(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년02월07일
(11) 등록번호 10-1359764
(24) 등록일자 2014년01월29일

(51) 국제특허분류(Int. Cl.)
G06F 19/00 (2011.01) *C12Q 1/68* (2006.01)
(21) 출원번호 10-2013-0071423
(22) 출원일자 2013년06월21일
심사청구일자 2013년06월21일
(56) 선행기술조사문헌
한국정보과학회, 2002*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
인하대학교 산학협력단
인천광역시 남구 인하로 100, 인하대학교 (용현동)
(72) 발명자
심정섭
인천 남동구 문화로 161, 대우재LH 102동 502호 (구월동)
정주희
서울 동작구 노량진로6가길 13, 2층 (노량진동)
김영호
서울 구로구 경인로 176, 701호 (오류동, 시티월드)
(74) 대리인
양성보

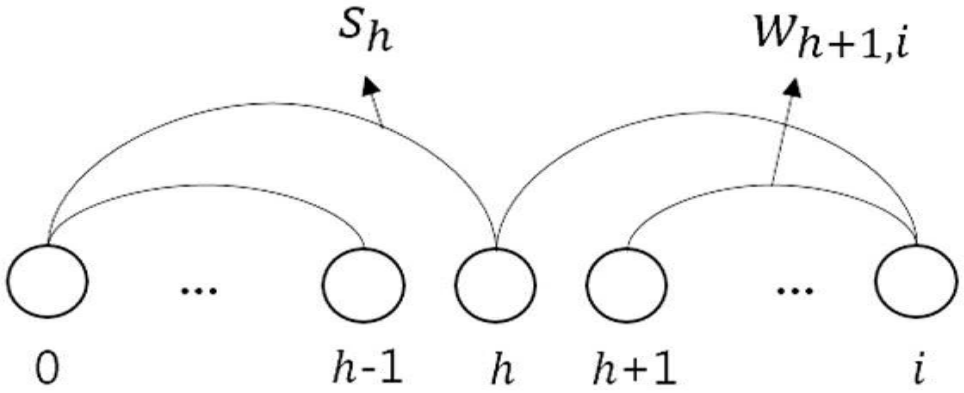
전체 청구항 수 : 총 1 항 심사관 : 박재용

(54) 발명의 명칭 **DNA 서열 분석을 위한 거리합 기반 문자열의 근사주기 계산 방법**

(57) 요 약

DNA 서열 분석을 위한 거리합 기반 문자열의 근사주기 계산 방법이 개시된다. 거리합 기반 근사주기거리 계산 방법은, 제1 문자열과 제2 문자열이 주어질 때, 문자열 매칭(character string matching)을 판단하기 위한 거리 함수(distance function)를 이용하여 상기 제1 문자열과 상기 제2 문자열의 부분문자열 사이의 거리를 계산하는 (1) 단계; 및 상기 제1 문자열과 상기 제2 문자열의 부분문자열 사이의 거리를 이용하여 상기 제1 문자열의 상기 제2 문자열에 대한 근사주기거리(approximate period distance)를 계산하는 (2) 단계를 포함할 수 있다.

대 표 도 - 도4



이 발명을 지원한 국가연구개발사업
과제고유번호 1345189451
부처명 교육부
연구사업명 핵심연구(개인)
연구과제명 차세대 유전체서열분석을 위한 근사문자열 알고리즘 개발
기 여 율 7/10
주관기관 인하대학교 산학협력단
연구기간 2013.05.01 ~ 2014.04.30
이 발명을 지원한 국가연구개발사업
과제고유번호 1415122511
부처명 지식경제부
연구사업명 디지털콘텐츠원천기술개발
연구과제명 상황대응형 분산 트랜스코딩기술을 이용한 저전력 고성능 멀티미디어 콘텐츠 관리기술 개발
기 여 율 3/10
주관기관 인하대학교 산학협력단
연구기간 2013.06.01 ~ 2014.05.31

특허청구의 범위

청구항 1

삭제

청구항 2

제1 문자열과 제2 문자열이 주어질 때, 문자열 매칭(character string matching)을 판단하기 위한 거리함수(distance function)를 이용하여 상기 제1 문자열과 상기 제2 문자열의 부분문자열 사이의 거리를 계산하는 (1) 단계; 및

상기 제1 문자열과 상기 제2 문자열의 부분문자열 사이의 거리를 이용하여 상기 제1 문자열의 상기 제2 문자열에 대한 근사주기거리(approximate period distance)를 계산하는 (2) 단계

를 포함하고,

상기 제1 문자열이 길이가 m인 문자열 p이고, 상기 제2 문자열이 길이가 n인 문자열 x이며, 상기 문자열 x가 서로 겹치지 않는 블록주기인 p_i로 나누어 질 때($x=p_1p_2\cdots p_r(1\leq i\leq r)$),

$$\sum_{i=1}^{r-1}\delta(p_i,p)+\delta(p_r,p')\leq s$$

(여기서, δ(A,B)는 문자열 A를 B로 변환하기 위한 거리함수, r은 블록주기의 수, p'는 문자열 p의 접두사를 나타낸다.)의 조건을 만족하면 상기 문자열 p를 상기 문자열 x에 대해 거리합(distance sum)이 s인 근사주기로 정의하며,

상기 문자열 x의 블록주기 각각에 대한 거리합 중 최소값을 상기 문자열 p의 상기 문자열 x에 대한 거리합 기반의 상기 근사주기거리로 정의하고,

상기 (1) 단계는,

상기 문자열 p와 상기 문자열 x의 부분문자열 $x[i\cdots j](1\leq i\leq j\leq n)$ 사이의 거리(w_{ij}), 및 상기 문자열 p의 접두사 p'와 상기 문자열 x의 마지막 문자를 포함한 문자열 $x[i\cdots n]$ 사이의 거리 중 최소값(w_{in})을 계산하고,

상기 (2) 단계는,

상기 블록주기가 나누어지는 위치가 변경되도록 상기 문자열 x의 길이를 1부터 n까지 1씩 늘려가면서 수학식 1을 통해 최소값을 가지는 상기 근사주기거리를 계산하는 것

을 특징으로 하는 거리합 기반 근사주기거리 계산 방법.

수학식 1:

$$s_i=\min_{0\leq h<i}(s_h+w_{h+1,i})$$

(여기서, $s_i(0\leq i\leq n, s_0=0)$는 문자열 p의 문자열 $x[1\cdots i](1\leq i\leq n)$에 대한 근사주기거리, s_h는 문자열 p의 문자열 $x[1\cdots h]$에 대한 근사주기거리, $w_{h+1,i}$는 문자열 p와 문자열 $x[h+1\cdots i](1\leq h+1\leq i\leq n)$ 사이의 거리이고, h는 거리 값 s_h와 $w_{h+1,i}$의 합이 최소가 되는 위치, h+1은 문자열 x의 마지막 블록주기의 시작 위치를 나타낸다.)

청구항 3

삭제

청구항 4

삭제

청구항 5

삭제

명 세 서

기 술 분 야

[0001] 본 발명의 실시예들은 두 문자열에서 거리합(distance sum) 기반의 근사주기거리(approximate period distance)를 계산하는 방법에 관한 것이다.

배 경 기 술

[0002] 문자열매칭(string matching) 알고리즘은 많은 분야에서 활발히 연구되고 있다. 특히, 주기(period)와 같은 반복문자열에 대한 연구는 데이터압축, 컴퓨터활용 음악분석, 바이오인포매틱스 등 다양한 분야에서 진행되고 있다.

[0003] 주기에 대한 정의는 다음과 같다. 길이가 각각 m과 n인 문자열 p와 x에 대해 p^k를 p가 k번 반복된 문자열이라 하고, p'를 p의 접두사라 할 때, $x=p^kp'(k\geq 1)$이면 p를 x의 주기라 한다. 예를 들어, x=abaabaab이면 aba가 x의 주기이다.

[0004] 바이오인포매틱스 분야에서 주기는 유전자 서열이 반복적으로 나타나는 종렬중복(tandem repeats)과 밀접한 관련이 있다. 인간 유전체의 10%는 이러한 종렬중복으로 이루어져 있다고 알려져 있다. 이러한 종렬중복에는 인간의 유전적 특성과 관련된 정보들을 담고 있어 유전체 분석에서 중요하게 다루어지고 있다. 한편, 유전체에서는 정확하게 일치하는 서열들만 반복되어 나타나는 것이 아니라, 어느 정도 다르지만 유사하게 반복되는 서열들도 자주 발생한다.

[0005] 이렇게 바이오인포매틱스 분야에서 발생하는 유사한 부분들의 반복에 대한 연구와 관련된 문제들은 근사문자열매칭(approximate character string matching) 알고리즘을 이용하여 해결할 수 있으며, 특히 종렬중복과 관련하여 근사적으로 반복되는 문자열에 대한 연구가 다양하게 진행되고 있다.

[0006] 예컨대, 논문 "J.S. Sim, C.S. Iliopoulos, K. Park, W.F. Smyth, 'Approximate periods of strings,' Theoretical Computer Science, 262, 557-568, 2001."에서는 반복되는 문자열 사이의 불일치를 허용한 근사주기에 대한 연구가 수행되어있다. 이때 문자열 사이에 나타나는 일치 정도를 오차(error) 또는 거리(distance)라 하며 이는 편집거리(edit distance), 가중편집거리(weighted edit distance), 해밍거리(Hamming distance)와 같은 거리함수를 이용하여 계산할 수 있다.

[0007] 기존의 근사주기거리를 찾는 문제는 오차가 골고루 분포했을 경우 정확한 척도를 제시할 수 있다. 한편, 대부분의 서열에서는 오차가 적지만, 특정 지역에서만 오차가 많이 발생한 경우, 바람직하지 않은 척도를 보일 수 있다.

[0008] 도 1은 기존의 근사주기거리가 부적합한 예를 도시한 것이다. 도 1의 (a)는 근사주기거리가 1이고, 거리합 기반 근사주기거리가 3인 경우, 도 1의 (b)는 근사주기거리가 2이고, 거리합 기반 근사주기거리가 2인 경우를 도시하고 있다. 도 1에서 (a)와 (b)의 상황을 비교했을 때 (b)가 더 근사주기에 유사한 경우로 판단되지만 기존 방법으로는 오차가 더 큰 결과를 가져오게 된다.

[0009] 이러한 문제를 보완하기 위해 거리합 기반 근사주기거리를 찾는 문제가 필요하다. 거리합 기반 근사주기거리 문제는 전체 오차의 수를 고려하여, 특정 지역에서만 많이 발생하는 오차에 영향을 기존의 근사주기거리를 찾는 문제보다 적게 받는다.

[0010] 따라서, 종렬중복을 찾는 새로운 척도로서 거리합 기반 근사주기거리를 찾는 문제가 중요한 연구 과제가 될 수 있다.

발명의 내용

해결하려는 과제

[0011] 본 발명에서는 거리합(distance sum)이라는 새로운 목적함수를 이용하여 거리합 기반의 근사주기와 근사주기거리를 찾는 문제를 정의하고 이를 해결하는 알고리즘을 제시한다.

[0012] 본 발명에서는 길이가 각각 m과 n인 두 문자열 p와 x가 주어졌을 때 p의 x에 대한 거리합 기반 근사주기거리를 가중편집거리에 대해 $O(n^2)$ 시간, 편집거리에 대해 O(mn) 시간, 해밍거리에 대해 O(n) 시간에 각각 계산할 수 있는 알고리즘을 제시한다.

과제의 해결 수단

[0013] 본 발명의 실시예에 따르면, 거리합 기반 근사주기거리 계산 방법은, 제1 문자열과 제2 문자열이 주어질 때, 문자열 매칭(character string matching)을 판단하기 위한 거리함수(distance function)를 이용하여 상기 제1 문자열과 상기 제2 문자열의 부분문자열 사이의 거리를 계산하는 (1) 단계; 및 상기 제1 문자열과 상기 제2 문자열의 부분문자열 사이의 거리를 이용하여 상기 제1 문자열의 상기 제2 문자열에 대한 근사주기거리(approximate period distance)를 계산하는 (2) 단계를 포함할 수 있다.

[0014] 일 측면에 따르면, 상기 제1 문자열이 길이가 m인 문자열 p이고, 상기 제2 문자열이 길이가 n인 문자열 x이며, 상기 문자열 x가 서로 겹치지 않는 블록주기인 p_i로 나누어 질 때($x=p_1p_2\cdots p_r(1\le i\le r)$), $\sum_{i=1}^{r-1}\delta(p_i,p)+\delta(p_r,p')\le s$ (여기서, δ(A,B)는 문자열 A를 B로 변환하기 위한 거리함수, r은 블록주기의 수, p'는 문자열 p의 접두사를 나타낸다.)의 조건을 만족하면 상기 문자열 p를 상기 문자열 x에 대해 거리합(distance sum)이 s인 근사주기로 정의하며, 상기 문자열 x의 블록주기 각각에 대한 거리합 중 최소값을 상기 문자열 p의 상기 문자열 x에 대한 거리합 기반의 상기 근사주기거리로 정의할 수 있다.

[0015] 다른 측면에 따르면, 상기 거리함수로는 해밍거리(Hamming distance), 편집거리(edit distance), 가중편집거리(weighted edit distance) 중 어느 하나가 이용될 수 있다.

[0016] 또 다른 측면에 따르면, 상기 (1) 단계는, 상기 제1 문자열이 길이가 m인 문자열 p이고, 상기 제2 문자열이 길이가 n인 문자열 x이며, 상기 문자열 x가 서로 겹치지 않는 블록주기인 p_i로 나누어 질 때($x=p_1p_2\cdots p_r(1\le i\le r)$), 상기 문자열 p와 상기 문자열 x의 부분문자열 $x[i\cdots j](1\le i\le j\le n)$ 사이의 거리(w_{ij}), 및 상기 문자열 p의 접두사 p'와 상기 문자열 x의 마지막 문자를 포함한 문자열 $x[i\cdots n]$ 사이의 거리 중 최소값(w_{in})을 계산할 수 있다.

[0017] 또 다른 측면에 따르면, 상기 (2) 단계는, 상기 제1 문자열이 길이가 m인 문자열 p이고, 상기 제2 문자열이 길이가 n인 문자열 x이며, 상기 문자열 x가 서로 겹치지 않는 블록주기인 p_i로 나누어 질 때($x=p_1p_2\cdots p_r(1\le i\le r)$), 상기 문자열 x의 길이를 1씩 늘려가면서 수학식 1을 통해 최소값을 가지는 상기 근사주기거리를 계산할 수 있다.

[0018] 수학식 1:

[0019] $$s_i=\min_{0\le h<i}(s_h+w_{h+1,i})$$

[0020] (여기서, $s_i(0\le i\le n, s_0=0)$는 문자열 p의 문자열 $x[1\cdots i](1\le i\le n)$에 대한 근사주기거리, s_h는 문자열 p의 문자열 $x[1\cdots h]$에 대한 근사주기거리, $w_{h+1,i}$는 문자열 p와 문자열 $x[h+1\cdots i](1\le h+1\le i\le n)$ 사이의 거리이고, h는 거리 값 s_h와 $w_{h+1,i}$의 합이 최소가 되는 위치, h+1은 문자열 x의 마지막 블록주기의 시작 위치를 나타낸다.)

발명의 효과

[0021] 본 발명의 실시예에 따르면, 거리합이라는 새로운 목적함수를 이용하여 거리합 기반의 근사주기와 근사주기거리를 찾는 문제를 정의함으로써 해밍거리일 때 O(n) 시간, 편집거리일 때 O(mn) 시간, 가중편집거리일 때, $O(n^2)$ 시간에 거리합 기반 근사주기거리를 효율적으로 계산할 수 있다.

도면의 간단한 설명

[0022] 도 1은 기존의 근사주기거리가 부적합한 예를 도시한 것이다.

도 2는 메트릭인 비용행렬에 대한 예를 도시한 것이다.

도 3은 문자열 abab와 bbaba의 편집거리를 계산하기 위한 D테이블을 도시한 것이다.

도 4는 본 발명의 일 실시예에 있어서, 거리합 기반 근사주기거리를 계산하는 방법을 설명하기 위한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0023] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0024] 본 발명은 거리합(distance sum)이라는 새로운 목적함수를 이용하여 거리합 기반의 근사주기와 근사주기거리를 찾는 문제를 정의하고 이를 해결하는 알고리즘에 관한 것으로, 두 문자열이 주어졌을 때 한 문자열의 다른 문자열에 대한 거리합 기반 근사주기거리를 계산하는 알고리즘을 제공할 수 있다.

[0025] 본 실시예들은 데이터압축, 컴퓨터활용 음악분석, 바이오인포매틱스 등 다양한 분야에서 적용될 수 있다.

[0026] 본 발명의 알고리즘을 위한 선행 연구에 대해 먼저 설명한다.

[0027] 거리함수

[0028] 문자열은 알파벳 집합 ∑에 속한 0개 이상의 문자들의 나열이다. 문자열 A의 길이는 |A|로 나타내며, A[i](1≤i≤|A|)는 A의 i번째 문자를 나타낸다. 또한, A의 A[i]A[i+1]··A[j]는 A[i··j]로 표기한다. 그리고, Δ는 공백(empty) 문자를 나타낸다.

[0029] 알파벳 집합 ∑에 대해 길이가 각각 m, n인 두 문자열 A, B가 주어졌을 때, 근사문자열매칭 알고리즘은 오차를 측정하기 위한 척도로서 거리함수 δ(A,B)를 이용할 수 있다. 이때, 거리함수 δ(A,B)는 A를 B로 변환하는데 필요한 최소 비용을 나타내며, 대표적인 거리함수로는 해밍거리, 편집거리, 가중편집거리 등이 있다.

[0030] 해밍거리는 |A|=|B|일 때 A를 B로 변환하는데 필요한 최소 교체연산의 수이다. 그리고, 편집거리는 A를 B로 변환하는데 필요한 최소 편집연산의 수로 정의된다. 이때 편집연산은 삽입(insertion), 삭제(deletion), 교체(change) 연산으로 구성된다. 마지막으로, 가중편집거리는 모든 문자 쌍에 대해 삽입, 삭제, 교체 비용을 나타낸 비용행렬(penalty matrix)이 주어졌을 때, 한 문자열이 다른 문자열로 변환하는데 필요한 최소 비용이다. 만약 모든 문자 a,b,c∈∑∪{Δ}에 대해 다음 네 가지 조건을 만족하면 비용행렬을 메트릭(metric)이라 한다.

[0031]
- $\delta(a,b) \geq 0$
- $\delta(a,b) = \delta(b,a)$
- $\delta(a,a) = 0$
- $\delta(a,c) \leq \delta(a,b) + \delta(b,c)$ (삼각부등식)

[0032] 도 2는 메트릭인 비용행렬에 대한 예이다. 가중편집거리는 도 2와 같이 비용행렬이 주어졌을 때 비용행렬에 저장된 값들을 참조하여 두 문자열 사이의 거리를 계산한다. 이때, 편집거리는 각 문자쌍 사이의 거리가 모두 1인 경우이다.

[0033] D테이블

[0034] 두 문자열 A와 B 가 주어졌을 때 잘 알려진 동적프로그래밍 기법을 이용하여 편집거리 및 가중편집거리를 계산할 수 있다. 이때 계산된 (|A|+1)×(|B|+1) 크기의 테이블을 D테이블이라 하자.

[0035] D테이블을 초기화 하는 방법은 수학식 1과 같다.

수학식 1

[0036]
$$\begin{cases} D[0,0] = 0, \\ D[i,0] = D[i-1,0] + d(A[i], \Delta)\ (1 \leq i \leq |A|), \\ D[0,j] = D[0,j-1] + d(\Delta, B[j])\ (1 \leq j \leq |B|) \end{cases}$$

[0037] 각 D[i,j](1≤i≤|A|, 1≤j≤|B|)는 수학식 2에 의해 계산될 수 있다.

수학식 2

[0038]
$$D[i,j]=\min\begin{pmatrix} D[i-1,j]+d(A[i],\Delta), \\ D[i,j-1]+d(\Delta,B[j]), \\ D[i-1,j-1]+d(A[i],B[j]) \end{pmatrix}$$

[0039] D[i,j]는 A[1··i]를 B[1··j]로 변환하는데 필요한 최소의 비용을 나타낸다. 따라서, D[|A|,j](1≤j≤|B|)에 저장된 값은 A와 B[1··j]의 편집거리 혹은 가중편집거리이다. 즉, A와 B의 접미사의 편집거리 혹은 가중편집거리를 계산하기 위해서 D테이블을 새로 계산할 필요가 없다. 그리고, D[|A|,|B|]가 A와 B의 편집거리 또는 가중편집거리를 의미한다.

[0040] 예를 들어, 도 3은 문자열 abab와 bbaba의 편집거리를 계산하기 위한 D테이블을 도시한 것이다. D[4,4]의 1은 A와 B[1··4]의 편집거리이며, D[4,5]의 2는 A와 B의 편집거리이다. 수학식 1과 수학식 2에 의해 각 D[i,j]는 O(1) 시간에 계산되므로 D테이블은 O(|A||B|) 시간에 계산된다.

[0041] 이하에서는 본 발명에 따른 거리합 기반 문자열의 최소 근사주기거리 찾기 알고리즘을 구체적으로 설명하기로 한다.

[0042] 거리합 기반 근사주기와 알고리즘

[0043] 본 발명에서 제시하는 거리합 기반 근사주기는 다음과 같이 정의한다.

[0044] 정의 1. 거리합 기반 근사주기와 근사주기거리

[0045] x가 $x=p_1p_2\cdots p_r(p_i\neq\Delta, 1\leq i\leq r)$과 같이 p_i로 나누어 질 수 있고 p'을 p의 접두사라고 하자. 이때, $\sum_{i=1}^{r-1}\delta(p_i,p)+\delta(p_r,p')\leq s$ 이면 p를 x의 거리합 기반 s-근사거리 또는 거리합이 s인 근사주기라고 정의한다. 이때, 각 p_i를 주기블록이라고 하며 s중 최소값을 p와 x의 거리합 기반 근사주기거리(이하, '근사주기거리'라 약칭함)라고 한다.

[0046] 근사주기거리를 찾는 문제를 다음과 같이 정의할 수 있다.

[0047] 문제1. 거리합 기반 문자열의 근사주기거리 찾기

[0048] 입력: 길이가 각각 m과 n인 문자열 p와 x, 거리함수 δ

[0049] 출력: p의 x에 대한 거리합 기반 최소 근사주기거리 s

[0050] 위의 거리합 기반 근사주기거리를 찾는 문제는 기존 문자열의 근사거리를 찾는 알고리즘을 변형하여 해결할 수 있다. 본 발명에서의 알고리즘은 p와 x의 부분문자열 사이의 거리를 계산하는 단계와 실제 최소근사거리 s를 계산하는 단계로 구성될 수 있다.

[0051] 단계1: p와 x의 부분문자열 사이의 거리 계산

[0052] w_{ij}를 p와 x의 부분문자열 x[i··j](1≤i≤j≤n) 사이의 거리, w_{in}을 p의 접두사들과 x[i··n] 사이의 거리의 최소값으로 정의한다. 이 값들을 계산하기 위해 p와 x의 모든 접미사들 사이의 동적프로그래밍 테이블인 D테이블을 생성한다. 생성된 p와 x[i··n] 사이의 D테이블에서 D[m,j]가 w_{ij}를 나타낸다. 또한, w_{in}은 n번째 열에서 최소값이 된다. 단계1에서 계산된 값들은 단계2에서 근사주기거리를 계산하는데 사용될 수 있다.

[0053] 단계2: 거리합 기반 근사주기거리 계산

[0054] 단계2에서는 p의 x에 대한 거리합 기반 근사주기거리 s를 동적프로그래밍 기법을 이용하여 계산할 수 있다. 총 n번의 계산 과정으로 이루어지며 각 계산 과정마다 s_i를 계산한다. s_i는 p의 x[1··i](1≤i≤n)에 대한 근사주기거리를 나타낸다. s_0=0으로 초기화 하고 i=1부터 n까지 차례로 수학식 3을 이용하여 s_i값을 계산할 수 있다.

수학식 3

[0055] $$s_i = \min_{0 \le h < i}(s_h + w_{h+1,i})$$

[0056] 여기서, $w_{h+1,i}$는 문자열 p와 x[h+1··i](1≤h+1≤i≤n) 사이의 거리이고, s_h는 p의 x[1··h]에 대한 근사주기거리를 의미한다. 다시 말해, 본 실시예의 알고리즘은 x의 길이를 처음부터 1(한문자)씩 늘려가면서 p와 x의 근사주기거리를 찾게 되는데, s_h는 p의 x[1··h]에 대한 거리합 기반 근사주기거리를 뜻하며, 결국 s_n이 본 알고리즘이 구하는 근사주기거리가 되는 것이다. 한 문자를 늘릴 때마다 마지막 주기를 블록의 길이가 달라질 수 있으므로 모든 가능성에 대해 다 비교해보기 위해서 수학식 3을 바탕으로 최소값을 만들어 내는 위치 h를 찾는다. 한편, 근사주기는 마지막 부분이 p와 정확히 배치되지 않아도 무관하다. p의 일부분, 즉 p의 접두사와 배치되면 되므로 끝부분은 w_{in}, 즉 어떤 p의 접두사 p'와 x[i··n]과의 거리를 이용할 수 있다. 이는 p와 x[i··j]와의 거리를 나타내는 w_{ij}의 정의와 같다.

[0057] 상기한 바와 같이, 문자열은 다양한 크기의 주기 블록으로 나눌 수 있으며 주기블록을 나누는 방법에 따라 계산되는 s_i가 달라진다. 그러므로, 수학식 3에서 0부터 i-1까지 주기블록을 나누는 위치를 변경하여 s_i를 계산하고 이 중 최소값을 찾아 근사주기거리를 계산한다. 도 4는 s_i까지의 거리합 기반 근사주기거리를 계산하는 방법을 도시한 것이다. 이때, 수학식 3에서 h값은 s_h와 $w_{h+1,i}$의 합이 최소가 되는 위치를 나타낸다.

[0058] 본 발명은 거리합이라는 새로운 목적함수를 이용하여 근사주기거리를 찾는 문제를 제시하고, 이 문제를 해결하기 위한 효율적인 알고리즘을 제안한다. 즉, 두 문자열과 거리함수가 주어졌을 때, 한 문자열이 다른 문자열의 거리합 기반 근사주기거리를 계산하는 알고리즘을 제시한다. 즉, p의 x[1··i](1≤i≤n)에 대한 근사주기거리를 찾고 주기블록들을 결정했을 때, h+1은 마지막 주기블록의 시작 위치를 나타낸다. 마지막으로 계산된 s_n이 p의 x에 대한 거리합 기반 근사주기거리가 된다.

[0059] 예를 들어, x가 GATGAATGAA일 때 근사주기거리는 3이 되고 이것은 x가 주기블록 GAT, GAAT, GAA로 나누어졌을 때 각 주기블록들과 p와의 거리의 합을 의미한다. 이는 주기블록을 나누는 여러 방법 중 최소 근사주기거리를 계산한다.

[0060] 거리함수로 편집거리, 가중편집거리, 해밍거리를 사용할 때 알고리즘의 수행시간은 다음과 같다.

[0061] 먼저 메트릭인 가중편집거리를 사용했을 때 알고리즘의 각 단계별 수행시간을 고려해 보자. 단계1에서 p와 x의 모든 접미사들 사이의 D테이블을 생성하므로 총 n개의 D테이블이 생성된다. 따라서, 단계1을 계산하는데 $O(mn^2)$ 시간이 필요하다. 그리고, 단계2에서 각 s_i를 계산할 때 h에 따라 O(n)번 비교를 하므로 $O(n^2)$ 시간이 필요하다. 따라서, 거리합 기반 근사주기거리를 계산하기 위해서는 총 $O(mn^2)$ 시간이 필요하다. P와 x[1··n] 사이의 가중편집거리를 알고 있을 때 p와 x[2··n]의 가중편집거리를 O(min{c(m+n),mn}) 시간에 계산하는 알고리즘(예컨대, 논문 "H. Hyyro K. Narisawa, S. Inenaga, 'Dynamic Edit Distance Table under a General Weighted Cost Function,' SOFSEM 2010: Theory and Practice of Computer Science, 5901, 515-527, 2010.")에서, 이때 c는 최대 가중치를 나타낸다. 만약, c가 상수라면 시간 복잡도는 O(m+n)이 되므로 단계1에서 n개의 D테이블을 생성하는 시간은 $O(n^2)$이 된다. 따라서, 가중편집거리를 사용했을 때 $O(n^2)$ 시간에 근사주기거리를 계산할 수 있다.

[0062] 두 번째로 편집거리를 사용할 때 알고리즘의 시간 복잡도를 계산해보자. 기존 근사주기거리를 찾는 알고리즘은 편집거리에 대해 다음과 같은 속성을 만족한다. X에 대해 무조건 길이가 m인 주기블록으로 나누면, 적어도 m-근사주기를 만족하기 때문에 오차가 m보다 큰 근사주기를 계산하는 것은 의미가 없다. 따라서, 길이가 2m 보다 긴 x의 부분문자열은 무조건 오차가 m보다 커지기 때문에, 근사주기거리를 계산할 때 p와 길이가 2m 이내인 x의

부분문자열들만 고려하면 된다. 이러한 속성은 거리합 기반 근사주기거리를 계산할 때는 만족하지 않을 수 있지만, 오차가 m보다 큰 주기블록이 나타난다면 반복되는 문자열이라고 보기 어렵다. 따라서, 본 발명의 알고리즘에서도 편집거리에 대해 근사주기거리를 계산할 때 길이가 2m을 넘지 않는 x의 부분문자열만 고려한다. 이때, 각 단계별 시간 복잡도는 다음과 같다. 단계1에서 크기가 (m+1)×(2m+1)인 D테이블이 n개 생성되므로 O(m^2n) 시간이 필요하다. 또한, 단계2에서 각 s_i를 계산할 때 길이가 2m을 넘지 않는 x의 부분문자열만 고려하므로 O(m) 번 비교한다. 총 n번의 계산 과정이 필요하므로 단계2는 O(mn) 시간에 계산된다. 따라서, 편집거리를 사용했을 때 O(m^2n) 시간에 근사주기거리를 계산할 수 있다. p와 x[1··2m]인 거리에 대한 해를 이용하여 p와 x[2··2m+1]의 거리에 대한 해를 O(m) 시간에 계산할 수 알고리즘(예컨대, 논문 "S.R. Kim and K. Park, 'A dynamic edit distance table,' Journal of Discrete Algorithms, 2, 303-312, 2004.")을 통해 단계1에서 필요한 시간을 O(mn)으로 개선할 수 있다. 그러므로 단계1을 O(mn) 시간에 해결할 수 있게 된다. 따라서, 편집거리를 사용했을 때 O(mn) 시간에 근사주기거리를 계산할 수 있다.

[0063] 세 번째로 해밍거리를 사용했을 때 알고리즘의 시간 복잡도를 계산해보자. 해밍거리는 비교하는 두 문자열 간의 길이가 같아야 한다. 따라서, 단순 문자비교를 이용하여 근사주기거리를 계산할 수 있으므로 해밍거리를 사용했을 때 O(n) 시간에 근사주기거리를 계산할 수 있다.

[0064] 본 발명에서 제시한 거리합 기반 근사주기의 정의는 기존의 근사거리를 이용했을 때 발생할 수 있는 오류를 보완할 수 있다. 즉, 다른 주기블록에 비해 하나의 주기블록에 많은 오차가 생겼을 때 이로 인해 주기블록을 제대로 분할하지 못할 수 있는 경우를 해결할 수 있다. 따라서, 본 발명에 따른 거리합 기반의 근사거리주기 계산 알고리즘은 바이오인포매틱스 분야에서 반복적인 서열에 대한 좀더 의미 있는 분석이 가능하다.

[0065] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0066] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(signal wave)에 영구적으로, 또는 일시적으로 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0067] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도

록 구성될 수 있으며, 그 역도 마찬가지이다.

[0068] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

[0069] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

도면1

x	ACAT	AAGT	ACCT
p^3	ACGT	ACGT	ACGT
error#	1	1	1

(a)

x	ACGT	AAGGT	ACGT
p^3	ACGT	ACGT	ACGT
error#	0	2	0

(b)

도면2

	a	b	c	Δ
a	0	2	1	2
b	2	0	2	1
c	1	2	0	1
Δ	2	1	1	0

도면3

	Δ	b	b	a	b	a
Δ	0	1	2	3	4	5
a	1	1	2	2	3	4
b	2	1	1	2	2	3
a	3	2	2	1	2	2
b	4	3	2	2	1	2

도면4