(19) Octrooicentrum Nederland

(11) **2012222**

(12)C **OCTROOI**

(21) Aanvraagnummer: **2012222**

(22) Aanvraag ingediend: **06.02.2014**

(51) Int.Cl.:
G06F 17/30 (2006.01) ***G06F 19/22*** (2011.01)
H03M 7/30 (2006.01)

(43) Aanvraag gepubliceerd:
-

(47) Octrooi verleend:
10.08.2015

(45) Octrooischrift uitgegeven:
19.08.2015

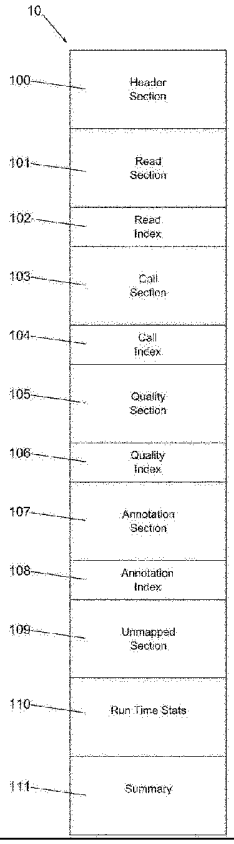
(73) Octrooihouder(s):
Genalice B.V. te Harderwijk.

(72) Uitvinder(s):
Johannes Karten te Harderwijk.

(74) Gemachtigde:
ir. J.H. de Hoog te Veenendaal.

(54) **A method of storing/reconstructing a multitude of sequences in/from a data storage structure.**

(57) The invention relates to a computer implemented method of storing/recovering in/from a storage data structure a multitude of sequences that have been aligned with a reference data structure. The information of the sequences is stored in different sections. Each section comprises data streams comprising specific data of the sequences having a reference position in the reference position range associated with the data stream. In a first section, the length of the sequences is stored. In a second section, the mutations of a sequence with respect to the reference sequence are stored. In a third section, consensus based quality values are linked with positions in the reference sequence. In a fourth section, the sequence identifiers are stored. The storage data structure has a format which is optimized for viewer, re-alignment, variant calling and other post-processing tools.



NL C 2012222

A method of storing/reconstructing a multitude of sequences in/from a data storage structure.

TECHNICAL FIELD

The invention relates to a computer implemented method of storing in a data storage structure a multitude of sequences that have been aligned with a reference data structure. The invention further relates to a computer implemented method of reconstructing from a data storage structure a sequence that have been aligned with a reference data structure.

BACKGROUND

Alignment is the process of mapping reads or pair-end reads on a reference based on the pattern of the read. A read is one sequence of elements wherein each element has one of the values A, C, G and T. A pair-end read comprises two sequences of elements, wherein one of the two sequences is a "plus strand" or "forward strand" and the other, the mate, is a "minus strand" or "reverse complement strand". The number of reads which stream from a sequencer varies depending on the read size. A high end sequencer can produce 120Gbase per day in short reads of 75, 100, 150 or 200 bases. The size of such stream is in the order of 300Gbytes. The data is stored in for example a FASTQ file.

Current alignment tools generate a SAM file. A SAM file is a tab-delimited text file that contains sequence alignment data. To reduce the size of the results, the SAM file is converted in a BAM file which is the binary version of a SAM file including an index.

In a BAM file, all the reads, which could be both mates of a pair-end read, are stored in the order of matching position. The order of position is essential to display with reasonable response times the alignment results with a viewer and to perform variant calling. However, to perform a re-alignment of the sequences stored in the BAM, it is desired that in case of pair-end read, both sequences are supplied simultaneous to the alignment process. Recovering the pair-end read from a BAM file requires a lot of time as the coupling between mates is 'soft'. To find the mate of a particular read, another part of the BAM-file has to

be read and decompressed to find the read with the corresponding sequence identifier.

SUMMARY

It is an object of the invention to provide an improved method of storing/recovering in/from a storage data structure a multitude of sequences that have been aligned with a reference data structure, which overcomes at least of the disadvantages described above.

The reference data structure describes reference data as one contiguous reference sequence wherein each element of the reference sequences has a position number and element value. A sequence comprises a number of elements with element values that matches a part of the reference sequence. The part of the reference sequence having a corresponding reference position.

According to a first aspect of the invention, this object is achieved by a method having the features of Claim 1. Advantageous embodiments and further ways of carrying out the invention may be attained by the measures mentioned in the dependent claims.

The improved method comprises: storing a first parameter in a header section of the data storage structure. The first parameter identifies the reference data structure. In a first storage section of the storage data structure for the multitude of sequences the reference positions and a multitude of first storage section records are stored. A first storage section record is associated with at least one sequence which has a corresponding reference position. The first storage section further comprises a length field with a value which enables to determine the number of elements of the at least one sequence.

The invention is based on the insight that about 95% of the reads that have to be aligned with a genome have a perfect match with the reference data sequence. A perfect match can be reconstructed from the reference data sequence if a start position on the reference data sequence is known and the length of the sequence is known. As the reference data sequence is always available in a reference data structure for further processing, only a linkage to the reference data structure is necessary in the data storage structure to enable retrieving element values from specific reference positions. By storing only the

reference position of a sequence and the length of the sequence/read the memory footprint is reduced significantly.

In an embodiment, the method further comprises counting the sequences having the same reference position to obtain a count value and generating a data stream by concatenating the count value and the first storage section records corresponding to the sequences that have the same reference position. To address a position in a genome with 3.2 bases, at least 32 bits are necessary. By grouping the sequences with the same reference position, the storage footprint to store all reference position can be reduced.

In a further embodiment, the position numbers of the reference sequence are segmented in non-overlapping blocks with a position range of S position numbers. The method generates for each block that has at least one sequence with a reference position in the position rang of said block a data stream wherein a course presence indicator and a fine presence indicator is present in the data stream before a first storage section record. The fine presence indicator indicates for each of F subsequent reference positions the presence of at least one sequence, the course presence indicator indicates for each group of C subsequent groups of F subsequent reference positions the present of at least one sequence in said group of F subsequent reference positions and wherein $S=F\times C$. With these features, the storage footprint for storing the reference positions of the multitude of sequences is reduced further. Assume S = 256, F=8 and C=32. To identify the presence of at least one sequence on each of the 256 reference positions, now only $32+32\times 8$ bits are needed. To store 256 individual addresses for the sequences 256×32 bits would be necessary.

In an embodiment, the position numbers of the elements of the reference sequence are segmented in non-overlapping sections with a position range of P positions. The method further comprises generating a first storage section index. An index entry is generated for each section of P positions that has at least one sequence with a reference position in its position range. A segment data stream is generated which comprises the first storage section records of the sequences having a reference position in a section of P positions. The segment data stream is stored at an address in the storage data structure. The address is assigned to the entry of the index corresponding to the section of P positions.

These features enables reducing the response time to recover the reads that have a reference position at positions of the genome that have to be studied, analysed, re-aligned or has to be used in Variant Calling. The index provides direct access to the address in the first storage section comprising a desired range.

In a further embodiment, the method comprises: determining for a segment data stream the position of the sequence having the lowest reference position and assigning a relative position value corresponding to the lowest reference position to the entry of the index corresponding to the section of P positions. These features reduces the footprint a segment data stream for a section as no data has to be stored for the part of the segment with reference positions before the lowest reference position.

In an embodiment, the method further comprises: storing a second parameter in the header section of the data structure, the second parameter enabling to obtain a value for a basis length of a sequence; and wherein the number of elements of a sequence corresponds to the value of the basis length minus the value of the length field. These features reduce the storage footprint to store the length values.

In an alternative embodiment, a record in the first storage section further comprises a first format field prior to the length field for storing a first parameter identifying the number of bits of the length field. These features reduce the storage footprint to store the length values.

In an embodiment, the method stores a first sequence and a second sequences which form a pair of sequences. A first storage section record is generated which comprising a first length field, a second length field and a gap field. The first length field and the second length field contain a value defining the number of elements of the first and second sequence respectively. The gap field has a value defining the distance between the reference position of the first sequence and the reference position of the second sequence. These features are advantageous for re-alignment of pair-end read.

According to a further embodiment, the method further comprises: generating an additional first storage section record for the second sequence in the segment data stream of the section of P positions comprising the reference position of the second sequence if the reference position of the second sequence

is located in another section of P positions than the section of P positions associated with the reference position of the first sequence. This feature is advantageous for performing Variant Calling processing on the sequences in the data storage structure. All reads that have been mapped in a particular segment of the genome could be reconstructed from the data in the respective sections associated with said segment.

In a further embodiment, the additional first storage section is preceded by a format field and a length field, and the combination of a predefined value of the format field and a predefined value of the length field indicates that the following data is an additional first storage section record. The combination of format field and length field enables to represent some values with two different formats, i.e. number of bits. By using one of the two value with a specific format as indicator, the length value can still be used but in the other format. The value with the format that represents the indicator indicates that the read length that follows is from a mate of a pair-end read.

In an embodiment, if the contiguous sequence of elements of the reference data structure does not have a sequence part of elements that fully matches a sequence the method comprises: storing for the sequence in a second storage section of the storage data structure a second storage section record. The second storage section record describes the sequence in terms enabling to reconstruct the element values of the sequence by retrieving the element value of the elements that have a matching position in the reference data structure from the associated position in the sequence of reference data structure and the element values of the elements of the sequence that does not have a matching position from the second storage section record. Contrary to what is expected, the separation of the storage of the length information of the multitude of sequences and the storage of the mutations of the sequence with respect to the reference sequence results in a reduction of the storage footprint. By using an index structure to read the data streams associated with a particular segment, the increase in time to reconstruct the sequences is negligible.

In a further embodiment, the second storage section record comprises a first field identifying the position of a mutation in the sequence and a second field identifying the type of mutation. The second storage section record

comprises a third field containing the quality of the elements which value differs from the reference. This allows reducing further the storage footprint.

Each element of a sequence has a quality value. In an embodiment, the method further comprises: determining for each position number of the reference data sequence the highest quality value of the elements of the multitude of sequences that has been mapped on said position number. A third storage section with an index is generated that enables retrieving the highest quality value for each position number from the storage section. It has been found that upgrading the quality of some elements of a read by assigning the higher quality value of the element of another read that is mapped on the same position in the reference sequence does not degraded the re-usability of the reads for re-alignment. It has been found that this increases the re-usability.

A quality value could have four different values and the position numbers of the reference sequence are segmented in non-overlapping blocks with a position range of Q position numbers. In a further embodiment, the method further comprises for each block of Q position numbers that has at least one element of the multitude of sequences mapped on the position range: determining the most common quality value; generating a first data structure identifying all positions having the most common quality value; generating a second data structure identifying all positions not having the most common quality value and the lowest quality value; generating a quality value stream identifying the quality values of all positions not having the most common quality value and the lowest quality value; and storing in the third storage section a stream of data that is a concatenation of a field with a value representing the most common quality value, the first data structure, the second data structure and the quality value stream. These features reduce the footprint to store the quality values.

Each sequence of the multitude of sequences comprises a sequence identifier. In an embodiment, the method comprises: storing the sequence identifiers in a fourth storage section that differs from the first storage section. Separating the storage of the sequence identifiers in a separate section decreases the time to retrieve the sequence information which is necessary for certain post-processing which do not need the sequence identifiers.

A sequence identifier is a string of characters with fields that are separated by a delimiter. A field is one of two types, a first type represents a string of digits, and a second type represents a string of characters with at least one letter. In a further embodiment, the method comprises: generating a lookup table comprising at least one entry with a template describing the field types of the fields of a sequence identifier and entries for each of the different values of the second type fields. For a sequence a fourth storage section record, a fourth storage section record is generated which comprises a first field with a pointer to the at least one entry with a template describing the field types of the sequence identifier and a number of next fields specified by the template retrieve from the al least one entry of the lookup table, a next field identified by the template as first type field contains a number corresponding the string of digits and a next identified by the template as second type field contains a pointer to the entry of the lookup table comprising the string of characters with at least one letter. These features reduce the storage footprint for storing the sequence identifiers.

According to a second aspect, there is provided a computer implemented method of reconstructing a sequence that have been aligned with a reference data structure from a storage data structure generated with the method according to any of the previous embodiments. The sequence comprises a number of elements having an element value; the reference data structure describes reference data as one contiguous reference sequence wherein each element of the reference sequence has a position number and element value. The method comprises: reading a first parameter from a header section of the data structure, the first parameter identifying the reference data structure; retrieving from a first storage section of the storage data structure a reference position of the sequence on the reference data structure; retrieving a length value from a length field of a first storage section record, the value enabling to determine the number of elements of the sequence; and, retrieving the values of the elements of the sequence by reading a part of the contiguous reference sequence which position is defined by the reference position and which length is defined by the length value.

According to a third aspect, there is provided a computer implemented system comprising a processor, an Input/Output device to connect to

the network system, a database and a data storage comprising instructions, which when executed by the processor cause the computer implemented system to perform any of the methods described above. There is further provided a computer program product, a processor readable medium and a database product comprising a storage data structure generated by any of the methods described above.

Other features and advantages will become apparent from the following detailed description, taken in conjunction with the accompanying drawings which illustrate, by way of example, various features of embodiments.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other aspects, properties and advantages will be explained hereinafter based on the following description with reference to the drawings, wherein like reference numerals denote like or comparable parts, and in which:

Fig. 1 illustrates an example of the general structure of a data storage structure for storing a multitude of sequences that have been aligned with a reference data structure;

Fig. 2 illustrates the data structure of the read index;

Fig. 3 illustrates an exemplary embodiment of a data stream in the read section;

Fig. 4 illustrates the data structure of a read section data record;

Fig. 5 illustrates the data structure of a stream in the Call section;

Fig. 6 illustrates the data structure of a Mutation Data record;

Fig. 7 illustrates the data structure of a quality section;

Fig. 8 illustrates an example of a data structure to store position data;

Fig. 9 illustrates the data structure of an annotation section data record;

Fig. 10 illustrates the data structure of an annotation section data record;

Fig. 11 is a block diagram illustrating a computer implemented system; and,

Fig. 12 illustrates an example of a read and corresponding reference data.

DETAILED DESCRIPTION

Alignment is the process of mapping random reads on a reference data structure based on the pattern of the read. The number of reads which stream from a sequencer varies depending on the read size. A high end sequencer can produce 120Gbase per day in short reads of 75, 100, 150 or 200 bases. The size of such stream of bases generated by a sequencer is in the order of 300Gbytes. After alignment the reads have to be stored for further processing or re-alignment. Further processing could be and is not limited to analysing the result of alignment and display the result on a display screen.

The reference data structure use for alignment could comprise the data of a genome. A genome exists of 4 different nucleotides or bases which form a string. The codes for these nucleotides are referred to in literature A C G T. These strings are connected where AT and CG form pairs. In the present application, the bases are encoded using two bits: 00 ⇔-A; 01 ⇔ C; 10 ⇔ G and 11 ⇔ T. A human genome exists of 3.2Billion base pairs. With an encoding of 2 bits per genome position, the description of the entire genome can be stored in ~750Mb. The genome is a combination of a number of chromosomes of the DNA. In a FASTA file, which is commonly used to store a genome as reference data, each chromosome is described in a section of the FASTA file.

In the present application, a sequence is a sequence of bases, for example a read generated by a sequencer, and a reference data structure is a data structure which includes the genome that is used as reference during the matching process. Each element value of the sequence has a quality value, which is an indication by the sequencing machine about the reliability of the value. A reference data structure is used which describes reference data as one contiguous reference sequence. Each element of the reference sequence has a position number and an element value. After the matching process a sequence for which at least a part with a minimal size could be matched on the reference data structure will have a corresponding matching location. There is a match when the

subsequent element values of a part of the data sequence are identical to a subsequent part of the element values of the reference sequence.

Fig. 1 illustrates an example of the general structure of a data storage structure 10 for storing a multitude of sequences that have been aligned with a reference data structure. The structure allows storing efficiently the element values of all reads generated by a sequencer, if present matching positions of the elements on the reference sequence and quality information related to each of the element values. The structure further allows reconstructing all reads that match a particular part of the reference sequence. It is not necessary to process the entire data storage structure, to retrieve only the reads of said particular part.

In the following description of the method the term "element" refers to a single base having a value A, C, G or T.

The basic concept of the method to store the data of reads is that the element value of an element of a read that has a match doesn't have to be saved as the value could be retrieved from the reference sequence. The reference sequence is always stored for future alignments. The general structure in Fig. 1 shows the sections that form part of the data storage structure.

First a short description will be given of the sections of the data storage structure. Thereafter, sections will be described in more detail with reference to corresponding Figures.

The header section 100 comprises general information about the stored data for example information about the reference data structure and information about the program that generated the storage date structure. The information about the reference data structure is used to link the data storage structure to the reference data. The information about the program that generated the data storage structure could provide the program that recovers the reads from the data structure information about parameters which define the data format. It should be noted that these parameters could also be stored in the header section directly. Furthermore, the header section comprises a parameter indicating whether single reads or pair-end reads are stored in the data storage structure.

The read section 101 and read index 102 comprises information about the length of the reads and the position of the read on the reference data sequence. The read section 101 is the first storage section. The position of a

read is determined from information stored in the read index 102 and information stored in the read section 101. The length of each read that has at least one relevant matching part on the reference sequence is stored in the read section 101. A relevant matching part is a sequence of elements with a predefined minimum length and optionally the quality of the elements is higher than a predefined minimum quality value. A predefined minimum length could be 20 If the reads are from pair-end sequence files, the length of the reads that form a pair-end and that have both at least one relevant matching part are stored together in a first storage section record.

The call section 103 and the call index 104 comprise mutation information of the reads with respect to the reference data sequence. Furthermore, it comprises, if present, quality information of the elements that does not have a matching position in the reference data sequence.

The quality section 105 and the quality index 106 comprise the quality value that is assigned to each element of the reference sequence based on the matching results of the alignment process of the multitude of reads. Each element of a read has obtained a quality value from the sequencing machine. The quality value indicates the reliability that the element value given to the element is correct. The quality value that is assigned to an element of the reference sequence corresponds to the quality value of the element of a read that matches the element of the reference sequence with the highest quality value. As a consequence of this the quality value of the elements of a read that is recovered from the data storage structure could have a higher value than originally assigned by the sequencing machine. However, this is acceptable as the quality value of an element will only be higher if there is another read which has an overlapping matching part which element has a higher quality value. As those reads have a matching part at the same location on the reference sequence, it is clear that the highest quality observation applies to the element value as the reads proof thru consensus that the element value of the elements having a particular matching position in the reference sequence has indeed the given element value.

The Annotation section 107 and annotation index 108 link to each read which length is stored in the read section a sequence identifier and an optional description. These parts of the data storage structure enable to recover

reads in their original FASTQ format. After recovering only the quality value of elements could differ from the original quality value that was assigned to the element by the sequencer.

The unmapped section 109 comprises the data of the reads that did not match with the reference sequence. In case the data storage structure stores a multitude of pair-end reads this section comprises two parts. In a first part all data of the reads that did not match but which mate did match is stored. In this part, to each record describing a read information is added that links the read to its mate which is described in the read section. This could be done by adding the reference position of the mate on the reference sequence and its index number indicating to which of the reads having the same reference position it is linked. For recovering the reads of a pair-end it is advantageous to store the unmapped reads in this part in the order of the reference position of the mates. In the second part all data of the reads is stored which did not match and if the reads are pair-end reads also its mate did not match. Pair-end reads are preferably stored as pair.

The Run Time Stats section 110 comprises information about the matching process of the reads. Information that could be stored is the amount of reads processed per time unit.

The summary section 111 provides coverage information of the aligned sequences on the reference data sequence. The coverage information comprises for each element of the reference data sequence the number of times an element of a read is mapped on the position of said element.

In case the data storage structure is one large file, the read index 102 is preferably stored after the read section 101. This also applies to the other sections. The main reason for this is that read index comprises links to addresses in the read section, which are known after generating the read section. In this way, the file could be generated without overwriting previously generated file parts on the storage medium, for example a hard disk.

Fig. 2 illustrates the data structure 200 of the read index 102, the first storage section index. The read index shown in Fig.2 comprises 2^{17} entries. The read index could be in the form of a table with 2^{17} rows and two columns. Each entry comprises a first field 202 and optionally a second field 204. A pointer 202 to an address in the read section is stored in the first field. A position number on the

reference 204 is stored in the second field. A reference data sequence of a genome of a human comprises 3.2Gbases. The 2^{17} entries are used to segment the reference data sequence in segments of equal size of 2^{15} = 32k bases. The first entry with index number 1 covers the positions 0 – 32767, the second entry with index 2 covers the positions 32768 – 65535, etc. The lengths of the reads or the pair-end reads having a reference position number in the range of an entry are stored as one stream of data in the read section 102. Preferably, the element having a matching position which has the lowest position number in the reference data sequence is used to indicate the reference position number. The pointer 202 of an entry points to the address in the read section where a stream with all reads having a reference position number in the range of positions on the reference sequence starts. The second field 204 is used to store a value to determine the position number on the reference data sequence when reading the data stream which starts in the data storage structure at the address indicated by the first field 202. The value of the second field could be an absolute position number on the reference data sequence or a relative position number in the range which corresponds to the index number of the entry. Sections of 32k positions make it possible to recover reads in a desired range of the reference data sequence in a short period of time. The sections of 32k positions are non-overlapping sections.

The number of entries of the read index 102 could be reduced by generating only an entry for a section of 32k bases if at least one read or fragment of a read in case of a break has reference position in said entry. In this case the second column with position number on the reference 204 is essentials as the index number does not provide any more the start address in the 32k range in the reference data sequence. This also applies to all other indexes that will be created for the other data sections.

Fig. 3 illustrates an exemplary embodiment of a data stream 300 or segment data stream, in the read section 102 corresponding to an entry of the index. The data stream 300 of an entry of the index enables to retrieve therefrom the reference position number and length of all reads or pair-end reads that have a reference position number in the position range on the reference data sequence defined by the index number of said entry. In the present application an entry corresponds to 32768 (32k) position numbers. The data stream 300 is composed

of a series of sub-streams. Each sub-stream enables to retrieve therefrom the reference position number and length of all reads or pair-end reads that have a reference position number in a range of 256 reference positioned numbers on the reference data sequence. The range of a sub-stream starts at a position which is a multiple of 256. The first position of the range of the first sub-stream is defined by the data from the read index 101. The first position of the range of the second sub-stream is 256 higher, etc. Accordingly, a stream 300 for 32K position numbers could comprise a series of 128 sub-streams. The sub-streams segment the position numbers of the reference sequence in non-overlapping block with a position range of 256 position numbers. It might be clear that other sizes of range are also possible.

A sub-stream starts with a first bit-mask field 301 or course presence indicator. Each bit of the first bit-mask field represents a number of position numbers. The number of position numbers depends on the number of bits of a second bit-mask field 302 which is a fine presence indicator. The first bit-mask provides coarse position information, in a range of eight positions, about the location of matched reads and the second bit-mask field provides fine information about the reference position of a read on the reference data sequence. Each bit of the second bit-mask field 302 represents one position number. A bit value “0” of a bit of the first bit-mask field 301 indicates that there is no read or pair-end read with a reference position in the range of position numbers represented by said bit. A bit value “1” of a bit the first bit-mask field 301 indicates that there is at least one read or pair-end read with a reference position in the range of positions of said bit. A bit value “1” is further an indication that a second bit-mask field 302 is present in the stream subsequent to the first bit-mask field 301. If all bits of the first bit-mask field are “0”, the sub-stream has a total length of 32 bits and will be followed by the first bit-mask field of a subsequent sub-stream. The number of bits with a value “1” defines the amount of second bit-mask fields in a sub-stream. The first second bit-mask field 302 is associated with the first bit of the first bit-mask field; the second bit-mask field 302A is associated with the second bit of the first bit-mask field 301, etc.

A bit value of “1” of the second bit-mask field 302, 302A indicates that there is at least one read or pair-end read with a reference position number

corresponding to the reference position of said bit. As there is always one bit with a value "1", the second bit-mask field 302A will be followed by a count field 303A. This count field is associated with the first bit of the second bit-mask field 302 having a value "1". The second count field 303B after the second bit-mask field is associated with the second bit of the second bit-mask field 302A having a value "1". The value of the count field 303 is the number of reads and/or pair-end raids that has a reference position at the position number associated with the bit to which the count field is associated. The count field 303, 303A has a value greater than or equal to 1. A count field will be followed by a number of read section data records equal to the value of the count field. The data structure of the read section data records will be described with reference to Fig. 4.

Fig. 3 shows an example of a first sub-stream and the beginning of a second sub-stream. The first bit-mask field 301 of the first sub-stream comprises three bits with a value "1". Consequently, the first bit-stream comprises three second bit-masks field 302, 302A and 302B. The first second bit-mask comprises one bit with a value "1" and is consequently followed by one count field 303 and a number of read section data records according to the value of the could field, in the present example two.

The reference position number on the reference data sequence of the first data record 304 and second data record 304A is 8 plus 10×8 plus the reference position number from the second field of the entry associated with the stream 300. The number of read section data records in a sub-stream is equal to the sum of all count field values of the sub stream. The number of count fields in a sub stream is equal to the number of bits of all the second bit-mask fields of a sub-stream with a value "1".

Fig. 4 illustrates the data structure of a read section data record 400 or first storage section record. This data structure is used to store the necessary information to recover the length of a single read or the lengths of both reads of a pair-end and the number of elements between the reference positions of the reads or pair-end reads that have both a match on the reference data structure. To store single read only fields 401 – 403 are used. From a parameter in the header section is known whether single reads or pair-end reads are stored. The length of a read is the number of elements or bases of the sequence of a read with a

minimal quality. Normally, the elements of a read with a quality lower than a predefined level are clipped from the sequence prior to matching. Therefore, the length of the reads does not have a fixed value. This phenomenon is known to the person skilled in the art.

In the length fields 402, 406 the absolute length of a read could be stored. Normally about 95% of the reads that matches have a length in the range of 70 – 100 when the sequencing machine generates reads with a length of 100 bases. To store a value of 100, the length field should have 7 bits. To store all absolute length of 100 reads 700 bits are needed.

According to an embodiment of the present application a common read length value is stored in a length field of the header section 101, for example the value 100. The difference of this common length value and the length in the length field 402 results in the stored length of the read. This allows a reduction of the number of bits for the reads having a length in the range of 70 – 100 to five bits. To be able to assign all length outside the range 7 bits are needed. The bits of the length field represent a positive integer. A format bit in a format field is needed to define the number of bits that is used to store the value. When 95% of the reads have a length in the range of 70 – 100, then 570 bits are needed to store the length of 95 reads and 40 bits are needed to store the length of the 5% reads outside the range. Thus 610 bits are now needed instead of 700 bits. If the format bit indicates that the length field comprises 5 bits, the length of the sequence is obtained by subtracting the value from the common length value. In the format bit indicates that the length field comprises 12 bits, the length of the sequence is the value of the length field. It should be noted that other solution are possible for example the length is determined by summing the value of the length field and the common length value. In this case the five bits of the length field should represent both positive and negative integer values.

Therefore a read section data record comprises a first format field 401 which is format-bit F-Bits1 indicating the number of bits that is used for the first length field 402. A value “0” is used to indicate that the first length field comprises V1 bits and a value “1” is used to indicate that the first length field comprises V2 bits. The parameters V1 and V2 could be defined in the header section 101 or could be defined by identification of the program that generated the

data storage structure. This identification is also stored in the header section 101. In an embodiment, V1 is 5 and V2 is 12. A value for V2 of 12 is used to make this concept also suitable for very long reads with an average length longer than 256 and that one method could be used to generate the data store structure. The combination of a format field before a value field wherein value of the format field defines the number of bits of the following value field will be referred to as variable bit length data format.

After the first length field 402, there is a first format field 403. The first format field 403 comprises two bits. A first bit indicates from which of the two files of pair-end sequences files the read is coming and a second bit indicates the order type on how the element values should be reconstructed from the reference data sequence. The order type could be forward or reverse-complement.

In distance field 404 the distance between the reference position of the first read and the second read is stored. In this way pair-end reads could be retrieved efficiently from the data storage structure and les bits are needed to store the reference position of the mate. The information of the read of a pair-end with the lowest reference position is stored in the fields prior to the distance field 404 and the information of the read of a pair-end with the highest reference position is stored in the field after the distance field. 404. The variable bit length data format is preferably used to store the value.

After the distance field 404, the same data structure is used to store the data of the mate. There is a second format field 405, a second length field 406 and a second formats field 407.

If a read does not have a mate that matches on the reference data sequence, the distance field 404 is used to indicate this. When the variable bit length data format is used to store an integer value, some values can be represented in two different formats. By reserving the highest value(s) of the representation with the lowest number of bits as indicator and not using these values as distance value, additional functionality can be added to the distance field.

To store the mutations of the read with respect to the reference sequence, the data storage structure comprises a Call section 103 and a Call index 104. The Call index 104 is an index which divides the position numbers of

reference data sequence in a similar manner as the read index 102. However an entry of the index comprises now only a pointer to the start address in the Call section of the stream comprising all mutations of the read having a reference position in the range corresponding to the entry number. It should be noted that frequently, most of the reads, 95%, are a perfect match and only 5% comprises mutations that have to be stored in the Call section.

Fig. 5 shows the data structure of a stream 500 in the Call section 103. The stream in the Call section is a concatenation of sub-streams. A stream comprises all mutations of the read having a reference position in a range of 32768 positions. Each sub-stream describes all mutations of all reads or pair-end reads having the same reference position. A sub-stream starts with a position field 501, followed by alternating an index number field 502, 502A and mutation data record 503, 503A and ends with an EOP (end of position) field. The position fields 501, 501A define the offset of the reference position of the reads with respect to the previously known reference position. The value of the first position field 501 of the stream is the offset in position with respect to the first reference position in the range of the associated entry which directed to read the present stream. The value of the second position field 501A in the stream is the offset with respect to the reference position calculated by the first position field 501. The index number field 502 indicates the index number of the reads at said reference position stored in the read section 102. In this way a link is created between the length information and mutation information of a read. The index number field is used as only mutation information of read that does not have a perfect match comprises a mutation data record 503, 503A in the Call Section 103. A stream of the Call section 103 ends with an EOS (End-Of-Stream) field 505. It should be noted that the EOS field 505 could be the EOP field of the last sub-stream in a section. The number of bits of the position fields could be fixed or a variable bit length data format could be used.

In Fig. 5, the first sub-stream comprises mutation information of two reads and the second sub-stream comprises mutation information of only one read.

Fig. 6 shows the data structure 600 of a Mutation Data Record 503, 503A. A mutation Data Record is also a data stream which is a concatenation of

one or more sub-streams and at the end and EOCS (End Of Call Stream) field 604. A sub-stream comprises a distance field 601, 601A, a call type field 602, 602A and a call data field 603, 603A.

The distance field 601A defines the offset of the first position of the mutation defined by call type field and call data field with respect to the last position of the previous mutation. The first distance field 601 defines the offset with respect to the reference position of the read.

In the Call type field 602, 602A the type of mutation is stored. The call type field comprises four bits and enables to define sixteen call types. Below a list of call types is given:

0	UN 1	
1	UP X	To indicate that two or subsequent elements of read have another values than the values at corresponding positions in reference sequence
2	UP A	To indicate that one element with value A of read has another value than the value at corresponding position in reference sequence
3	UP C	To indicate that one element with value C of read has another value than the value at corresponding position in reference sequence
4	UP G	To indicate that one element with value G of read has another value than the value at corresponding position in reference sequence
5	UP T	To indicate that one element with value T of read has another value than the value at corresponding position in reference sequence
6	IN #	To indicate that two or elements are not present in reference sequence
7	IN A	To indicate that one element with value A is not present in reference sequence
8	IN C	To indicate that one element with value C is not present in reference sequence
9	IN G	To indicate that one element with value G is not present in reference sequence
10	IN T	To indicate that one element with value T is not present in reference

		sequence
11	DEL	To indicate to one or more subsequent elements of reference sequence read are not present in read
12	UNM	To store large sections of a long read that could not be mapped
13	BRK	To indicate a break in the sequence
14	RID	To couple mutations that follow a read
15	CLIP	To store initial/end part of data sequence that is not used for mapping

Depending the call type the call data field 603, 603A comprises at least one of length data, a sequence of element values, quality information of the changed or inserted elements, relative or absolute position information.

To store the quality information of the elements of the reads, for each element of the reference data sequence a quality value is determined by means of the quality information of the elements of the read having a matching position and value on the reference data sequence. As the quality value is an indication of the likelihood that a value has the assigned value, the highest quality value is the best estimate of the consensus quality value. Consequently the highest quality value will be stored. In FASTQ files the quality could have an integer value from 0 to 40. It has been found that 4 levels could be used without losing essential information. In the default setting of present application quality values in the range 0 – 16 are mapped on value 0, the values in the range 17 – 25 are mapped on value 1, the values in the range 26 – 34 are mapped on value 2 and the values 35 – 40 are mapped on value 3. This makes it possible to store just one quality value of two bits for all elements of reads that are mapped on a reference position of the reference data sequence and that has the same element value.

In the quality section 106 to each position of the reference data sequence is assigned a quality value of two bits. The quality section 105 comprises sections of 32k positions similar to the division in sections as the read section and call section. Each section is divided into subsections of 128 reference positions. Consequently, a section of 32k comprises 256 subsections. The Quality index has an index similar to the index of the read section. To reduce the size of the index, only entries are generated for the 32k sections wherein at least one element of a read has a matching location.

Fig. 7 shows the data structure 700 of the quality section 105. It comprises one data stream that is a concatenation of 256 sub-streams. Each sub-stream comprises the quality information of 128 reference positions. A sub-stream comprises four fields, a first field 701, 701A, a second field 702, 702A, a third field 703, 703A and a fourth field 704, 704A. The first field 701 comprises two bits which value corresponds to the quality value unequal to "0" that has the most occurrences in the range associated with the sub-stream. This value is called the Most Common Quality Value MCQV. The second field 702 identifies all positions in the range having a quality value equal to the MCQV. The third field 703 identifies all positions having a value not equal to "0" and the MCQV, thus all other quality values. In the fourth field 704, the quality values of the positions identified in the third field 703 are given as a sequence. In principle one bit is sufficient to identify the value of the other values other than "0" and the MCQV. To increase the retrieving speed it is advantageous to use the representation of the original format of the quality value.

The position data in the second field 702 and third field 703 could be compressed by run-length encoding. Fig. 8 illustrates another embodiment to reduce the amount of bits to store the position data. The data structure 800 is similar to the data structure used in the read section to identify a read at a particular position. The data structure comprises a first part 801 of 8 bits and a second part 802 of 16 times the amount of bits in the first part having a value "1". The first 16 bits are associated with the first bit of the first part having a value "1"; the second 16 bits are associated with the second bit of the first part having a value "1". In Fig. 8, the first part comprises two bits with a value "1". Therefore, the second part comprises 2×16 bits. In the example shown in Fig. 8, position 72 and positions $78 - 91$ are identified.

It should be noted that if a section only comprises "0" the first field 701 could have a value "0".

In a FASTQ-file all reads/sequences have a sequence identifier. A general example of a sequence identifier is EAS234:6:FC693VJ:1056:17853:204586:1. In general a sequence identifier is a string of characters with fields that are separated by a Colon ":". The given example comprises seven fields. The fields could represent sequentially: the

unique instrument name; the run id; the flowcell id, the tile number within the flowcell lane; 'x'-coordinate of the cluster within the tile; 'y'-coordinate of the cluster within the tile; the member of a pair. There are generally two types of fields: 1) string with at least one letter and 2) string with only digits. It has been found that the amount of different values in fields with at least one letter is limited whereas the amount of different values in field with only digits is huge. Furthermore, the number of different formats of a sequence identifier that has to be stored in one data storage structure is limited. Therefore, a special data structure is developed to reduce the amount of storage capacity to store the sequence identifiers.

For each data storage structure a look-up table is generated. In some entries the different formats of the sequence identifiers is store. In other entries the values of the fields with at least one letter are stored. The look-up table could be stored in the header section 100 or in the annotation index section 108. To link the sequence identifier with the read stored in the read section 101 an index structure is used which is similar to the read index structure as shown in Fig. 2. Furthermore, the annotation section has a data structure similar to that of the read section 101 as shown in Fig. 3 and described in the corresponding part of the description. The only difference is the content and format of the Data Record fields 304, 304A in Fig. 3.

Fig. 9 illustrates the data structure 900 of an annotation section data record. The first field 901 of the annotation section data record comprises a pointer to the entry of the loop-up table which comprises the description of the format of the read_name/sequence identifier for the read/sequence. Fig. 10 illustrates a possible data structure 1000 to describe the format of a sequence identifier. The data structure comprises a number of fields. Each of the fields comprises a value identifying the type of the corresponding field. Given the example of a sequence identifier above, the first field 1001 comprises a value indicating the first data field 902 after first field 901 of the annotation section data record is a pointer to an entry of the look-up table. The entry which comprises the string of characters “EAS234”, which is the first part of the sequence identifier. The second field 1002 comprises a value indicating the second data field 903 after first field 901 of the annotation section data record is an integer value. The integer

value could have a variable bit length data format which comprises two fields. The first field, a format field, indicates the number of bits in the second field to represent the integer value. In an embodiment, the first field comprises 2 bits, wherein the values "00", "01", "10" and"11" indicates that the second field comprises respectively 4, 8, 16 and 32 bits to represent the integer value. It should be noted that this format could also be used to store a pointer to the look-up table. The third field 1003 comprises a value indicating that the third data field 904 after first field 901 of the annotation section data record is a pointer to an entry of the look-up table. The entry which comprises the string of characters "FC693VJ", which is the third part of the sequence identifier. The fourth to seventh field 1004 - 1007 comprise values indicating the fourth to seventh data field 905 - 908 after first field 901 of the annotation section data record are integer value. The variable bit length data format is used to store the integer values. The last field 1008 in the present example comprises a value EOD (End of Description) to indicate that there are no further parts to describe the sequence identifier.

When the sequence identifier of given example is stored as a string of 36 characters with six Colons separating the seven fields, 288 bits storage capacity is needed. With the described data storage structure the string could be stored in less than 238 bits. Assuming that all values in an annotation record can be represented with 16 bits or less, the 36 characters can be stored in less than 126 bits.

Due to a break or a fusion, it might happen that a read has a first part, initial sequence part, in one section of 32k bases and a subsequent second part, fragment sequence part, in another section of 32K bases, wherein the two sections are not subsequent. For variant calling it is important that all fragments/reads that have a match in a particular section could easily be retrieved from the data storage structure. To enable this, the information to reconstruct the second part from the storage data structure is stored twice in the data storage structure by generating an additional read section data record and if necessary mutation data record. The first time as part of the read in the section of 32k bases wherein the reference position of the read is located, and a second time in the section of 32k bases wherein the reference position of the second part is located. Similarly, it might happen that from a pair-end read, the first read has a reference

position in one section of 32k bases and the mate has a reference position in another section of 32k. The mate is than a so-called cross reference read. The information to reconstruct the mate is stored twice in the data storage structure. To enable the second storage of the fragment or mate in a read section data record, the two highest values of the variable bit length format with the less number of bits are used to indicate the type of data that is stored twice. After this special use of fields 401 and 402 of a read section data record, the fragment or mate will be stored with the format as shown in Fig. 4. It should be noted that if a read with a break has a reference position in one section and the part of the sequence after the break is in another section and the mate is also in the another section, the sequence part after the break is stored as a fragment read in the another section and the mate is stored in the another section as an cross-reference read.

Referring to Fig. 11, there is illustrated a block diagram of exemplary components of a computer implemented system 1100. The computer implemented system 1100 can be any type of computer having sufficient memory and computing resources to perform the described method. As illustrated in Fig. 11, the processing unit 1100 comprises a processor 1110, data storage 1120, an Input/Output unit 1130 and a database 1140. The data storage 1120, which could be any suitable memory to store data, comprises instructions that, when executed by the processor 1110 cause the computer implemented system 1100 to perform the actions corresponding to any of the methods described in the present application. The Input/Output unit 1130 retrieves the aligned sequences directly from an alignment process or in the form of a combination of FASTQ, SAM or BAM files. The database 1140 could be used to store the reference data structure and the data storage structure.

The method could be embodied as a computer program product comprising instructions that can be loaded by a computer arrangement, causing said computer arrangement to perform any of the methods described above. The computer program product could be stored on a computer readable medium.

A product of the method of storing a multitude of sequences is a database product comprising a data storage structure and reference data

structure. The database product could be in the form of one or more files on a computer readable medium.

Fig. 12 illustrates an example where a data sequence 1202 of 36 bases is mapped on a reference data sequence 1201. The data sequence is stored in the data storage structure in the following way. First, the length, 36 bases, is stored with the read index and bit-mask field 301 and 302 at reference position 856. Secondly, a mutation data record is created which is a stream of fields with the following data:

Dist=7; Call_Type=UPD X; Call_Data=2,TT,2 Quality Values;

Dist=9; Call_Type=DEL; Call_Data=3;

Dist=8; Call_Type=IN #; Call_Data=4,GTAC,4 Quality Values;

Dist=5; Call_Type=UP C; Call_Data= 1 Quality Value.

The data storage structure shown in Fig. 2 could be stored as one single file. However it is also possible that each section or combination of data section and index section are stored as separate files.

From the detailed description of the storage data structure above, a skilled person could easily develop a computer implemented method of storing in the described format of the present application for storing a multitude of sequences which generates the necessary sections, storage section records, count values, data streams, bit-masks, presence indicators, indexes, lookup tables, etc. Similarly, a skilled person could easily develop a computer implemented method of reconstructing a multitude of sequences from the described storage data structure.

The present application describes a method of storing/reconstructing in/from a storage data base a multitude of sequences. The method of storing is a combination of encoding and compression. The method of reconstructing is a combination of decoding and decompression. The storage data structure is such that the response time to reconstruct sequences from a desired position range of the reference data sequence is short for viewers, re-alignment, variant calling and other known post processing tools.

While the invention has been described in terms of several embodiments, it is contemplated that alternatives, modifications, permutations and equals thereof will become apparent to those skilled in the art upon reading the

specification and upon study of the drawings. The invention is not limited to the illustrated embodiments. Changes can be made without departing from the idea of the invention.

****

5

CONCLUSIES:

1. Een op een computer geïmplementeerde werkwijze voor het opslaan in een opslaggegevensstructuur (10) van een veelheid van sequenties die zijn uitgelijnd met een referentiegegevensstructuur, de referentiegegevensstructuur beschrijft referentiedata als een aaneengesloten referentiesequentie waarbij elk element van de referentiesequentie een positienummer en elementwaarde heeft, een sequentie omvat een aantal elementen met elementwaarden die overeenkomen met een deel van de referentiesequentie, het deel van de referentiesequentie heeft een overeenkomstige referentiepositie, waarbij de werkwijze omvat:

- het opslaan van een eerste parameter in een headersectie (100) van de gegevensstructuur, de eerste parameter identificeert de referentiegegevensstructuur;
- het opslaan van de referentieposities en een veelheid van eerste opslagsectierecords (304, 304A) voor de veelheid van sequenties in een eerste opslagsectie (101) van de gegevensopslag, een eerste opslagsectie record is gekoppeld aan tenminste een sequentie die heeft een overeenkomstige referentiepositie en de eerste opslagsectie omvat verder een lengteveld (402) met een waarde die het mogelijk maakt het aantal elementen van de ten minste ene sequentie te bepalen.

2. Werkwijze volgens conclusie 1, waarbij de werkwijze verder omvat:

- het tellen van de sequenties met dezelfde referentiepositie om een telwaarde te verkrijgen;
- het genereren van een gegevensstroom door aaneenschakeling van de telwaarde (303) en de eerste opslagsectierecords (304, 304A) behorende bij de sequenties die dezelfde referentiepositie hebben.

3. Werkwijze volgens conclusie 2, waarbij het positienummers van de referentiesequentie zijn gesegmenteerd in niet-overlappende blokken met een positie bereik van S positienummers, de methode genereert een datastroom (300) voor elk blok dat ten minste een sequentie met een referentiepositie in het

positiebereik van het blok heeft, waarin een grove aanwezigheidsindicator (301) en een fijne aanwezigheidsindicator (302) in de datastroom aanwezig is voor een eerste opslagsectierecord (304, 304A), de fijne aanwezigheidsindicator geeft voor elk van F opeenvolgende referentieposities de aanwezigheid van tenminste een sequentie, de grove aanwezigheidsindicator geeft voor elke groep van C opeenvolgende groepen van F opeenvolgende referentieposities de aanwezigheid van ten minste een sequentie in genoemde groep van F opeenvolgende referentie posities aan en waarbij $S=F\times C$.

4. Werkwijze volgens een van de conclusies 1 - 3, waarbij de positienummers van de elementen van de referentiesequentie zijn gesegmenteerd in niet-overlappende secties met een positiebereik van P posities, de werkwijze omvat verder:

- het genereren van een eerste opslagsectie-index (200) waarbij elke sectie van P posities die ten minste een sequentie met een referentiepositie in het positie bereik heeft een entry heeft;
- het genereren van een segmentdatastroom (300) omvattende de eerste opslagsectierecords (304, 304A) van de sequenties met een referentiepositie in een sectie van P posities;
- het opslaan van de segmentdatastroom op een adres in de opslaggegevensstructuur, en,
- het toekennen van het adres (202) aan de entry van de index die overeenkomt met de sectie van P posities.

5. Werkwijze volgens conclusie 4, waarbij de werkwijze verder omvat:

- het bepalen van de positie van de sequentie met het laagste referentiepositie voor een segmentdatastroom;
- het toekennen van een relatieve positie die behoort bij de laagste referentiepositie aan de entry van de index die behoort bij het gedeelte van P posities.

6. Werkwijze volgens een van de conclusies 1 - 5, waarbij de werkwijze verder omvat:

- het opslaan van een tweede parameter in de headersectie van de gegevensstructuur, de tweede parameter maakt het mogelijk een waarde voor een basislengte van een sequentie te verkrijgen; en waarbij het aantal elementen van de sequentie overeenkomt met de waarde van de basislengte minus de waarde van het lengteveld.

7. Werkwijze volgens een van de conclusies 1 - 6, waarbij een record (304) in de eerste opslagsectie verder omvat een eerste formaatveld (401) voorafgaand aan het lengteveld (402) voor het opslaan van een eerste parameter die aangeeft het aantal bits van het lengteveld (402).

8. Werkwijze volgens een van de conclusies 1 - 7, waarbij de werkwijze voor het opslaan van een eerste sequentie en een tweede sequentie die een sequentiepaar vormen omvat:

- het genereren van een eerste opslagsectie record (304) omvattende een eerste lengteveld (402), een tweede lengteveld (406) en een tussenruimteveld (404), het eerste lengteveld en het tweede lengte veld hebbende een waarde die het aantal elementen van respectievelijk de eerste en tweede sequentie aangeeft en het tussenruimteveld heeft een waarde die het verschil tussen de referentiepositie van de eerste sequentie en de referentiepositie van de tweede sequentie aangeeft.

9. Werkwijze volgens conclusie 8 in combinatie met conclusie 4, waarbij de werkwijze verder omvat:

- het genereren van een extra eerste opslagsectierecord voor de tweede sequentie in de segment datastroom van de sectie van P posities omvattende de referentiepositie van de tweede sequentie indien de referentiepositie van de tweede sequentie is gelegen in een andere sectie van P posities dan de sectie van P posities geassocieerd met de referentiepositie van de eerste sequentie.

10. Werkwijze volgens conclusie 9, waarbij het extra eerste opslagsectierecord voorafgegaan wordt door een formaatveld en een lengteveld en de combinatie van een vooraf bepaalde waarde van het formaatveld en een

vooraf bepaalde waarde van het lengteveld indiceert dat de volgende data een extra eerste opslagsectierecord is.

11. Werkwijze volgens een van de conclusies 1 - 10, waarbij wanneer de opeenvolgende sequentie van elementen van de referentiedatastructuur niet een sequentiegedeelte van elementen heeft die volledig overeenkomt met een sequentie omvat de werkwijze:

- het opslaan voor de sequentie een tweede-opslagsectierecord (503) in een tweede opslagsectie (103) van de opslaggegevensstructuur, het tweede opslagsectierecord beschrijft de sequentie in termen die het mogelijk maken de elementwaarden van de sequentie te reconstrueren door het ophalen van de elementwaarde van de elementen die een overeenkomende positie in de referentiedatastructuur vanuit de bijbehorende positie in de sequentie van referentiedatastructuur hebben en de element waarden van de elementen van de sequentie die niet een overeenkomende positie vanuit het tweede opslagsectierecord (503).

12. Werkwijze volgens conclusie 11, waarbij het tweede opslagsectierecord omvat een eerste veld (601) dat aangeeft de positie van een mutatie in de sequentie en een tweede veld (602) dat aangeeft het mutatietype.

13. Werkwijze volgens conclusie 12, waarbij het tweede opslagsectierecord een derde veld (603) omvat die bevat de kwaliteit van de elementen waarvan waarde afwijkt van de referentie

14. Werkwijze volgens conclusie 10 in combinatie met conclusie 4, waarbij een sequentie een initieel sequentiedeel met een referentiepositie in een eerste sectie van P posities en een daaropvolgende fragmentsequentiedeel met een referentiepositie in een tweede sectie van P posities omvat, de werkwijze omvat verder:

- het genereren van een extra eerste opslagsectierecord voor het fragmentsequentiedeel in de tweede sectie van P posities.

15. Werkwijze volgens een van de conclusies 1 - 14, waarbij elk element van een sequentie een kwaliteit waarde heeft, de werkwijze omvat verder:

- het bepalen voor elk positienummer van de referentiedatasequence van de hoogste kwaliteitswaarde van de elementen van de veelheid van sequenties die zijn gemapt op genoemde positienummer;
- het genereren van een derde opslagsectie (105) met een index (106) die het mogelijk maakt de hoogste kwaliteit voor elk positienummer vanuit de opslagsectie (105) op te halen.

16. Werkwijze volgens conclusie 15, waarbij een kwaliteitswaarde vier verschillende waarden kan hebben en de positienummers van de referentiesequentie zijn gesegmenteerd in niet-overlappende blokken met een positiebereik van Q positienummers, de werkwijze verder omvat voor elk blok van Q positie nummers die ten minste een element van de veelheid van sequenties heeft die gemapt is op het positie bereik:

- het bepalen van de meest voorkomende kwaliteitswaarde;
- het genereren van een eerste datastructuur (702) die alle posities met de meest voorkomende kwaliteitswaarde identificeert;
- het genereren van een tweede datastructuur (703) die identificeert alle posities die niet hebben de meest voorkomende kwaliteitswaarde en de laagste kwaliteitswaarde;
- het genereren van een kwaliteitswaardenstroom (704) die identificeert de kwaliteitswaarden van alle posities die niet de meest voorkomende kwaliteitswaarde en de laagste kwaliteitswaarde hebben;
- het opslaan in de derde opslagsectie van een datastroom welke een aaneenschakeling is van een veld (701) met een waarde die representeert de meest voorkomende kwaliteitswaarde, de eerste datastructuur (702), de tweede datastructuur (703) en de kwaliteitswaardenstroom (704).

17. Werkwijze volgens een van de conclusies 1 - 16, elke sequentie van de veelheid van sequenties omvat een sequentie-identificator, waarbij de werkwijze verder omvat:

- het opslaan van de sequentie-identificatoren in een vierde opslagsectie (107) die verschilt van de eerste opslagsectie (101).

18. Werkwijze volgens conclusie 17, een sequentie-identificator is een tekenreeks met velden die worden gescheiden door een scheidingsteken een veld is een van twee typen, een eerste type representeert een reeks cijfers, een tweede type is een tekenreeks met ten minste een letter, waarbij de werkwijze verder omvat:

- het genereren van een zoektabel omvattende ten minste een entry met een sjabloon (1000) die beschrijft de veld types van de velden van een sequentie-identifier en entries voor elk van de verschillende waarden van de velden van het tweede type;

- het genereren voor een sequentie van een vierde opslagsectierecord (304), het vierde opslagsectierecord omvat een eerste veld (901) met een pointer naar de ten minste ene entry met een sjabloon die beschrijft de veld types van de sequentie-identificator en een aantal volgende velden die gespecificeerd worden door het sjabloon dat opgehaald is uit de ten minste ene entry van de zoektabel, een volgend veld (901 ... 908) geïdentificeerd door het sjabloon als eerste type veld bevat een nummer dat correspondeert met de reeks cijfers en een volgende geïdentificeerd door het sjabloon als tweede type veld bevat een verwijzing naar de entry van de zoektabel die omvat de tekenreeks met minstens een letter.

19. Een op een computer geïmplementeerde werkwijze voor het reconstrueren van een sequentie die is uitgelijnd met een referentiegegevensstructuur uit een opslaggegevensstructuur die gemaakt is met de werkwijze volgens een van de conclusies 1 - 18, de sequentie omvat een aantal elementen met een elementwaarde, de referentiegegevensstructuur beschrijft referentiedata als een aaneengesloten referentiesequentie waarbij elk element van de referentiesequentie een positienummer en elementwaarde heeft, de werkwijze omvat:

- het lezen van een eerste parameter van een headersectie van de gegevensstructuur, de eerste parameter identificeert de referentiegegevensstructuur;

- het ophalen vanuit een eerste opslagsectie van de opslaggegevensstructuur van een referentiepositie van de sequentie op de referentiegegevensstructuur;
- het ophalen van een lengtewaarde vanuit een lengteveld van een eerste opslagsectierecord, de waarde maakt het mogelijk het aantal elementen van de sequentie te bepalen, en,
- het ophalen van de waarden van de elementen van de sequentie door het lezen van een deel van de aaneengesloten referentiesequentie waarvan positie wordt bepaald door de referentiepositie en waarvan de lengte wordt bepaald door de lengtewaarde.

20. Werkwijze volgens conclusie 19, waarbij de eerste opslagsectie een datastroom omvat die is verkregen door aaneenschakeling van een telwaarde die het aantal sequenties met dezelfde referentiepositie aangeeft en de eerste opslagsectierecords behorende bij de sequenties die dezelfde referentie positie hebben, de werkwijze omvat verder:
- het ophalen van de telwaarde vanuit de datastroom, en,
- het ophalen van de gegevens van N eerste opslagsectie records, waarbij N overeenkomt met de telwaarde.

21. Werkwijze volgens conclusie 19, waarbij de werkwijze verder omvat:
- het lezen van een tweede parameter in de headersectie van de gegevensstructuur, de tweede parameter maakt het mogelijk een waarde voor een basislengte van een sequentie te verkrijgen;
- het aftrekken van de lengtewaarde van de waarde voor de basislengte om het aantal elementen van het sequentie te verkrijgen.

22. Werkwijze volgens conclusie 20, waarbij de werkwijze verder omvat:
- het lezen van een eerste parameter die het aantal bits van het lengte veld van een eerste opslagsectierecord identificeert, en,
- het lezen van een aantal bits corresponderend met de eerste parameter om de waarde van het lengteveld te verkrijgen.

23. Werkwijze volgens een der conclusies 19 - 22, waarbij de werkwijze verder omvat een paar sequenties vanuit de opslag gegevensstructuur voor het ophalen van:

- het bepalen van een eerste referentiepositie geassocieerd met het eerste opslagsectierecord met de gegevensstructuur om toegang te verkrijgen tot het eerste opslagsectierecord;
- het lezen vanuit een eerste lengteveld, een tweede lengteveld en een tussenruimteveld uit het eerste opslagsectierecord van een eerste lengtewaarde, een tweede lengtewaarde, en een afstandswaarde;
- het reconstrueren van een eerste sequentie door het lezen van een deel van de aaneengesloten referentiesequentie waarvan de positie wordt bepaald door de eerste referentiepositie en waarvan de lengte wordt bepaald door de eerste lengtewaarde;
- het optellen van de afstandswaarde bij de referentiepositie om een tweede referentiepositie geassocieerd met een tweede sequentie te verkrijgen, en,
- het reconstrueren van de tweede sequentie door het lezen van een deel van de aaneengesloten referentiesequentie waarvan de positie wordt bepaald door de tweede referentiepositie en waarvan de lengte wordt bepaald door de tweede lengtewaarde.

24. Werkwijze volgens een der conclusies 19 - 23, waarbij de werkwijze is geconfigureerd om een sequentie te reconstrueren door het combineren van elementwaarden opgehaald uit de aaneengesloten referentiesequentie en elementwaarden opgehaald uit een tweede opslagsectie dat alle mutaties van de sequentie ten opzichte van de referentiesequentie bevat.

25. Werkwijze volgens een der conclusies 19 - 24, waarbij de werkwijze verder omvat het ophalen van de kwaliteitswaarden behorende bij elementen van de sequentie waarvan de waarden opgehaald zijn uit de referentiesequentie vanuit een derde opslagsectie die een kwaliteitswaarde toekent aan een positie van de referentiesequentie.

26. Werkwijze volgens conclusie 22, waarbij de werkwijze een vooraf bepaalde combinatie van eerste parameterwaarde en de waarde van het bijbehorende lengteveld als een fragment-read detecteert en de sequentie geassocieerd met de volgende lengte-informatie dienovereenkomstig verwerkt.

27. Een op een computer geïmplementeerd systeem (1100) omvattende een processor (1110), een invoer / uitvoer inrichting (1130), een databank (1140) en een gegevensopslag (1120) verbonden met de processor, de gegevensopslag omvat instructies die, wanneer uitgevoerd door de processor (1110), ertoe leiden dat het op de computer geïmplementeerde systeem de werkwijze uitvoert volgens een van de conclusies 1 - 26.

28. Een computerprogramma omvattende instructies die door een computerinrichting kunnen worden geladen, en die er toe leiden dat de computerinrichting een van de werkwijzen volgens conclusies 1 - 26 uitvoert.

29. Een voor een processor leesbaar medium voorzien van een computerprogramma omvattende instructies die door een computer inrichting kunnen worden geladen, en die er toe leiden dat de computer inrichting een van de werkwijzen volgens conclusies 1 - 26 uitvoert.

30. Een databankproduct omvattende een opslaggegevensstructuur die gegenereerd is door een van de werkwijzen volgens conclusies 1 - 18.

****

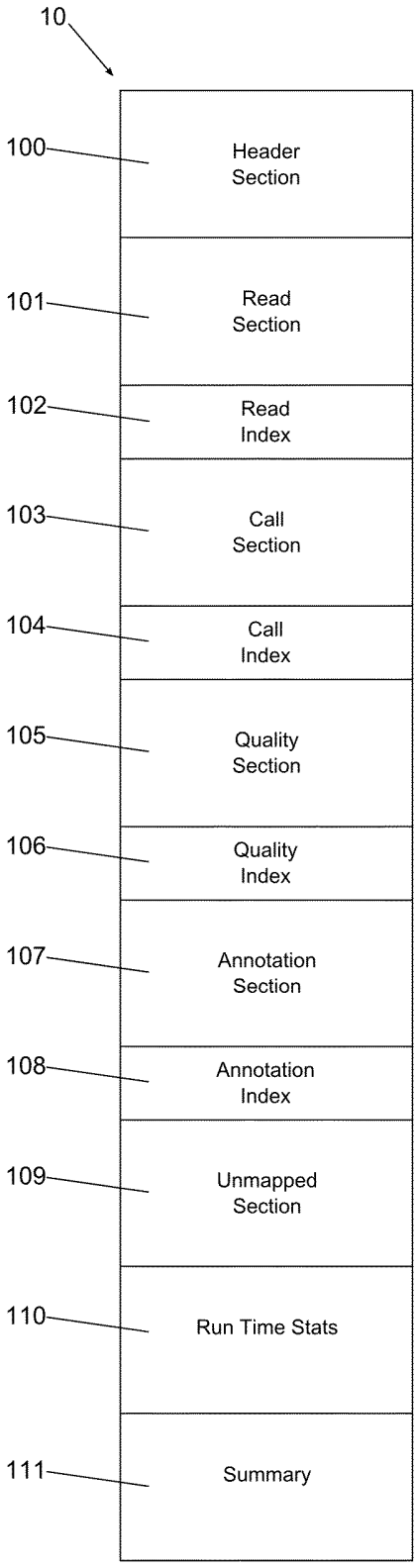

10
100
Header Section
101
Read Section
102
Read Index
103
Call Section
104
Call Index
105
Quality Section
106
Quality Index
107
Annotation Section
108
Annotation Index
109
Unmapped Section
110
Run Time Stats
111
Summary

FIG. 1

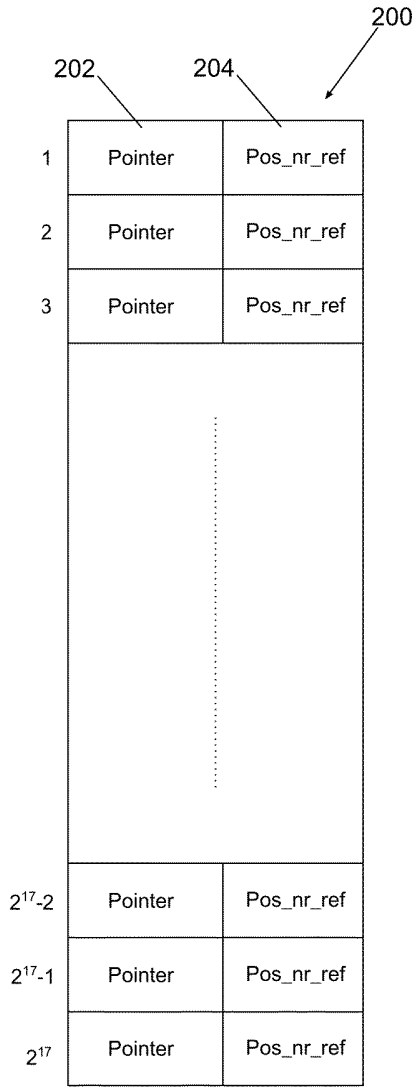

200
202
204
1 Pointer Pos_nr_ref
2 Pointer Pos_nr_ref
3 Pointer Pos_nr_ref
2^{17}-2 Pointer Pos_nr_ref
2^{17}-1 Pointer Pos_nr_ref
2^{17} Pointer Pos_nr_ref

FIG. 2

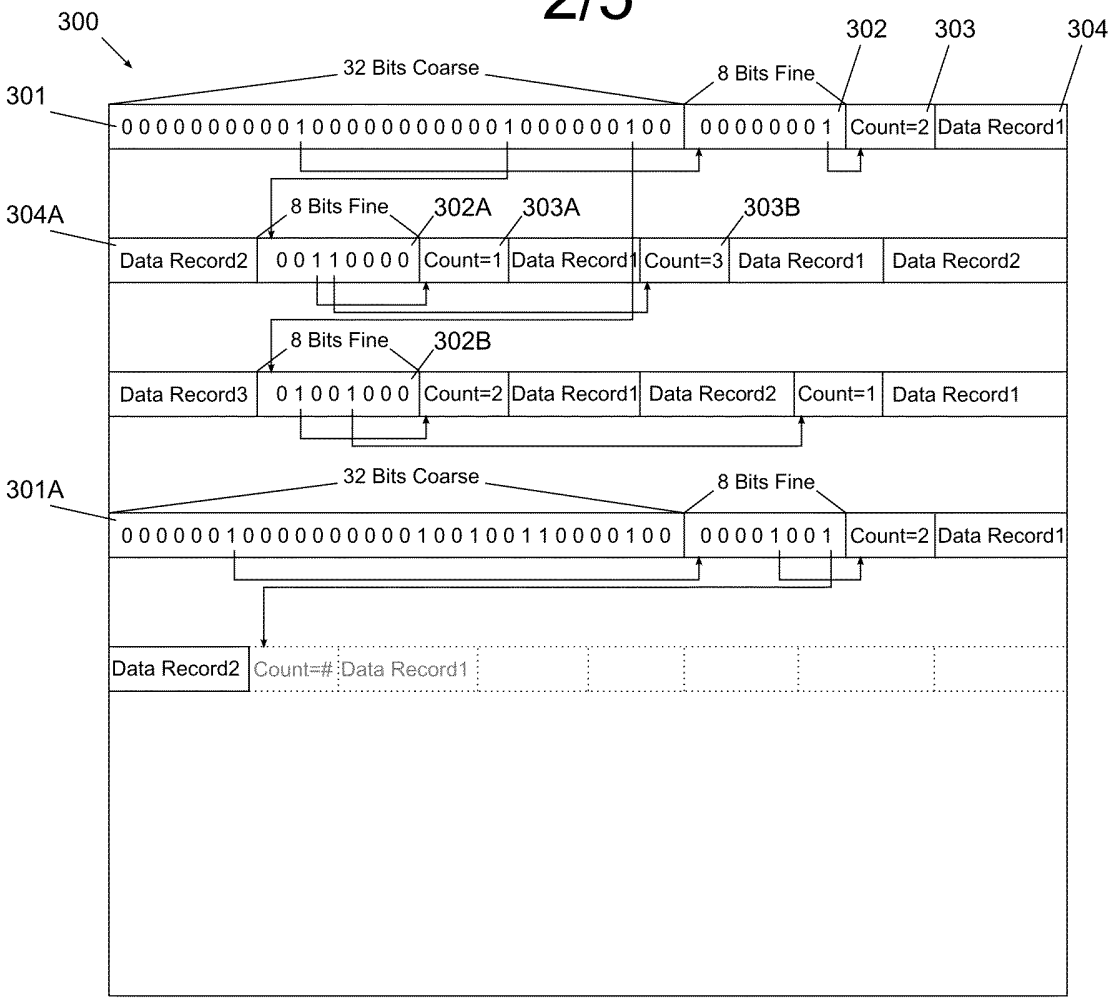

300
301
32 Bits Coarse
8 Bits Fine
302
303
304
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1
Count=2
Data Record1
304A
8 Bits Fine
302A
303A
303B
Data Record2
0 0 1 1 0 0 0 0
Count=1
Data Record1
Count=3
Data Record1
Data Record2
8 Bits Fine
302B
Data Record3
0 1 0 0 1 0 0 0
Count=2
Data Record1
Data Record2
Count=1
Data Record1
301A
32 Bits Coarse
8 Bits Fine
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 1 0 0
0 0 0 0 1 0 0 1
Count=2
Data Record1
Data Record2
Count=#
Data Record1

FIG. 3

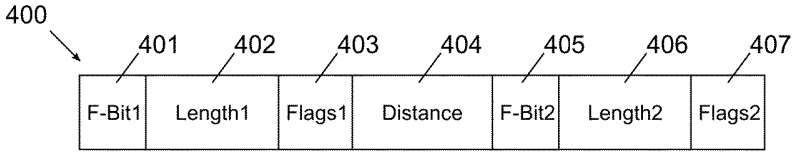

400
401
402
403
404
405
406
407
F-Bit1
Length1
Flags1
Distance
F-Bit2
Length2
Flags2

FIG. 4

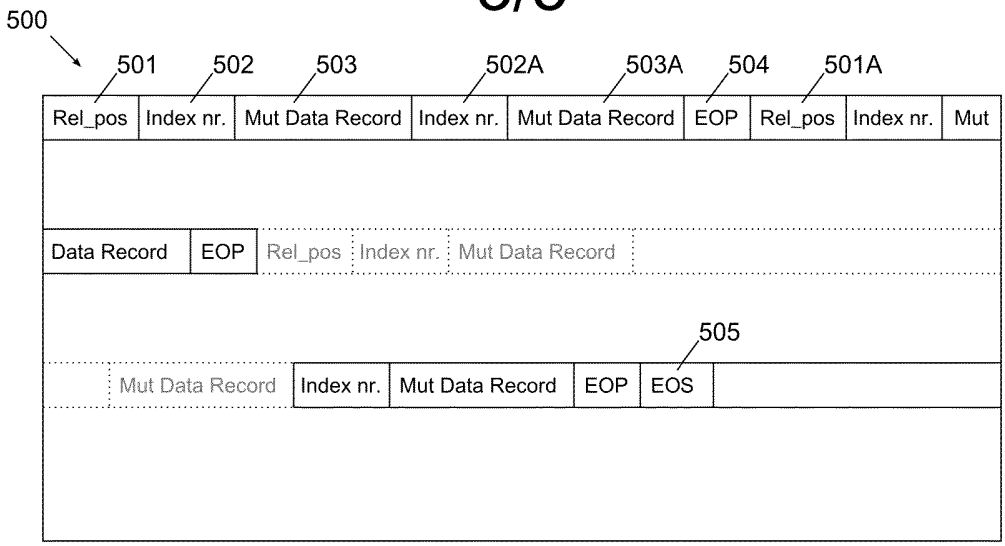


FIG. 5

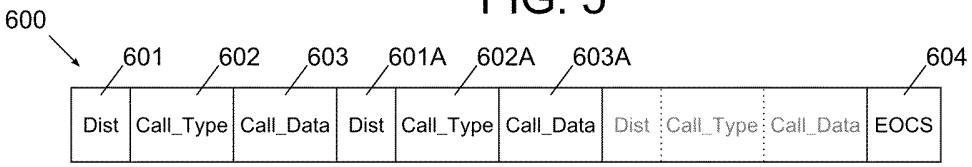


FIG. 6

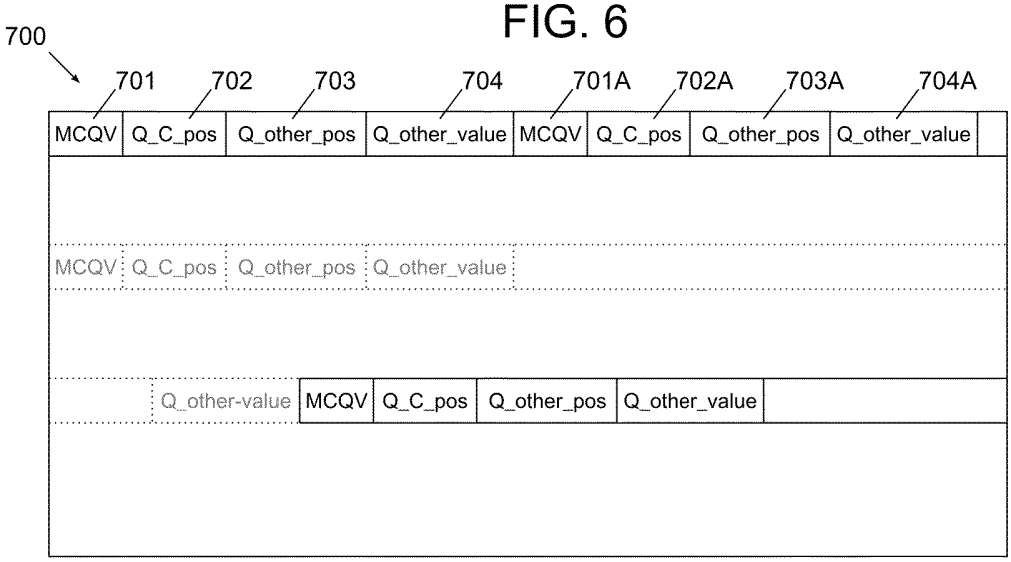


FIG. 7

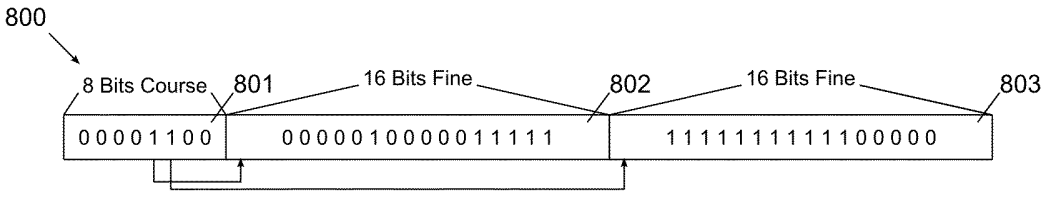


FIG. 8

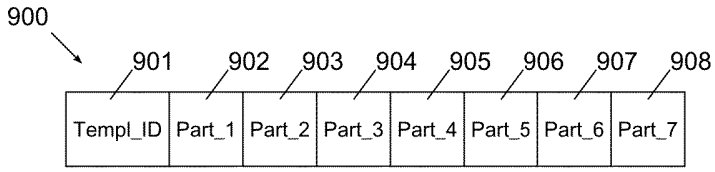


FIG. 9

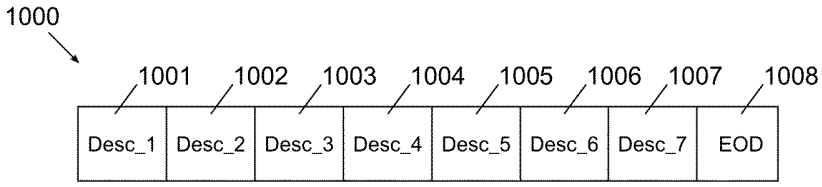


FIG. 10

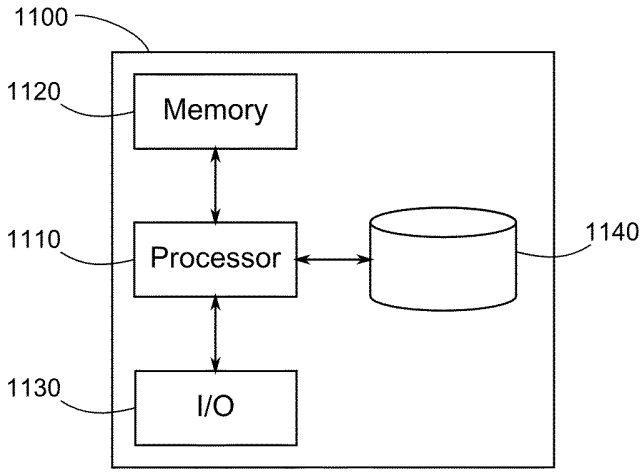


FIG. 11

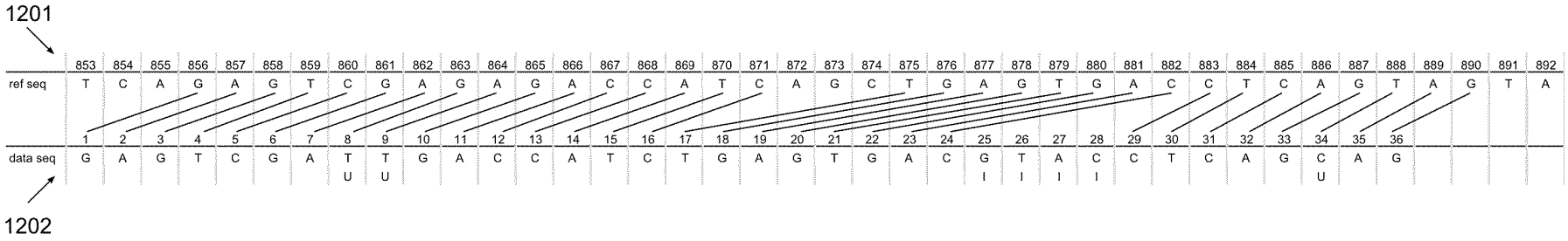

1201
853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892
ref seq T C A G A G T C G A G A G A C C A T C A G C T G A G T G A C C T C A G T A G T A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
data seq G A G T C G A T T G A C C A T C T G A G T G A C G T A C C T C A G C A G
U U I I I I U
1202

FIG. 12

SAMENWERKINGSVERDRAG (PCT)

RAPPORT BETREFFENDE NIEUWHEIDSONDERZOEK VAN INTERNATIONAAL TYPE

IDENTIFICATIE VAN DE NATIONALE AANVRAGE	KENMERK VAN DE AANVRAGER OF VAN DE GEMACHTIGDE
	0053P-NL
Nederlands aanvraag nr. **2012222**	Indieningsdatum **06-02-2014**
	Ingeroepen voorrangsdatum
Aanvrager (Naam) **Genalice B.V.**	
Datum van het verzoek voor een onderzoek van internationaal type **03-05-2014**	Door de Instantie voor Internationaal Onderzoek aan het verzoek voor een onderzoek van internationaal type toegekend nr. **SN 61938**

I. CLASSIFICATIE VAN HET ONDERWERP (bij toepassing van verschillende classificaties, alle classificatiesymbolen opgeven)

Volgens de internationale classificatie (IPC)

G06F17/30 **G06F19/22** **H03M7/30**

II. ONDERZOCHTE GEBIEDEN VAN DE TECHNIEK

Onderzochte minimumdocumentatie

Classificatiesysteem	Classificatiesymbolen
IPC	**G06F** **H03M**

Onderzochte andere documentatie dan de minimum documentatie, voor zover dergelijke documenten in de onderzochte gebieden zijn opgenomen

III.	☐ **GEEN ONDERZOEK MOGELIJK VOOR BEPAALDE CONCLUSIES**	(opmerkingen op aanvullingsblad)
IV.	☐ **GEBREK AAN EENHEID VAN UITVINDING**	(opmerkingen op aanvullingsblad)

ONDERZOEKSRAPPORT BETREFFENDE HET RESULTAAT VAN HET ONDERZOEK NAAR DE STAND VAN DE TECHNIEK VAN HET INTERNATIONALE TYPE

Nummer van het verzoek om een onderzoek naar de stand van de techniek
NL 2012222

A. CLASSIFICATIE VAN HET ONDERWERP
INV. G06F17/30 G06F19/22 H03M7/30
ADD.

Volgens de Internationale Classificatie van octrooien (IPC) of zowel volgens de nationale classificatie als volgens de IPC.

B. ONDERZOCHTE GEBIEDEN VAN DE TECHNIEK

Onderzochte miminum documentatie (classificatie gevolgd door classificatiesymbolen)
G06F H03M

Onderzochte andere documentatie dan de mimimum documentatie, voor dergelijke documenten, voor zover dergelijke documenten in de onderzochte gebieden zijn opgenomen

Tijdens het onderzoek geraadpleegde elektronische gegevensbestanden (naam van de gegevensbestanden en, waar uitvoerbaar, gebruikte trefwoorden)
EPO-Internal, WPI Data

C. VAN BELANG GEACHTE DOCUMENTEN

Categorie °	Geciteerde documenten, eventueel met aanduiding van speciaal van belang zijnde passages	Van belang voor conclusie nr.
X	US 2006/112264 A1 (AGARWAL RAMESH C [US] AGARWAL RAMESH CHANDRA [US]) 25 mei 2006 (2006-05-25) * alineas [0032], [0033], [0047], [0052], [0064], [0103], [0104], [0106]; figuur 8 *	1-30
X	US 2013/185267 A1 (GATEWOOD JOE M [US] ET AL) 18 juli 2013 (2013-07-18) * alineas [0003], [0007], [0010], [0042] - [0049], [0071], [0072], [0077], [0079], [0095], [0097] *	1-30
A	US 8 239 421 B1 (MARWAH VINEET [US] ET AL) 7 augustus 2012 (2012-08-07) * samenvatting * * kolom 6, regel 6 - regel 24 *	1-30

☐ Verdere documenten worden vermeld in het vervolg van vak C.

☒ Leden van dezelfde octrooifamilie zijn vermeld in een bijlage

° Speciale categorieën van aangehaalde documenten

"A" niet tot de categorie X of Y behorende literatuur die de stand van de techniek beschrijft

"D" in de octrooiaanvrage vermeld

"E" eerdere octrooi(aanvrage), gepubliceerd op of na de indieningsdatum, waarin dezelfde uitvinding wordt beschreven

"L" om andere redenen vermelde literatuur

"O" niet-schriftelijke stand van de techniek

"P" tussen de voorrangsdatum en de indieningsdatum gepubliceerde literatuur

"T" na de indieningsdatum of de voorrangsdatum gepubliceerde literatuur die niet bezwarend is voor de octrooiaanvrage, maar wordt vermeld ter verheldering van de theorie of het principe dat ten grondslag ligt aan de uitvinding

"X" de conclusie wordt als niet nieuw of niet inventief beschouwd ten opzichte van deze literatuur

"Y" de conclusie wordt als niet inventief beschouwd ten opzichte van de combinatie van deze literatuur met andere geciteerde literatuur van dezelfde categorie, waarbij de combinatie voor de vakman voor de hand liggend wordt geacht

"&" lid van dezelfde octrooifamilie of overeenkomstige octrooipublicatie

Datum waarop het onderzoek naar de stand van de techniek van internationaal type werd voltooid	Verzenddatum van het rapport van het onderzoek naar de stand van de techniek van internationaal type
30 september 2014	

Naam en adres van de instantie	De bevoegde ambtenaar
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Haffner, Ronald

Formulier PCT/ISA/201 (tweede blad) (Januari 2004)

ONDERZOEKSRAPPORT BETREFFENDE HET RESULTAAT VAN HET ONDERZOEK NAAR DE STAND VAN DE TECHNIEK VAN HET INTERNATIONALE TYPE

Informatie over leden van dezelfde octrooifamilie

Nummer van het verzoek om een onderzoek naar de stand van de techniek
NL 2012222

In het rapport genoemd octrooigeschrift	Datum van publicatie	Overeenkomend(e) geschrift(en)	Datum van publicatie
US 2006112264 A1	25-05-2006	GEEN	
US 2013185267 A1	18-07-2013	US 2008077607 A1	27-03-2008
		US 2013185267 A1	18-07-2013
		WO 2006052242 A1	18-05-2006
US 8239421 B1	07-08-2012	GEEN	

WRITTEN OPINION

File No.	Filing date *(day/month/year)*	Priority date *(day/month/year)*	Application No.
SN61938	06.02.2014		NL2012222

International Patent Classification (IPC)
INV. G06F17/30 G06F19/22 H03M7/30

Applicant
Genalice B.V.

This opinion contains indications relating to the following items:

- ☒ Box No. I Basis of the opinion
- ☐ Box No. II Priority
- ☐ Box No. III Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
- ☐ Box No. IV Lack of unity of invention
- ☒ Box No. V Reasoned statement with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
- ☐ Box No. VI Certain documents cited
- ☒ Box No. VII Certain defects in the application
- ☒ Box No. VIII Certain observations on the application

	Examiner
	Haffner, Ronald

Application number

WRITTEN OPINION

NL2012222

Box No. I Basis of this opinion

1. This opinion has been established on the basis of the latest set of claims filed before the start of the search.

2. With regard to any **nucleotide and/or amino acid sequence** disclosed in the application and necessary to the claimed invention, this opinion has been established on the basis of:

   a. type of material:

   ☐ a sequence listing

   ☐ table(s) related to the sequence listing

   b. format of material:

   ☐ on paper

   ☐ in electronic form

   c. time of filing/furnishing:

   ☐ contained in the application as filed.

   ☐ filed together with the application in electronic form.

   ☐ furnished subsequently for the purposes of search.

3. ☐ In addition, in the case that more than one version or copy of a sequence listing and/or table relating thereto has been filed or furnished, the required statements that the information in the subsequent or additional copies is identical to that in the application as filed or does not go beyond the application as filed, as appropriate, were furnished.

4. Additional comments:

Box No. V Reasoned statement with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1. Statement

Novelty	Yes:	Claims	2-18, 20-26
	No:	Claims	1, 19, 27-30
Inventive step	Yes:	Claims	
	No:	Claims	1-30
Industrial applicability	Yes:	Claims	1-30
	No:	Claims	

2. Citations and explanations

see separate sheet

Application number
NL2012222

WRITTEN OPINION

Box No. VII Certain defects in the application

see separate sheet

Box No. VIII Certain observations on the application

see separate sheet

Re Item V

Reasoned statement with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1 Reference is made to the following documents:

D1 US 2006/112264 A1 (AGARWAL RAMESH C [US] AGARWAL RAMESH CHANDRA [US]) 25 mei 2006 (2006-05-25)

D2 US 2013/185267 A1 (GATEWOOD JOE M [US] ET AL) 18 juli 2013 (2013-07-18)

D3 US 8 239 421 B1 (MARWAH VINEET [US] ET AL) 7 augustus 2012 (2012-08-07)

2 The present application does not meet the criteria of patentability, because the subject-matter of claim 1, 19, 27-30 is not new.

2.1 Document D1 discloses the following subject-matter of claim 1 (references in parenthesis refer to D1):

I. **Een op een computer geïmplementeerde werkwijze voor het opslaan in een opslaggegevensstructuur van een veelheid van sequenties die zijn uitgelijnd met een referentiegegevensstructuur** (§32, §33, §47, §64, §103),

II. **de referentiegegevensstructuur beschrijft referentiedata als een aaneengesloten referentiesequentie waarbij elk element van de referentiesequentie een positienummer en elementwaarde heeft** (§33, §52, §64),

III. **een sequentie omvat een aantal elementen met elementwaarden die overeenkomen met een deel van de referentiesequentie** (§32, §33),

IV. **het deel van de referentiesequentie heeft een overeenkomstige referentiepositie** (§32, §33), **waarbij de werkwijze omvat:**

V. **- het opslaan van een eerste parameter in een headersectie van de gegevensstructuur, de eerste parameter identificeert de referentiegegevensstructuur** (§103, §104, §106; Fig. 8);

VI. **- het opslaan van de referentieposities en een veelheid van eerste opslagsectierecords voor de veelheid van sequenties in een eerste opslagsectie van de gegevensopslag** (§103, §104, §106; Fig. 8),

VII. **een eerste opslagsectie record is gekoppeld aan tenminste een sequentie die heeft een overeenkomstige referentiepositie** (§103, §104, §106; Fig. 8) **en**

VIII. **de eerste opslagsectie omvat verder een lengteveld met een waarde die het mogelijk maakt het aantal elementen van de ten minste ene sequentie te bepalen** (§103, §104, §106; Fig. 8).

These are all the features of claim 1 whose subject-matter is, hence, not novel.

2.2 Claims 19, 27-30 are claims corresponding to method claim 1 and their subject-matter is, hence, not novel for the same reasoning.

2.3 Alternatively, document D2 can be considered the closest prior art to the subject-matter of claim 1. D2 discloses the following features of claim 1 (references in parenthesis refer to D2, crossed-out features not disclosed):

I. **Een op een computer geïmplementeerde werkwijze voor het opslaan in een opslaggegevensstructuur van een veelheid van sequenties die zijn uitgelijnd met een referentiegegevensstructuur** (§3, §7, §10, §48; §95, Fig. 15),

II. **de referentiegegevensstructuur beschrijft referentiedata als een aaneengesloten referentiesequentie waarbij elk element van de referentiesequentie een positienummer en elementwaarde heeft** (Fig. 2, §49),

III. **een sequentie omvat een aantal elementen met elementwaarden die overeenkomen met een deel van de referentiesequentie** (Fig.2, §49),

IV. **het deel van de referentiesequentie heeft een overeenkomstige referentiepositie** (Fig.2, §49), **waarbij de werkwijze omvat:**

V. **- het opslaan van een eerste parameter in ~~een headersectie van~~ de gegevensstructuur, de eerste parameter identificeert de referentiegegevensstructuur** (§71, Fig. 5; §95, Fig. 15; §97);

VI. **- het opslaan van de referentieposities en een veelheid van eerste opslagsectierecords voor de veelheid van sequenties in ~~een eerste opslagsectie van~~ de gegevensopslag** (§72; §79; §95, Fig. 15; §97),

VII. **een eerste opslagsectie record is gekoppeld aan tenminste een sequentie die heeft een overeenkomstige referentiepositie** (§49; §77, Fig. 8; §79; §95, Fig. 15; §97) **en**

VIII. **de eerste opslagsectie omvat verder een lengteveld met een waarde die het mogelijk maakt het aantal elementen van de ten minste ene sequentie te bepalen** (§49; §77, Fig. 8; §79; §97).

Claim 1 thus differs from D2 in that (represented by the crossed-out features above) the first parameter is saved in the header of the data structure and further data is saved in a first section of the storage.

The technical effect of these features is that it specifies the details of where to store the data. Since D2 does give any details to where to store the data, the skilled person would encounter the technical problem to be solved of where to save the data in the data storage.

The solution proposed in claim 1 of the present application cannot be considered as involving an inventive step for the following reason:

- Where to store the data in a data structure and/or the data storage is just a mere choice of the skilled person, made according to circumstances, without inventive skill or surprising technical effect (see e.g. D1 and D3 and the passages indicated in the search report).

3 The dependent claims does not appear to contain any additional features which, in combination with the features of any claim to which it refers, meet the requirements of inventive step, the reasons being as follows:

- They all appear to relate to minor implementation details of data compression using delta encoding which are all commonly known to the skilled person, who would apply such features according to circumstances, as a mere choice, without inventive skill or surprising technical effect

Re Item VII

Certain defects in the application

4 The independent claims are not in the two-part form.

5 The relevant background art disclosed in the documents D1, D2 is not mentioned in the description, nor are these documents identified therein.

Re Item VIII

Certain observations on the application

6 The independent claims are directed to storing in a storage data structure a multitude of sequences that have been aligned with a reference data structure, the reference data structure describes reference data as one continuous reference sequence wherein each element of the reference sequences has a position number and element value, a sequence comprises a number of elements with element values that matches a part of the reference sequence, the part of the reference sequence having a corresponding reference position.

The very general definition of "sequence" includes any kind of sequence in the scope of the claims. However, on the one hand, the claims cannot be understood for any kind of sequence, as, for example, there is no clearly defined meaning for such sequences that have been aligned with the reference data structure. On the other hand, the description does not provide support for any kind of sequences, but only provides support, and gives meaning to the wording of the claim, in the field of genetic related data-processing. As a result, the scope of the claims is not clear and they are not supported by the description.