



(19) 대한민국특허청(KR)

(12) 등록특허공보(B1)

(45) 공고일자 2016년01월08일

(11) 등록번호 10-1583152

(24) 등록일자 2015년12월30일

(51) 국제특허분류(Int. Cl.)

G06F 17/00 (2006.01) G06F 17/20 (2006.01)

(21) 출원번호 10-2014-0116153

(22) 출원일자 2014년09월02일

심사청구일자 2014년09월02일

(56) 선행기술조사문헌

김영호, 정주희, 강대웅, 심정섭, “4-러시안 알고리즘 기반의 편집거리 병렬계산”, 정보처리학회논문지, 2013년 2월, 제2권, 제2호, 페이지 67-74.

(73) 특허권자

인하대학교 산학협력단

인천광역시 남구 인화로 100, 인하대학교 (용현동)

(72) 발명자

심정섭

인천광역시 남동구 구월동 161, 102동 502호(구월동, 인천대우재아파트)

김영호

서울 구로구 경인로 176, 701호 (오류동, 시티월드)

(뒷면에 계속)

(74) 대리인

양성보

전체 청구항 수 : 총 4 항

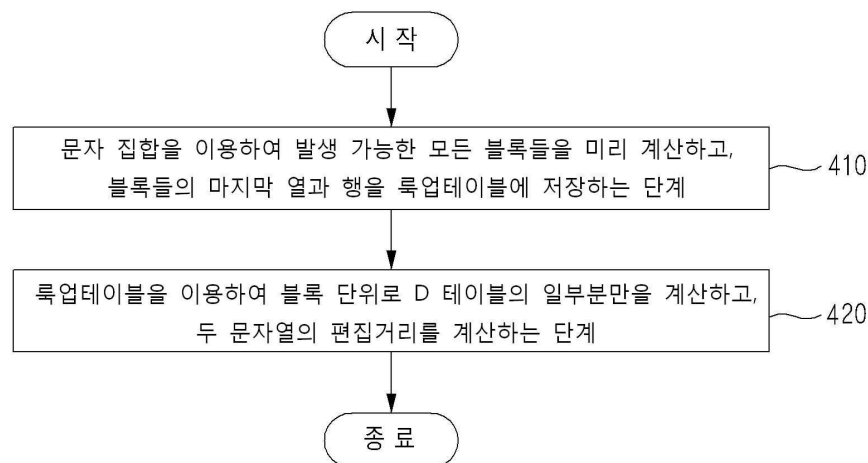
심사관 : 이석형

(54) 발명의 명칭 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 방법 및 시스템

(57) 요약

4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 방법 및 시스템이 제시된다. 본 발명에서 제안하는 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 방법은 문자 집합을 이용하여 발생 가능한 모든 블록들을 미리 계산하고, 블록들의 마지막 열과 행을 룩업테이블에 저장하는 단계, 상기 룩업테이블을 이용하여 블록 단위로 D 테이블의 일부분만을 계산하고, 두 문자열의 편집거리를 계산하는 단계를 포함할 수 있다.

대표도 - 도4



(72) 발명자

조석현

인천 남구 매소홀로 340, 105동 501호 (학익동, 학
익풍림아이원아파트)

허성찬

대전 동구 용운로 193, 140동 403호 (용운동, 주공
아파트1단지)

이 발명을 지원한 국가연구개발사업

과제고유번호 1345189451

부처명 미래창조과학부

연구관리전문기관 한국연구재단

연구사업명 핵심연구(개인)

연구과제명 차세대 유전체서열분석을 위한 근사문자열 알고리즘 개발

기 여 율 7/10

주관기관 인하대학교 산학협력단

연구기간 2014.05.01 ~ 2015.04.30

이 발명을 지원한 국가연구개발사업

과제고유번호 1415122511

부처명 미래창조과학부

연구관리전문기관 한국산업기술평가관리원

연구사업명 산업융합원천기술개발

연구과제명 상황대응형 분산 트랜스코딩기술을 이용한 저전력 고성능 멀티미디어 콘텐츠 관리기술 개

발

기 여 율 3/10

주관기관 인하대학교 산학협력단

연구기간 2014.06.01 ~ 2015.05.31

명세서

청구범위

청구항 1

편집거리계산의 전처리 단계 개선 방법에 있어서,

문자 집합을 이용하여 발생 가능한 모든 블록들의 편집거리를 미리 계산하고, 상기 블록들의 마지막 열과 행을 록업테이블에 저장하는 단계; 및

상기 록업테이블을 이용하여 블록 단위로 D 테이블 일부의 편집거리를 계산하고, 두 문자열의 편집거리를 계산하는 단계

를 포함하고,

상기 록업테이블을 이용하여 블록 단위로 D 테이블 일부의 편집거리를 계산하고, 두 문자열의 편집거리를 계산하는 단계는,

미리 정해진 크기의 D테이블을 생성하고, 상기 D테이블의 첫 번째 행과 열을 초기화하는 단계;

상기 블록들 중 같은 블록으로 계산되는 부분문자들에 대하여 한번만 계산하도록 동일한 형태로 변환하는 단계; 및

상기 발생 가능한 모든 블록의 조합에 해당하는 록업테이블의 인덱스를 계산하고, 상기 록업테이블을 이용하여 상기 블록 단위로 D테이블의 편집거리를 계산하는 단계

를 포함하는 편집거리계산의 전처리 단계 개선 방법.

청구항 2

삭제

청구항 3

제1항에 있어서,

상기 블록들 중 같은 블록으로 계산되는 부분문자들을 정수문자열로 변환하는

편집거리계산의 전처리 단계 개선 방법.

청구항 4

편집거리계산의 전처리 단계 개선 시스템에 있어서,

문자 집합을 이용하여 발생 가능한 모든 블록들의 편집거리를 미리 계산하고, 상기 블록들의 마지막 열과 행을 록업테이블에 저장하는 전처리부; 및

상기 록업테이블을 이용하여 블록 단위로 D 테이블 일부의 편집거리를 계산하고, 두 문자열의 편집거리를 계산하는 계산부

를 포함하고,

상기 계산부는, 미리 정해진 크기의 D테이블을 생성하고, 상기 D테이블의 첫 번째 행과 열을 초기화하고, 상기 블록들 중 같은 블록으로 계산되는 부분문자들에 대하여 한번만 계산하도록 동일한 형태로 변환하는

편집거리계산의 전처리 단계 개선 시스템.

청구항 5

제4항에 있어서,

상기 계산부는,

상기 발생 가능한 모든 블록의 조합에 해당하는 록업테이블의 인덱스를 계산하고, 상기 록업테이블을 이용하여 상기 블록 단위로 D테이블의 편집거리를 계산하는

편집거리계산의 전처리 단계 개선 시스템.

발명의 설명

기술 분야

[0001] 본 발명은 4-러시안 알고리즘을 이용한 편집거리 계산 방법 및 시스템에 관한 것이다.

배경 기술

[0002] 오차를 허용하는 근사문자열매칭(approximate string matching) 알고리즘은 다양한 분야에서 연구되고 있다. 특히, 검색엔진, 컴퓨터보안, 생물정보학 분야에서 근사문자열매칭 알고리즘들이 유사어검색, 침입탐지, 레퍼런스 매핑 등 다양한 문제들을 해결하는데 이용되고 있다.

[0003] 근사문자열매칭은 문자열의 오차를 판별하는 척도로써 거리함수를 이용한다. 알파벳 Σ 의 문자들로 구성된 두 문자열 X 와 Y 의 거리 $\delta(X, Y)$ 는 X 를 Y 로 변환하기 위해 필요한 최소 비용으로 정의된다. 대표적인 거리함수는 해밍거리(Hamming distance), 편집거리(edit distance), 가중편집거리(weighted edit distance) 등이 있다. 두 문자열 X 와 Y 에 대한 해밍거리는 두 문자열의 길이가 같을 때 X 를 Y 로 변환하기 위해 필요한 최소 교체연산의 수이다. 두 문자열의 편집거리는 X 를 Y 로 변환하기 위해 필요한 최소 편집연산의 수이다. 이때 편집연산은 삽입, 삭제, 교체연산으로 구성된다. 두 문자열의 가중편집거리는 편집연산에 대한 비용을 일반화한 비용행렬(penalty matrix)을 이용하여 X 를 Y 로 변환하는데 필요한 최소 비용으로 정의된다.

[0004] 알파벳 Σ 의 문자들로 구성된 길이가 각각 m, n 인 두 문자열 X 와 Y 의 편집거리는 4-러시안 알고리즘을 이용하여 계산할 수 있다. 편집거리를 계산하기 위한 4-러시안 알고리즘은 두 단계로 구성된다. 첫 번째 단계인 전처리 단계는 전처리를 위해 사용되는 문자열들의 길이가 t 일 때, $O((3|\Sigma|)^{2t}t)$ 시간과 $O((3|\Sigma|)^{2t}t)$ 공간을 이용하여 수행된다. 두 번째 단계인 계산단계는 $O(mn/t)$ 시간과 $O((3|\Sigma|)^{2t}t+mn)$ 공간을 이용하여 수행된다.

발명의 내용

해결하려는 과제

[0005] 본 발명이 이루고자 하는 기술적 과제는 4-러시안 알고리즘 기반 편집거리 계산의 전처리 단계를 개선하는 방법 및 시스템을 제공하는데 있다.

과제의 해결 수단

[0006] 일 측면에 있어서, 본 발명에서 제안하는 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 방법은 문자 집합을 이용하여 발생 가능한 모든 블록들을 미리 계산하고, 상기 블록들의 마지막 열과 행을 록업테이블에 저장하는 단계, 상기 록업테이블을 이용하여 블록 단위로 D 테이블의 일부분만을 계산하고, 두 문자열의 편집거리를 계산하는 단계를 포함할 수 있다.

[0007] 상기 록업테이블을 이용하여 블록 단위로 D 테이블의 일부분만을 계산하고, 두 문자열의 편집거리를 계산하는 단계는 미리 정해진 크기의 D테이블을 생성하고, 상기 D테이블의 첫 번째 행과 열을 초기화하는 단계, 상기 블록들 중 같은 블록으로 계산되는 부분문자열에 대하여 한번만 계산하도록 동일한 형태로 변환하는 단계, 상기 발생 가능한 모든 블록의 조합에 해당하는 록업테이블의 인덱스를 계산하고, 상기 록업테이블을 이용하여 상기 블록 단위로 D테이블을 계산하는 단계를 포함할 수 있다.

[0008] 상기 블록들 중 같은 블록으로 계산되는 부분문자열을 정수문자열로 변환할 수 있다.

[0009] 또 다른 일 측면에 있어서, 본 발명에서 제안하는 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 시스템은 문자 집합을 이용하여 발생 가능한 모든 블록들을 미리 계산하고, 상기 블록들의 마지막 열과 행을 록업테이블에 저장하는 전처리부, 상기 록업테이블을 이용하여 블록 단위로 D 테이블의 일부분만을 계산하고, 두 문

자열의 편집거리를 계산하는 계산부를 포함할 수 있다.

[0010] 상기 계산부는 미리 정해진 크기의 D테이블을 생성하고, 상기 D테이블의 첫 번째 행과 열을 초기화하고, 상기 블록들 중 같은 블록으로 계산되는 부분문자들에 대하여 한번만 계산하도록 동일한 형태로 변환하고, 상기 발생 가능한 모든 블록의 조합에 해당하는 록업테이블의 인덱스를 계산하고, 상기 록업테이블을 이용하여 상기 블록 단위로 D테이블을 계산할 수 있다.

발명의 효과

[0011] 본 발명의 실시예들에 따르면 전처리를 위해 사용되는 문자열들의 수를 줄임으로써 기존의 알고리즘보다 록업테이블의 메모리 크기를 축소할 수 있고, 록업테이블을 생성하는 시간을 감소시킬 수 있다.

도면의 간단한 설명

[0012] 도 1은 종래기술에 따른 D테이블을 나타내는 도면이다.
 도 2는 종래기술에 따른 4-러시안 알고리즘 기반 D 테이블을 나타내는 도면이다.
 도 3은 종래기술에 따른 같은 패턴을 보이는 벡터를 변환하는 과정을 설명하기 위한 도면이다.
 도 4는 본 발명의 일 실시예에 따른 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 방법을 설명하기 위한 흐름도이다.
 도 5는 같은 블록으로 계산되는 문자열을 변환하는 과정을 설명하기 위한 도면이다.
 도 6는 본 발명의 일 실시예에 따른 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 시스템을 설명하기 위한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0013] 본 발명에서는 4-러시안 알고리즘 기반 편집거리 계산의 전처리 단계를 개선하는 알고리즘을 제시한다. 기존의 4-러시안 알고리즘은 전처리를 위해 사용되는 길이 t 인 문자열들을 단순 변환하여 록업테이블을 생성하였다. 본 발명에서는 록업테이블 생성을 위해 사용되는 변환된 문자열들의 수를 줄임으로써 $O(3^{2t}(\min(2t, |\Sigma|))! \times |\Sigma|^{\max(2t-|\Sigma|, 0)})$ 시간과 $O(3^{2t}(\min(2t, |\Sigma|))! \times |\Sigma|^{\max(2t-|\Sigma|, 0)})$ 공간을 이용하여 전처리 단계를 수행하는 알고리즘을 제시한다. 이하, 본 발명의 실시 예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0014]
 [0015] 본 발명의 실시 예를 설명하기에 앞서 제안된 방법에서 사용되는 4-러시안 알고리즘에 대하여 설명한다. 먼저, 용어에 대하여 정리하면, 알파벳 Σ 의 문자들로 구성된 길이 m 인 문자열들의 집합을 Σ^m 으로 표기한다. 공백문자는 λ 로 나타낸다. 문자열 X 의 길이는 $|X|$ 로 표현한다. $X[i]$ ($1 \leq i \leq |X|$)는 X 의 i 번째 문자를 나타낸다. X 의 i 번째부터 j 번째($1 \leq i \leq j \leq |X|$)까지의 부분문자열(substring)은 $X[i..j]$ 로 표기한다. 길이가 m 인 벡터 A 는 $[a_1, a_2, \dots, a_m]$ 로 표기한다.

[0016] 도 1은 종래기술에 따른 D테이블을 나타내는 도면이다.
 [0017] 두 문자열 $X(\in \Sigma^m)$ 와 $Y(\in \Sigma^n)$ 의 편집거리는 동적 프로그래밍을 이용하여 계산할 수 있다. 이때 계산된 $(m+1) \times (n+1)$ 크기의 테이블을 D 테이블이라 하자. D 테이블의 $D[i..j]$ 는 $X[1..i]$ 와 $Y[1..j]$ 의 편집거리를 저장한다. 따라서 $D[m,n]$ 이 두 문자열 X 와 Y 의 편집거리이다.
 [0018] D 테이블을 계산하는 방법은 다음과 같다. 먼저 첫 번째 열과 행에 대해 $D[i,0] = i(0 \leq i \leq m)$, $D[0,j] = j(1 \leq j \leq n)$ 로 초기화한다. 이후 각 $D[i,j]$ 는 수학적식(1)에 의해 계산될 수 있다.

$$D[i,j] = \min \begin{cases} D[i,j-1] + 1 \\ D[i-1,j] + 1 \\ D[i-1,j-1] + \delta(X[i], Y[j]) \end{cases}$$

[0019]

수학식(1)

[0020]

$\delta(X[i], Y[j])$ 는 $X[i]=Y[j]$ 이면 0, 그렇지 않으면 1이다. 각 $D[i, j]$ 는 수학식(1)을 이용하여 $O(1)$ 시간에 계산할 수 있고, 테이블의 크기가 $(m+1) \times (n+1)$ 이므로 $O(mn)$ 시간과 공간을 이용하여 D테이블을 계산할 수 있다. 도 1은 $X="ababca"$ 와 $Y="abcacdcac"$ 에 대한 D테이블을 보여준다.

[0021]

도 2는 종래기술에 따른 4-러시안 알고리즘 기반 D테이블을 나타내는 도면이다. 다시 말해, 도 2는 $t=3$ 일 때 4-러시안 알고리즘을 이용하여 계산한 $X="ababca"$ 와 $Y="abcacdcac"$ 에 대한 D 테이블의 일부분을 나타낸 도면이다.

[0022]

편집거리계산을 위한 4-러시안 알고리즘은 도 2와 같이 D 테이블을 $(t+1) \times (t+1)$ 크기인 블록(t-블록) 단위로, 블록의 테두리 부분만 계산하여 두 문자열의 편집거리를 계산하는 것이다. 이러한 알고리즘은 크게 두 단계로 구성된다. 첫 번째 단계는 전처리 단계로서, 발생 가능한 모든 t-블록을 미리 계산하고, t-블록의 마지막 열과 행을 록업테이블에 저장한다. 두 번째 단계는 계산단계로서, 록업테이블을 이용하여 블록 단위로 D 테이블의 일부분만 계산하면서 두 문자열의 편집거리를 계산한다. 이때 각 t-블록은 D 테이블을 계산하는 방법과 같이 수학식(1)을 통해 계산된다.

[0023]

계산단계에서 왼쪽 상단이 (i, j) (단, i, j 는 t 의 배수)인 t-블록을 계산할 때, $\{A, B, K, C, E\}$ 를 이용하여 A', B', K' 을 결정할 수 있다. K 는 $D[i, j]$ 이고, A와 B는 각각 t-블록에서 $D[i, j]$ 을 제외한 첫 번째 열과 행을 나타내는 벡터 $[D[i+1, j], D[i+2, j], \dots, D[i+t, j]]$ 와 $[D[i, j+1], D[i+1, j+2], \dots, D[i+1, j+t]]$ 를 나타낸다. C와 E는 각각 해당 t-블록에 대응하는 X와 Y의 부분문자열인 $X[(i+1)..(i+t)]$ 와 $Y[(j+1)..(j+t)]$ 이다. A'과 B'은 각각 t-블록에서 마지막 열과 행의 첫 번째 원소를 제외한 벡터인 $[D[i+1, j+t], D[i+2, j+t], \dots, D[i+t, j+t]]$ 와 $[D[i+t, j+1], D[i+t, j+2], \dots, D[i+t, j+t]]$ 이고, K'은 t-블록의 편집거리인 $D[i+t, j+t]$ 이다. 예를 들어, 도 2에서 왼쪽 상단이 $(3,3)$ 인 t-블록은 $K=1$, $a[2,2,3]$, $B=[1,2,3]$, $C="bca"$, $E="ACD"$ 이고, 계산 결과인 $A'=[3,3,3]$, $B'=[2,3,3]$, $K'=3$ 이다. A', B', K'은 인접한 t-블록을 계산할 때, 다시 입력으로 사용된다.

[0024]

한편, 편집거리계산에서 A, B, A', B'은 D 테이블의 임의의 행 또는 열에서 이전의 값과 차이가 최대 1이라는 속성을 이용하여 $\{-1, 0, 1\}$ 의 조합으로 변환하여 경우의 수를 줄일 수 있다. 변환하는 방법은 다음과 같다. A, B, A', B'이 $[a_1, a_2, \dots, a_t]$ 의 형태라면, A와 B는 K를 이용하여 $[(a_1-k), (a_2-a_1), \dots, (a_t-a_{t-1})]$ 로 변환할 수 있고, A'과 B'도 K 대신 각각 변환되기 전의 B[t]와 A[t]를 이용하여 $[(a_1-B[t]), (a_2-a_1), \dots, (a_t-a_{t-1})]$ 과 $[(a_1-A[t]), (a_2-a_1), \dots, (a_t-a_{t-1})]$ 으로 변환할 수 있다. 예를 들어 도 2에서 좌측 상단이 $(3,3)$ 인 t-블록은 $A=[1,0,1]$, $B=[0,1,1]$, $A'=[0,0,0]$, $B'=[-1,1,0]$ 으로 변환할 수 있다.

[0025]

도 3은 종래기술에 따른 같은 패턴을 보이는 벡터를 변환하는 과정을 설명하기 위한 도면이다. 도 3을 참조하면 같은 패턴을 보이는 벡터 A, B, A', B'은 동일한 $\{-1, 0, 1\}$ 의 조합의 벡터로 변환할 수 있다. 변환된 벡터는 계산단계에서 복원할 수 있다. 다시 말해, A와 B는 K를 이용하여 복원할 수 있고, A'과 B'은 각각 복원된 B[t]와 A[t]를 이용하여 복원할 수 있다. 변환된 벡터 A, B에 대해 3^{2t} 개의 경우의 수가 존재하며, 문자열 C, E에 대해 $|\Sigma|^{2t}$ 개의 경우의 수가 존재하므로 록업테이블을 생성하는데 $O((3|\Sigma|)^{2t})$ 시간과 $O((3|\Sigma|)^{2t}t)$ 공간이 필요하다.

[0026]

도 4는 본 발명의 일 실시예에 따른 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 방법을 설명하기 위한 흐름도이다.

[0027]

4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 방법은 문자 집합을 이용하여 발생 가능한 모든 블록

들을 미리 계산하고, 상기 블록들의 마지막 열과 행을 록업테이블에 저장하는 단계(410), 상기 록업테이블을 이용하여 블록 단위로 D 테이블의 일부분만을 계산하고, 두 문자열의 편집거리를 계산하는 단계(420)를 포함할 수 있다.

[0028] 단계(410)에서, 문자 집합을 이용하여 발생 가능한 모든 블록들을 미리 계산하고, 상기 블록들의 마지막 열과 행을 록업테이블에 저장할 수 있다.

[0029] 다시 말해, 길이가 n인 문자열을 X, 길이가 m인 문자열을 Y, X와 Y를 구성하는 문자 집합 Σ 블록의 크기 t라고 할 때, $|\Sigma|$ 와 t에 따라 변환될 수 있는 모든 가능한 길이가 t인 문자열들이 나타날 수 있는 개수를 계산하고 록업테이블을 생성할 수 있다. 그리고, 모든 가능한 {A, B, C, E} 조합에 대해 t-블록의 편집거리를 계산할 수 있다. 계산된 t-블록의 마지막 열과 행인 A' 와 B'를 록업테이블에 저장할 수 있다.

[0030] 단계(420)에서, 상기 록업테이블을 이용하여 블록 단위로 D 테이블의 일부분만을 계산하고, 두 문자열의 편집거리를 계산할 수 있다.

[0031] 단계(420)는 미리 정해진 크기의 D테이블을 생성하고, 상기 D테이블의 첫 번째 행과 열을 초기화하는 단계, 상기 블록들 중 같은 블록으로 계산되는 부분문자들에 대하여 한번만 계산하도록 동일한 형태로 변환하는 단계, 상기 발생 가능한 모든 블록의 조합에 해당하는 록업테이블의 인덱스를 계산하고, 상기 록업테이블을 이용하여 상기 블록 단위로 D테이블을 계산하는 단계를 포함할 수 있다.

[0032] 먼저, $(n+1) \times (m+1)$ 크기의 D 테이블을 생성할 수 있다. 이후, D 테이블의 첫 번째 행과 열을 $D[i, 0] = i (0 \leq i \leq m)$, $D[0, j] = j (1 \leq j \leq n)$ 과 같이 초기화 할 수 있다. 그리고, D 테이블을 계산하기 위해 t-블록의 C, E를 정수 문자열로 변환할 수 있다. {A, B, C, E}에 해당하는 록업테이블의 인덱스를 계산하고, 록업테이블을 이용하여 t-블록단위로 D 테이블을 계산할 수 있다. 이렇게 계산된 $D[m, n]$ 을 출력할 수 있다.

[0033] 도 5는 같은 블록으로 계산되는 문자열을 변환하는 과정을 설명하기 위한 도면이다.

[0034] 예를 들어, $\sigma_1 \dots \sigma_t$, $E = \sigma_{t+1} \dots \sigma_t (\sigma_i \in \Sigma, 1 \leq i \leq 2t)$ 를 변환하는 방법은 다음과 같다. 문자열의 왼쪽부터 변환하며, 문자에 새로운 숫자를 할당할 변수 count를 이용한다. σ_i 가 처음 나타나는 문자라면, count를 이용하여 숫자를 0부터 1씩 증가하며 할당한다. 숫자를 할당 받은 문자는 다음에 다시 나타나면 처음 할당 받은 숫자로 변환된다. 도 5를 참조하면, 두 문자열 "aad", "abc"의 경우인 aadabc는 "001023"으로 변환된다. 이러한 변환은 문자열 "dda", "dcb" 처럼 같은 t-블록으로 계산될 수 있는 경우에 대해 "001023"으로 변환하여 t-블록의 C, E에 대한 경우의 수를 줄일 수 있다. 다시 말해, 기존의 4-러시안 알고리즘은 C, E에 대해 "000000" ("aaaaaa") 부터 "333333" ("dddddd")인 경우를 모두 계산하였다면, 이 알고리즘은 "000000"부터 "012333"인 경우만 계산하면 된다.

[0035] 본 발명에서 제시하는 알고리즘의 시간 및 공간복잡도는 다음과 같다. C와 E를 연결한 문자열을 $S(\in \Sigma^{2t})$ 라 하자. 서로 다른 S의 모든 경우의 수를 고려해보자. $S[1]$ 은 무조건 0으로만 변환되므로 경우의 수는 1개이고, $S[2]$ 는 $0(S[1])$ 과 같은 경우) 또는 $1(S[1])$ 과 다른 경우)인 경우로 경우의 수가 2개이다. 다시 말해, $S[1](1 \leq i \leq |\Sigma|)$ 는 각 i개의 경우가 있다. $S[|\Sigma|]$ 부터는 0부터 $(|\Sigma|-1)$ 로 변환될 수 있으므로 경우의 수는 $|\Sigma|$ 개이다. 따라서 S가 나타날 수 있는 모든 경우의 수는 $2t < |\Sigma|$ 일 때, $(2t)!$ 이고, $2t \geq |\Sigma|$ 일 때, $|\Sigma|! \times |\Sigma|^{2t-|\Sigma|}$ 이다. 다시 말해, $(\min(2t, |\Sigma|))! \times |\Sigma|^{\max(2t-|\Sigma|, 0)}$ 이 된다. 그러므로 전처리 단계는 $O(3^{\frac{2t}{|\Sigma|}} (\min(2t, |\Sigma|))! \times |\Sigma|^{\max(2t-|\Sigma|, 0)} t^2)$ 시간과 $O(3(2t(\min(2t, |\Sigma|))! \times |\Sigma|^{\max(2t-|\Sigma|, 0)} t))$ 공간으로 개선될 수 있다.

[0036] 종래기술에 따른 4-러시안 알고리즘(O4R)과 본 발명에서 제안하는 t-블록에 대한 두 문자열의 경우의 수를 축소 한 4-러시안 알고리즘(R4R)의 필요한 록업테이블의 메모리 크기와 전처리 단계의 수행시간을 비교하였다. 실험 환경은 다음의 표 1과 같다.

[0037] <표 1>

Items	Specifications
OS	Window 7 64bit
CPU	Intel core i7 3820
Main memory	PC12800 (32 GB)
Language	C++
Development tool	Visual studio 2012

[0038]

[0039] 표 2, 3, 4는 t 에 따른 O4R과 R4R에 대해 필요한 록업테이블의 메모리 크기를 나타낸다. 표 3과 4에서 * 표시가 있는 부분은 메모리 공간의 부족으로 실제 록업테이블을 생성하는 실험은 할 수 없었고, 필요한 록업테이블의 크기를 계산하여 나타내었다. 표 2, 3, 4를 통해 $|\Sigma|$ 가 커질수록 공간측면에서 많은 이득을 얻을 수 있다는 것을 알 수 있다. R4R이 O4R보다 $|\Sigma|=2$ 일 때 모든 t 에 대해 2배, $|\Sigma|=4$, $t=4$ 일 때 약 10배, $|\Sigma|=26$, $t=2$ 일 때 약 19,000배만큼 록업테이블의 필요한 메모리 크기를 축소하였다. <표 2>는 $|\Sigma|=2$ 일 때, <표 3>는 $|\Sigma|=4$ 일 때, <표 4>는 $|\Sigma|=26$ 일 때, 각각의 t 에 따른 록업테이블의 메모리 크기를 나타낸 것이다.

[0040] <표 2>

t	O4R	R4R	O4R/R4R
1	72	36	2.00
2	5,184	2,592	2.00
3	279,936	139,968	2.00
4	13,436,928	6,718,464	2.00
5	604,661,760	302,330,880	2.00
6	26,121,388,032	13,060,694,016	2.00

[0041]

[0042] <표 3>

t	O4R	R4R	O4R/R4R
1	288	36	8.00
2	82,944	7,776	10.67
3	17,915,904	1,679,616	10.67
4	3,439,853,568	322,486,272	10.67
5	619,173,642,240*	58,047,528,960*	10.67

[0043]

[0044] <표 4>

t	O4R	R4R	O4R/R4R
1	12,168	36	338.00
2	148,060,224	7,776	19,040.67
3	1,351,197,604,224*	3,149,280	429,049.69
4	10,960,914,965,465,088*	2,116,316,160	5,179,242.67

[0045]

[0046] 표 5, 6, 7은 t 에 따른 O4R과 R4R의 전처리 단계의 수행시간을 보여준다. $|\Sigma|=2$, $t=6$ 일 때는 오랜 수행시간 때문에 100번 수행한 시간의 평균시간을 나타내고, 그 외의 경우는 1,000번 수행한 시간의 평균시간을

나타낸다. 표 5, 6, 7에서 $t=1$ 일 때의 개선율은 무시하였고, 표 7에서 t 가 3과 4일 때, O4R은 메모리 공간의 부족으로 실험할 수 없었다. R4R이 O4R보다 $|\Sigma|=2$, $t=6$ 일 때 약 2배, $|\Sigma|=4$, $t=4$ 일 때 약 10배, $|\Sigma|=26$, $t=2$ 일 때 약 11,000배 빠른 수행시간을 보였다. <표 5>는 $|\Sigma|=2$ 일 때, <표 6>는 $|\Sigma|=4$ 일 때, <표 7>는 $|\Sigma|=26$ 일 때, 각각의 전처리 단계 수행시간을 비교한 것이다.

<표 5>

t	O4R	R4R	O4R / R4R
1	0.000000000	0.000015000	•
2	0.000078000	0.000047000	1.66
3	0.003840000	0.002088000	1.84
4	0.192723000	0.095909000	2.01
5	8.942816000	4.467529000	2.00
6	368.262380000	180.899520000	2.04

<표 6>

t	O4R	R4R	O4R / R4R
1	0.000000000	0.000015000	•
2	0.000889000	0.000094000	9.46
3	0.208047000	0.019308000	10.78
4	41.626347000	3.918416000	10.62

<표 7>

t	O4R	R4R	O4R/R4R
1	0.000187000	0.000000000	•
2	1.589360000	0.000142000	11,192.68
3	•	0.035833000	•
4	•	25.298502000	•

이와 같이 4-러시안 알고리즘은 록업테이블을 생성하는 전처리 단계를 한번만 수행하면, 계산단계에서 다양한 문자열 쌍에 대해 편집거리를 계산할 수 있다. 본 발명에서 제안한 알고리즘을 이용하면, 비슷한 메모리 사용량으로 더 큰 값을 할당할 수 있으므로 전처리 단계 대신 계산단계를 빠르게 수행할 수도 있을 것으로 기대된다.

도 6는 본 발명의 일 실시예에 따른 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 시스템을 설명하기 위한 도면이다.

4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 시스템(600)은 프로세서(610), 버스(620), 네트워크 인터페이스(630), 메모리(640), 및 데이터베이스(650)를 포함할 수 있다. 메모리(640)는 운영체제(641) 및 서비스 제공 루틴(642)을 포함할 수 있다. 프로세서(610)는 전처리부(611), 계산부(612)를 포함할 수 있다. 다른 실시예들에서 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 시스템(600)은 도 6의 구성요소들보다 더 많은 구성요소들을 포함할 수도 있다.

메모리(640)는 컴퓨터에서 관독 가능한 기록 매체로서, RAM(random access memory), ROM(read only memory) 및 디스크 드라이브와 같은 비소멸성 대용량 기록장치(permanent mass storage device)를 포함할 수 있다. 또한,

메모리(640)에는 운영체제(641)와 서비스 제공 루틴(642)을 위한 프로그램 코드가 저장될 수 있다. 이러한 소프트웨어 구성요소들은 드라이브 메커니즘(drive mechanism, 미도시)을 이용하여 메모리(640)와는 별도의 컴퓨터에서 판독 가능한 기록 매체로부터 로딩될 수 있다. 이러한 별도의 컴퓨터에서 판독 가능한 기록 매체는 플로피 드라이브, 디스크, 테이프, DVD/CD-ROM 드라이브, 메모리 카드 등의 컴퓨터에서 판독 가능한 기록 매체(미도시)를 포함할 수 있다. 다른 실시예에서 소프트웨어 구성요소들은 컴퓨터에서 판독 가능한 기록 매체가 아닌 네트워크 인터페이스(630)를 통해 메모리(640)에 로딩될 수도 있다.

[0057] 버스(620)는 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 시스템(600)의 구성요소들간의 통신 및 데이터 전송을 가능하게 할 수 있다. 버스(620)는 고속 시리얼 버스(high-speed serial bus), 병렬 버스(parallel bus), SAN(Storage Area Network) 및/또는 다른 적절한 통신 기술을 이용하여 구성될 수 있다.

[0058] 네트워크 인터페이스(630)는 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 시스템(600)을 컴퓨터 네트워크에 연결하기 위한 컴퓨터 하드웨어 구성요소일 수 있다. 네트워크 인터페이스(630)는 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 시스템(600)을 무선 또는 유선 커넥션을 통해 컴퓨터 네트워크에 연결시킬 수 있다.

[0059] 데이터베이스(650)는 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선을 위한 모든 정보를 저장 및 유지하는 역할을 한다. 도 6에서는 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 시스템(600)의 내부에 데이터베이스(650) 포함하는 것으로 도시하고 있으나, 별개의 다른 시스템 상에 구축된 외부 데이터베이스로서 존재하는 것 또한 가능하다.

[0060] 프로세서(610)는 기본적인 산술, 4-러시안 알고리즘 기반 편집거리계산의 전처리 단계 개선 시스템(600)의 입출력 연산을 수행함으로써, 컴퓨터 프로그램의 명령을 처리하도록 구성될 수 있다. 명령은 메모리(640) 또는 네트워크 인터페이스(630)에 의해, 그리고 버스(620)를 통해 프로세서(610)로 제공될 수 있다. 프로세서(610)는 전처리부(611), 계산부(612)를 위한 프로그램 코드를 실행하도록 구성될 수 있다. 이러한 프로그램 코드는 메모리(640)와 같은 기록 장치에 저장될 수 있다.

[0061] 전처리부(611)는 문자 집합을 이용하여 발생 가능한 모든 블록들을 미리 계산하고, 상기 블록들의 마지막 열과 행을 록업테이블에 저장할 수 있다.

[0062] 계산부(612)는 록업테이블을 이용하여 블록 단위로 D 테이블의 일부분만을 계산하고, 두 문자열의 편집거리를 계산할 수 있다. 다시 말해, 계산부(612)는 미리 정해진 크기의 D테이블을 생성하고, 상기 D테이블의 첫 번째 행과 열을 초기화할 수 있다. 그리고, 상기 블록들 중 같은 블록으로 계산되는 부분문자들에 대하여 한번만 계산하도록 동일한 형태로 변환하고, 상기 발생 가능한 모든 블록의 조합에 해당하는 록업테이블의 인덱스를 계산할 수 있다. 이렇게 계산된 록업테이블을 이용하여 상기 블록 단위로 D테이블을 계산할 수 있다.

[0063] 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, FPA(field programmable 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.

[0064] 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.

[0065]

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 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0066]

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.

[0067]

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

도면

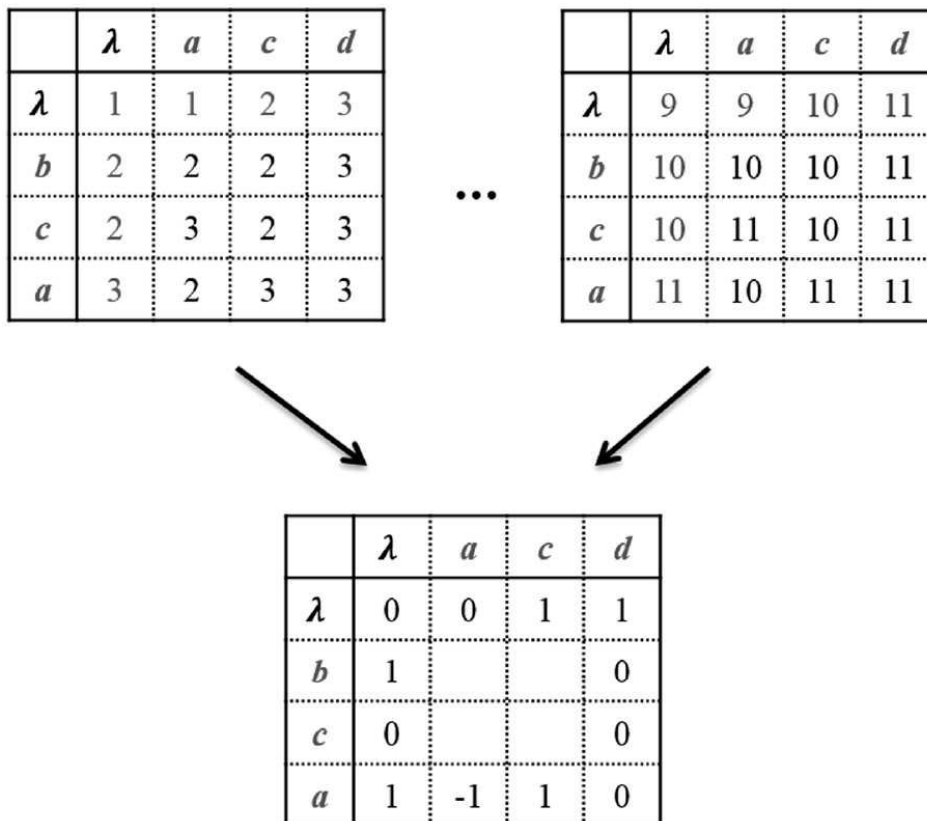
도면1

	λ	a	b	c	a	c	d	c	a	c
λ	0	1	2	3	4	5	6	7	8	9
a	1	0	1	2	3	4	5	6	7	8
b	2	1	0	1	2	3	4	5	6	7
a	3	2	1	1	1	2	3	4	5	6
b	4	3	2	2	2	2	3	4	5	6
c	5	4	3	2	3	2	3	3	4	5
a	6	5	4	3	2	3	3	4	3	4

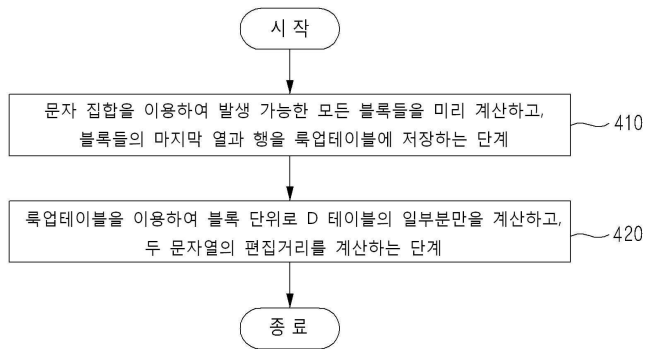
도면2

	λ	a	b	c	a	c	d	c	a	c
λ	0	1	2	3	4	5	6	7	8	9
a	1			2			5			8
b	2			1			4			7
a	3	2	1	1	1	2	3	4	5	6
b	4			2			3			6
c	5			2			3			5
a	6	5	4	3	2	3	3	4	3	4

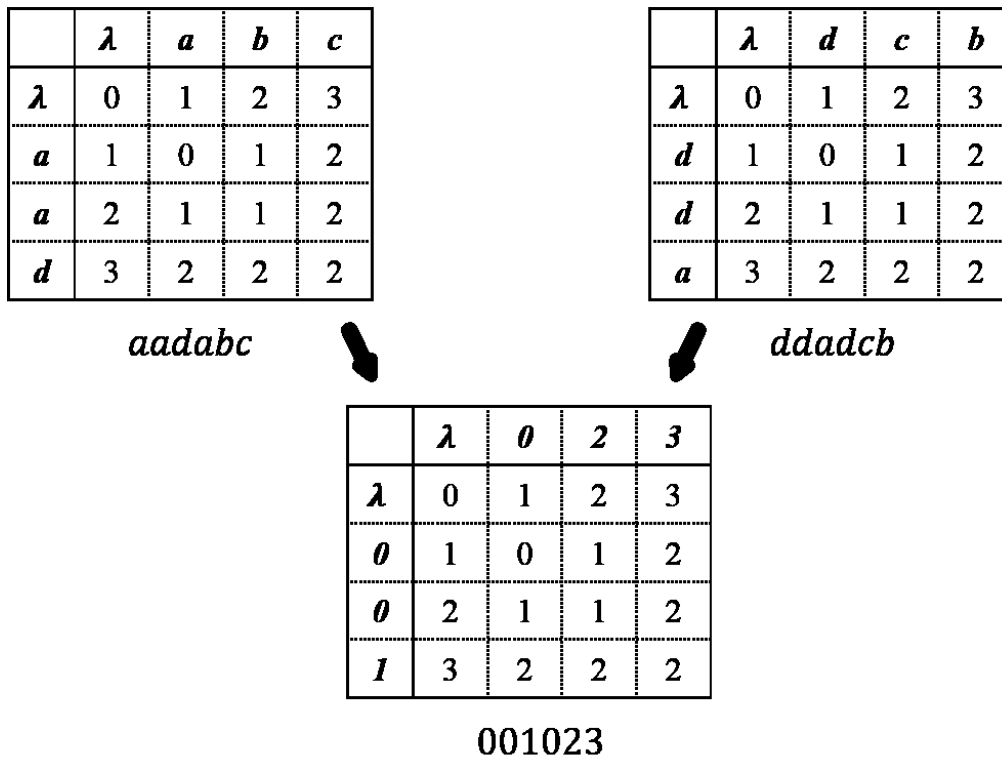
도면3



도면4



도면5



도면6

