pool=3
False positive Signal: fp=TAGAT pool=2
False positive Signal: fp=AGGCG pool=2
False positive Signal: fp=GTCTA pool=11
False positive Signal: fp=CAATA pool=1
False positive Signal: fp=GTAGG pool=8
False positive Signal: fp=GTGAC pool=2
False positive Signal: fp=GATGC pool=4
False positive Signal: fp=GACGC pool=2
False positive Signal: fp=AGCCA pool=12
False positive Signal: fp=GCAGC pool=7
False positive Signal: fp=GGTGA pool=7
False positive Signal: fp=TATCT pool=6
False positive Signal: fp=CATAT pool=15
False positive Signal: fp=AGATC pool=7
False positive Signal: fp=TATAG pool=14
False positive Signal: fp=TCAAA pool=0
False positive Signal: fp=ACTCA pool=10
False positive Signal: fp=GACAA pool=3
False positive Signal: fp=GTCTA pool=9
False positive Signal: fp=ACTCC pool=1
False positive Signal: fp=CGGAG pool=6
False positive Signal: fp=CCTAA pool=8
False positive Signal: fp=GTCCG pool=13
False positive Signal: fp=CGACA pool=15
False positive Signal: fp=CCTGA pool=10
False positive Signal: fp=CCATT pool=9
False positive Signal: fp=ACTAT pool=4
False positive Signal: fp=AACCG pool=9
False positive Signal: fp=CGATC pool=11
False positive Signal: fp=TGGAG pool=3
False positive Signal: fp=AGCCC pool=0
False positive Signal: fp=ATCTC pool=10
False positive Signal: fp=CATTA pool=6
False positive Signal: fp=GCTGG pool=12
False positive Signal: fp=GTGCA pool=13
False positive Signal: fp=CACTC pool=10
False positive Signal: fp=AACAT pool=14
False positive Signal: fp=GCCAC pool=7
False positive Signal: fp=AAGAC pool=3
False positive Signal: fp=CGTGG pool=12
False positive Signal: fp=CGTTT pool=0
False positive Signal: fp=CTCGC pool=13
False positive Signal: fp=GGAAA pool=9
False positive Signal: fp=GGTCC pool=15
False positive Signal: fp=TCTGA pool=15
False positive Signal: fp=TCAAC pool=15
False positive Signal: fp=AAGCA pool=9
False positive Signal: fp=GGAAG pool=1
False positive Signal: fp=GTGGG pool=1
False positive Signal: fp=TAAGC pool=9
False positive Signal: fp=TGGGA pool=10
False positive Signal: fp=GTTTA pool=2
False positive Signal: fp=GGGCG pool=12
False positive Signal: fp=ACAGG pool=0
False positive Signal: fp=ACATC pool=9
False positive Signal: fp=CAATG pool=3
False positive Signal: fp=AAAGC pool=9
False positive Signal: fp=GGAAC pool=5
False positive Signal: fp=GGGGA pool=0
False positive Signal: fp=CTGGT pool=13
False positive Signal: fp=GGGTA pool=15
False positive Signal: fp=ATCTC pool=9
False positive Signal: fp=GTCAC pool=15
False positive Signal: fp=AAGTT pool=7
False positive Signal: fp=CCATG pool=8
False positive Signal: fp=TAAGG pool=15
False positive Signal: fp=AAAGC pool=6
False positive Signal: fp=CCGGT pool=3
False positive Signal: fp=ACAAA pool=13
False positive Signal: fp=TCTTT pool=14
False positive Signal: fp=CTGTA pool=6
False positive Signal: fp=CAGTG pool=15
False positive Signal: fp=CCCAG pool=0
False negative : fp= pool=
False negative : fp=CTCGA pool=7
False negative : fp=CTACG pool=1
False negative : fp=CTACG pool=2
False negative : fp=GTACC pool=0
False negative : fp=ATCGC pool=1
False negative : fp=GAATG pool=15
False negative : fp=ATCGG pool=13
False negative : fp=GTCGC pool=13
False negative : fp=ACCCA pool=14
False negative : fp=CTGGG pool=10
False negative : fp=CAATT pool=3
False negative : fp=GACAA pool=1
False negative : fp=TACTA pool=3
False negative : fp=ACCCC pool=6
10mers:23552
11mers:20332
12mers:15187
13mers:10500
14mers:8165
15mers:6357
16mers:5426
17mers:4711
18mers:4327
19mers:4105
20mers:4006
21mers:3949
22mers:3895
23mers:3800
24mers:3721
25mers:3650
26mers:3611
27mers:3627
28mers:3613
29mers:3613
30mers:3605
31mers:3596
32mers:3619
33mers:3656
34mers:3673
35mers:3700
36mers:3714
```

```
37mers:3768
38mers:3822
39mers:3838
40mers:3845
41mers:3856
42mers:3920
43mers:3982
44mers:4015
45mers:4080
46mers:4132
47mers:4109
48mers:4126
49mers:4098
50mers:4084
51mers:4096
52mers:4131
53mers:4180
54mers:4257
55mers:4320
56mers:4384
57mers:4486
58mers:4532
59mers:4565
60mers:4567
61mers:4624
62mers:4729
63mers:4873
64mers:4994
65mers:5081
66mers:5141
67mers:5169
68mers:5191
69mers:5220
70mers:5299
71mers:5427
72mers:5558
73mers:5648
74mers:5674
75mers:5691
76mers:5716
77mers:5777
78mers:5833
79mers:5865
80mers:5893
81mers:5968
82mers:6075
83mers:6198
84mers:6331
```

```
85mers:6394
86mers:6470
87mers:6535
88mers:6606
89mers:6668
90mers:6721
91mers:6778
92mers:6842
93mers:6891
94mers:6895
95mers:6881
96mers:6901
97mers:6920
98mers:6925
99mers:6908
100mers:6883
101mers:4871
102mers:4792
103mers:4761
104mers:4729
105mers:4714
106mers:4751
107mers:4810
108mers:4879
109mers:4878
110mers:4811
111mers:4738
112mers:4684
113mers:4614
114mers:4555
115mers:4502
116mers:4475
117mers:4448
118mers:4402
119mers:4399
120mers:4435
121mers:4439
122mers:4449
123mers:4453
124mers:4419
125mers:4380
126mers:4363
127mers:4304
128mers:4243
129mers:4166
130mers:4087
131mers:4068
132mers:4041
```

```
133mers:4003
134mers:3959
135mers:3906
136mers:3859
137mers:3802
138mers:3743
139mers:3713
140mers:3616
141mers:3577
142mers:3589
143mers:3572
144mers:3618
145mers:3668
146mers:3697
147mers:3670
148mers:3639
149mers:3580
150mers:3503
151mers:3431
152mers:3384
153mers:3359
154mers:3330
155mers:3321
156mers:3288
157mers:3313
158mers:3325
159mers:3313
160mers:3273
161mers:3251
162mers:3212
163mers:3196
164mers:3185
165mers:3179
166mers:3182
167mers:3129
168mers:3091
169mers:3048
170mers:3080
171mers:3069
172mers:3061
173mers:3036
174mers:3012
175mers:2970
176mers:2911
177mers:2912
178mers:2891
179mers:2925
180mers:2945
```

```
181mers:2992
182mers:3019
183mers:3002
184mers:2973
185mers:2965
186mers:2973
187mers:2981
188mers:2955
189mers:2899
190mers:2836
191mers:2756
192mers:2707
193mers:2673
194mers:2646
195mers:2628
196mers:2618
197mers:2591
198mers:2580
199mers:2596
200mers:2623
201mers:2623
202mers:2595
203mers:2583
204mers:2529
205mers:2505
206mers:2524
207mers:2527
208mers:2555
209mers:2523
210mers:2487
211mers:2431
212mers:2364
213mers:2307
214mers:2263
215mers:2227
216mers:2168
217mers:2123
218mers:2077
219mers:2065
220mers:2035
221mers:2020
222mers:2034
223mers:2038
224mers:2026
225mers:2000
226mers:1975
227mers:1943
228mers:1879
```

```
229mers:1808
230mers:1771
231mers:1720
232mers:1687
233mers:1620
234mers:1548
235mers:1492
236mers:1453
237mers:1405
238mers:1381
239mers:1338
240mers:1272
241mers:1222
242mers:1190
243mers:1171
244mers:1129
245mers:1104
246mers:1095
247mers:1066
248mers:1021
249mers:996
250mers:939
251mers:896
252mers:850
253mers:795
254mers:742
255mers:679
256mers:649
257mers:631
258mers:613
259mers:602
260mers:605
261mers:600
262mers:585
263mers:568
264mers:540
265mers:509
266mers:487
267mers:472
268mers:451
269mers:418
270mers:395
271mers:365
272mers:337
273mers:319
274mers:285
275mers:266
276mers:246
```

```
277mers:223
278mers:203
279mers:194
280mers:183
281mers:173
282mers:173
283mers:161
284mers:145
285mers:136
286mers:135
287mers:130
288mers:123
289mers:121
290mers:105
291mers:91
292mers:84
293mers:66
294mers:53
295mers:41
296mers:31
297mers:26
298mers:21
299mers:16
300mers:10
GTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCTG
GGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAAC
GAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGGC
CTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATGT
CTTCCCGAAAGCTATCGGGTAGAATATCAGATTCGTTTAA
  True solution  DotsOn=285

GTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCTG
GGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAAC
GAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGGC
CTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATGT
CTTCCCGAAAGCTATCGGGTAGAATATCAGATTCCCATGT
  DotsOn=283

GGTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCT
GGGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAA
CGAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGG
CCTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATG
TCTTCCCGAAAGCTATCGGGTAGAATATCAGATTCGTTTT
  DotsOn=285

GTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCTG
GGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAAC
GAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGGC
```

```
CTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATGT
CTTCCCGAAAGCTATCGGGTAGAATATCAGATTCGTTTTG
   DotsOn=285

GGTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCT
GGGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAA
CGAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGG
CCTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATG
TCTTCCCGAAAGCTATCGGGTAGAATATCAGATTCCCATG
   DotsOn=284

GGGTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGC
TGGGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTA
ACGAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAG
GCCTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGAT
GTCTTCCCGAAAGCTATCGGGTAGAATATCAGATTCGTTT
   DotsOn=284

GGTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCT
GGGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAA
CGAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGG
CCTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATG
TCTTCCCGAAAGCTATCGGGTAGAATATCAGATTCGTTTG
   DotsOn=285

GGGTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGC
TGGGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTA
ACGAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAG
GCCTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGAT
GTCTTCCCGAAAGCTATCGGGTAGAATATCAGATTCCCAT
   DotsOn=284

GGTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCT
GGGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAA
CGAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGG
CCTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATG
TCTTCCCGAAAGCTATCGGGTAGAATATCAGATTCGTTTA
   DotsOn=285

GTAGGGGTAGACATCGCGTAAAAGGGGCGTACCCAGGACCCCCCTTGGCTCAATAAGTAGCGCTG
GGGTGCTACTACGGGTCTCGACACGCATTCAACTAAAAGCTTCCATTCGCACGGGCTTATTTAAC
GAAGGTCGCGATAAGGTGCCGAATAGGCTGCAGAGCGGCAGCCTGTCCAGTGAATGCTGTGAGGC
CTCCAGCTGACTCATGAGAGAAGCCCAGTATTCAAACTACGATTCCACTCGACAATTTAGGATGT
CTTCCCGAAAGCTATCGGGTAGAATATCAGATTCGTTTGA
   DotsOn=284

Solutions: 10
```

WHAT IS CLAIMED IS:

1. A method of identifying one or more sequences of a target nucleic acid comprising:

a. contacting a target nucleic acid with a first set of pools of probes, wherein at least one pool in the set comprises a mixture of two or more probes having different sequences in information regions of the probes, under conditions which produce, on average, more probe:target hybridization with probes which are perfectly complementary to the target nucleic acid in the information region of the probes than with probes which are mismatched in the information regions;

b. detecting a first subset of pools for which a level of hybridization indicates that there is at least one perfectly complementary probe within each pool; and

c. identifying one or more sequences of the target nucleic acid from the first subset of pools detected in step (b) by compiling overlapping sequences of the information regions of the probes in the subset of detected pools, wherein one or more pooling false positive probes are eliminated as a result of compilation of overlapping sequences.

2. A method of identifying one or more sequences of a target nucleic acid comprising:

a. contacting a target nucleic acid with a first set of pools of probes, wherein at least one pool in the set comprises a mixture of two or more probes having different sequences in information regions of the probes, under conditions which produce, on average, more probe:target hybridization with probes which are perfectly complementary to the target nucleic acid in the information region of the probes than with probes which are mismatched in the information regions;

b. assigning a hybridization score to each probe wherein each probe within a pool is assigned the same hybridization score, and

c. identifying one or more sequences of the target nucleic acid by analysis of hybridization scores of overlapping probes, wherein one or more probes with false high scores arising from pooling of probes are eliminated by analysis of hybridization scores of overlapping probes.

3. The method of claim 2 wherein a statistical analysis of hybridization scores is performed in step (c).

4. The method of claim 3 wherein step (c) further comprises calculating a score for the identified one or more sequences of the target nucleic acid.

5. The method of claim 1 further comprising, following step (b) and before step (c), the steps of:

a. contacting the target nucleic acid with a second set of pools of probes containing at least one probe having the same information region as a probe in the first set,

b. detecting a second subset of pools for which the level of hybridization indicates that there is at least one perfectly complementary probe within each pool; and

c. eliminating probes with the same information regions present in both the first set of pools of probes and the second set of pools of probes that are not present in both the first detected subset of pools and the second detected subset of pools.

6. The method of claim 5 wherein the first and second sets of pools of probes comprise the same information regions.

7. The method of claim 5 wherein the first and second sets of pools of probes comprise the same probes.

8. The method of claim 2 further comprising, after step (b) and before step (c), the steps of:

a. contacting the target nucleic acid with a second set of pools of probes containing at least one probe having the same information region as a probe in the first set,

b. assigning a hybridization score to each probe wherein each probe within a pool is assigned the same hybridization score.

9. The method of claim 8 further comprising the step of:

c. eliminating the higher of two scores for probes present in both the first set and second set of pools of probes.

10. The method of claim 8 wherein the first and second sets of pools of probes comprise the same information regions.

11. The method of claim 8 wherein the first and second sets of pools of probes comprise the same probes.

12. The method of claim 1 or 2 in which the target nucleic acid is labeled.

13. The method of claim 1 or 2 in which the probes are labeled.

14. The method of claim 1 or 2 in which the label is a fluorophore.

15. The method of claim 1 or 2 in which the label is attached to a terminal nucleotide.

16. The method of claim 1 or 2 in which the label is attached to an internal nucleotide.

17. The method of claim 1 or 2 in which the first set of pools of probes is immobilized on one or more solid supports.

18. The method of claim 17 in which the pools of probes are arranged in a spatially-addressable array in which each pool has a unique address.

19. The method of claim 1 or 2 in which the target nucleic acid is immobilized on one or more solid supports.

20. A method of identifying one or more sequences of a target nucleic acid comprising:

a. contacting a target nucleic acid with a first set of pools of immobilized probes and a first set of pools of labeled probes, wherein at least one pool in either the first set of pools of immobilized probes, or in the first set of pools of labeled probes, or in both, comprises a mixture of two or more probes having different sequences in the information regions of the probes, under conditions which produce, on average, more probe:target hybridization for probes which are perfectly complementary to the target nucleic acid in the information region than with probes which are mismatched in the information region;

b. covalently joining adjacently hybridized immobilized probes and labeled probes to provide a first set of covalently joined probes;

c. detecting a first subset of pools of covalently joined probes for which a level of hybridization indicates that there is at least one perfectly complementary covalently joined probe within each pool; and

d. identifying one or more sequences of the target nucleic acid from the first subset of covalently joined pools of probes detected in step (c) by compiling overlapping sequences of the information regions of covalently joined probes in the subset of detected pools, wherein one or more covalently joined pooling false positive probes are eliminated as a result of compilation of overlapping sequences.

21. The method of claim 20 further comprising, following step (c) and before step (d), the steps of:

a. contacting the target nucleic acid with a second set of pools of immobilized probes and a second set of pools of labeled probes, wherein at least one probe in the second set of immobilized probes has the same information region as a probe in the first set of pools of immobilized probes, or at least one probe in the second set of labeled probes has the same information region as a probe in the first set of pools of labeled probes,

b. covalently joining adjacently hybridized immobilized probes and labeled probes to provide a second set of covalently joined probes;

c. detecting a second subset of covalently joined pools of probes for which a level of hybridization indicates that there is at least one perfectly complementary probe within each pool; and

d. eliminating covalently joined probes with the same information regions present in both the first set of covalently joined pools of probes and the second set of covalently joined pools of probes that are not present in both the first detected subset of covalently joined pools of probes and the second detected subset of covalently joined pools of probes.

22. A method of identifying one or more sequences of a target nucleic acid comprising:

a. contacting a target nucleic acid with a first set of pools of immobilized probes and a first set of pools of labeled probes, wherein at least one pool in either the first set of pools of immobilized probes, or in the first set of pools of labeled probes, or in both, comprises a mixture of two or more probes having different sequences in the information regions of the probes, under conditions which produce, on average, more probe:target hybridization for probes which are perfectly complementary to the target nucleic acid in the information region than with probes which are mismatched in the information region;

b. covalently joining adjacently hybridized immobilized probes and labeled probes to provide a first set of covalently joined probes;

c. assigning a hybridization score to each covalently joined probe in the first set wherein each probe within a pool of covalently joined probes is assigned the same hybridization score, and

e. identifying one or more sequences of the target nucleic acid from overlapping covalently joined probes by analysis of hybridization scores of overlapping covalently joined probes wherein one or more covalently joined probes with false high scores arising from pooling of probes are eliminated by analysis of hybridization scores of overlapping probes.

23. The method of claim 22 further comprising after step (c) and before step (d) the steps of:

a. contacting the target nucleic acid with a second set of pools of immobilized probes and a second set of pools of labeled probes, wherein at least one probe in the second set of immobilized probes has the same information region as a probe in the first set of pools of immobilized probes, or at least one probe in the second set of labeled probes has the same information region as a probe in the first set of pools of labeled probes,

b. covalently joining adjacently hybridized immobilized probes and labeled probes to provide a second set of covalently joined probes;

c. assigning a hybridization score to each covalently joined probe of the second set wherein each probe within a pool of covalently joined probes is assigned the same hybridization score.

24. The method of claim 23 further comprising the step of

d. eliminating the higher of two scores for covalently joined probes present in both the first set and second set of covalently joined pools of probes.

25. The method of claim 21, 23 or 24 wherein the first and second sets of pools of immobilized probes, or the first and second sets of pools of labeled probes, or both, comprise the same information regions.

26. The method of claim 21, 23 or 24 wherein the first and second sets of pools of immobilized probes, or the first and second sets of pools of labeled probes, or both, comprise the same probes.

27. The method of any one of claims 20 through 24 in which a label of the labeled probe is a fluorophore.

28. The method of any one of claims 20 through 24 in which a label of the labeled probe is attached to a terminal nucleotide.

29. The method of any one of claims 20 through 24 in which a label of the labeled probe is attached to an internal nucleotide.

30. The method of any one of claims 20 through 24 in which the set of pools of immobilized probes is immobilized on one or more solid supports.

31. The method of claim 30 in which the sets of pools of immobilized probes are arranged in a spatially-addressable array in which each pool has a unique address.

32. The method of claim 22, 23 or 24 wherein a statistical analysis of hybridization scores is performed.

33. The method of claim 22 wherein step (d) further comprises calculating a score for the identified one or more sequences of the target nucleic acid.

34. The method of any one of claims 20 through 24 wherein the pools of immobilized probes each consist of one probe.

35. The method of any one of claims 20 through 24 wherein the pools of labeled probes each consist of one probe.

36. A set of pools of probes wherein each probe comprises an information region, wherein said set of probes is sufficient to determine the sequence of an unknown target nucleic acid by overlapping sequences of the information region of two or more of said probes, and wherein at least one pool comprises two or more probes having different sequences in the information regions and having the same label or no label, and wherein the set of the pools of probes also satisfies one or more of the following rules describing the information regions of the probes, said rules selected from the group consisting of:

a. a consensus sequence of at least one pool in the set consists only of the letters selected from the group consisting of V, H, D, B, and N;

b. a consensus sequence of probes in each pool in the set comprises more than three different letters selected from the group consisting of A, C, G, T, U, M, R, W, S, Y, K, V, H, D, B, and N;

c. consensus sequences from each informative position of all pools in the set comprise more than eight letters selected from the group consisting of A, C, G, T, U, M, R, W, S, Y, K, V, H, D, B, and N; and

d. consensus sequences from each information region of all pools in the set comprise more than five different letters selected from the group consisting of A, C, G, T, U, M, R, W, S, Y, K, V, H, D, B, and N, wherein at least one letter is selected from the group consisting of M, R, W, S, Y, and K.

37. The set of pools of probes of claim 36 wherein said set comprises all possible probes of the same length K, where K is greater than 3.

38. The set of pools of probes of claim 36 wherein each pool comprises more than 16 different probes.

39. The set of pools of probes of claim 38 wherein each pool comprises at least 32 different probes.

40. The set of pools of probes of claim 36 in which the pools are arranged in a spatially-addressable array, and wherein each pool has an address.

41. The set of pools of probes of claim 36 wherein at least two pools are mixed, wherein any two pools that are mixed are associated with different labels, and wherein all probes in a single pool are associated with the same label.

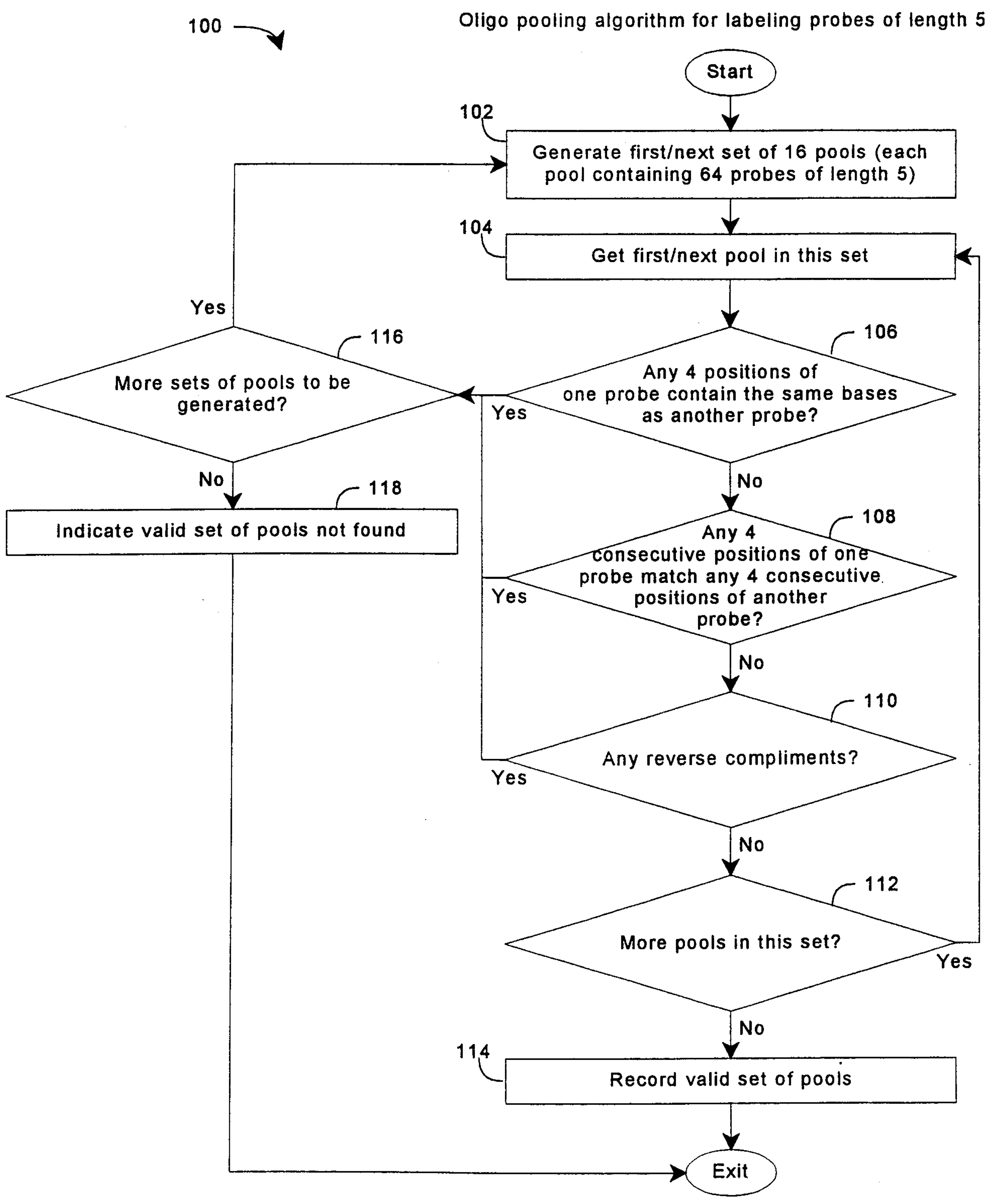
100
Oligo pooling algorithm for labeling probes of length 5
Start
102
Generate first/next set of 16 pools (each pool containing 64 probes of length 5)
104
Get first/next pool in this set
106
Any 4 positions of one probe contain the same bases as another probe?
Yes
No
108
Any 4 consecutive positions of one probe match any 4 consecutive positions of another probe?
Yes
No
110
Any reverse compliments?
Yes
No
112
More pools in this set?
Yes
No
114
Record valid set of pools
Exit
116
More sets of pools to be generated?
Yes
No
118
Indicate valid set of pools not found

FIG. 1

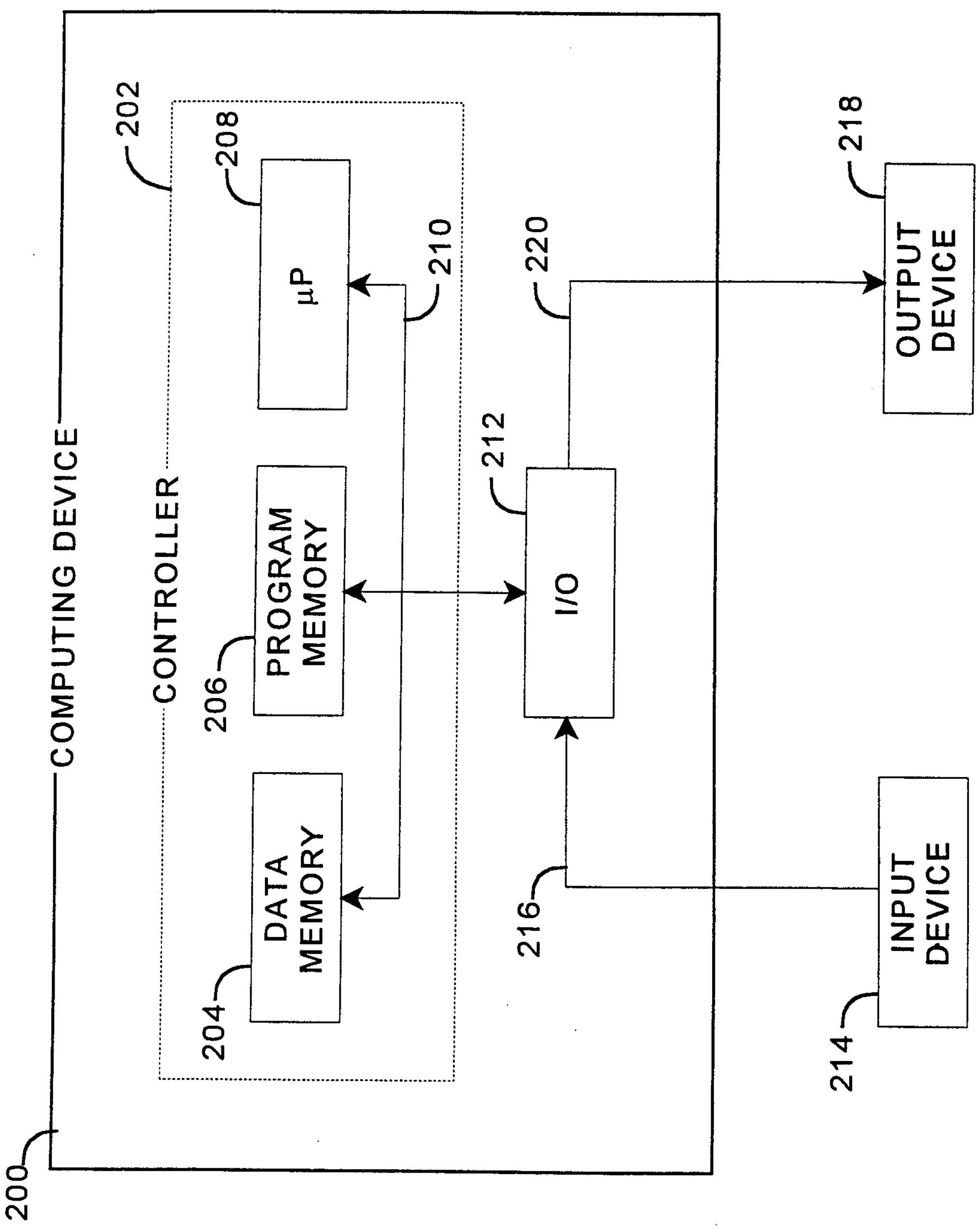
COMPUTING DEVICE
CONTROLLER
DATA MEMORY
PROGRAM MEMORY
μP
I/O
INPUT DEVICE
OUTPUT DEVICE
200
202
204
206
208
210
212
214
216
218
220

FIG. 2

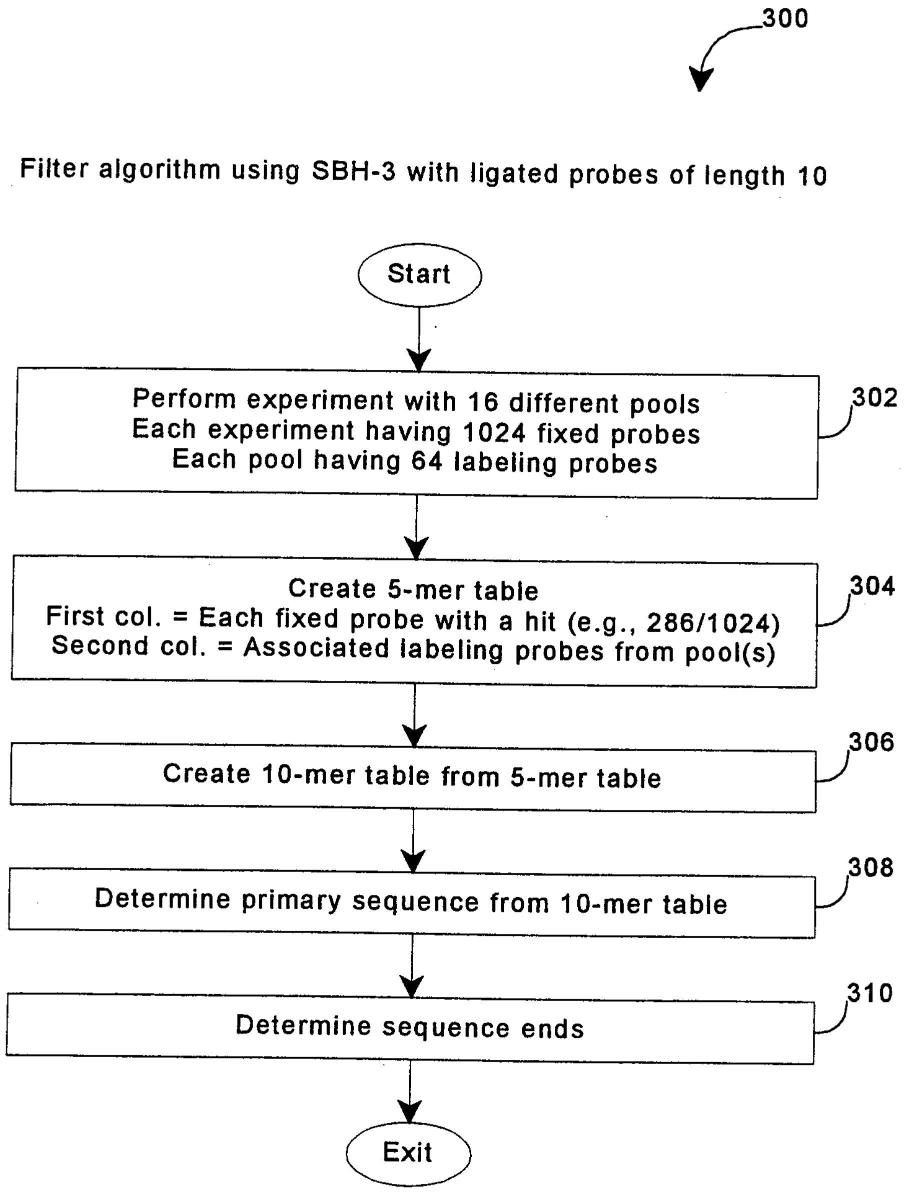
300
Filter algorithm using SBH-3 with ligated probes of length 10
Start
Perform experiment with 16 different pools
Each experiment having 1024 fixed probes
Each pool having 64 labeling probes
302
Create 5-mer table
First col. = Each fixed probe with a hit (e.g., 286/1024)
Second col. = Associated labeling probes from pool(s)
304
Create 10-mer table from 5-mer table
306
Determine primary sequence from 10-mer table
308
Determine sequence ends
310
Exit

FIG. 3

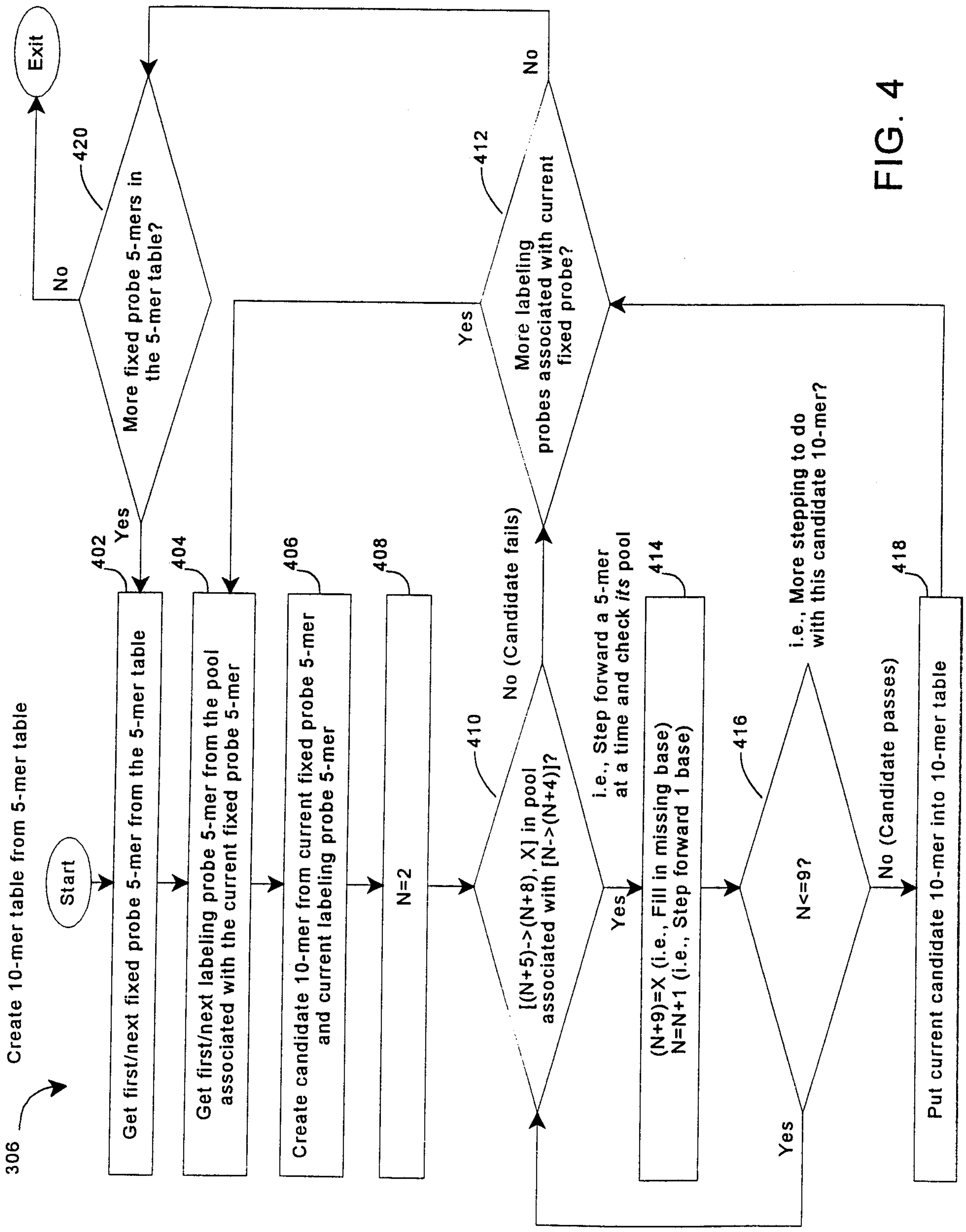
306
Create 10-mer table from 5-mer table
Start
402
Get first/next fixed probe 5-mer from the 5-mer table
404
Get first/next labeling probe 5-mer from the pool associated with the current fixed probe 5-mer
406
Create candidate 10-mer from current fixed probe 5-mer and current labeling probe 5-mer
408
N=2
410
[(N+5)->(N+8), X] in pool associated with [N->(N+4)]?
Yes
i.e., Step forward a 5-mer at a time and check its pool
No (Candidate fails)
414
(N+9)=X (i.e., Fill in missing base)
N=N+1 (i.e., Step forward 1 base)
416
N<=9?
i.e., More stepping to do with this candidate 10-mer?
Yes
No (Candidate passes)
418
Put current candidate 10-mer into 10-mer table
412
More labeling probes associated with current fixed probe?
Yes
No
420
More fixed probe 5-mers in the 5-mer table?
Yes
No
Exit

FIG. 4