

(19) 대한민국특허청(KR)
(12) 공개특허공보(A)(11) 공개번호 10-2021-0078332
(43) 공개일자 2021년06월28일

(51) 국제특허분류(Int. Cl.)

G06F 16/903 (2019.01)

(52) CPC특허분류

G06F 16/90344 (2019.01)

(21) 출원번호 10-2019-0170219

(22) 출원일자 2019년12월18일

심사청구일자 2019년12월18일

(71) 출원인

인하대학교 산학협력단

인천광역시 미추홀구 인하로 100(용현동, 인하대학교)

(72) 발명자

심정섭

인천광역시 남동구 남동대로 860, 120동 601호(간석동, 간석래미안자이아파트)

김영준

경기도 고양시 일산동구 일산로 205, 211동 702호(마두동, 백마마을2단지아파트)

김영호

인천광역시 미추홀구 낙섬동로 6, 102동 301호(용현동, 한국아파트)

(74) 대리인

양성보

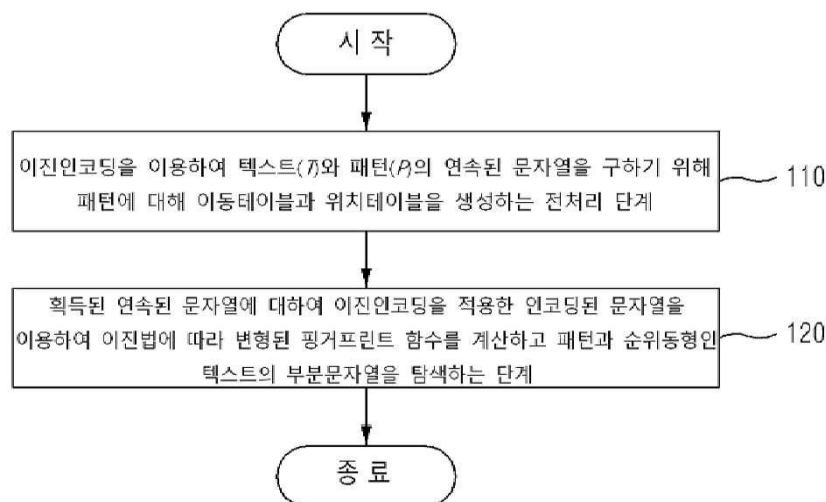
전체 청구항 수 : 총 7 항

(54) 발명의 명칭 이진인코딩을 이용한 호스폴 알고리즘 기반 순위패턴매칭 방법 및 장치

(57) 요약

이진인코딩을 이용한 호스폴 알고리즘 기반 순위패턴매칭 방법 및 장치가 제시된다. 본 발명에서 제안하는 이진인코딩을 이용한 호스폴 알고리즘 기반 순위패턴매칭 방법은 이진인코딩을 이용하여 텍스트(T)와 패턴(P)의 연속된 문자열을 구하기 위해 패턴에 대해 이동테이블과 위치테이블을 생성하는 전처리 단계 및 획득된 연속된 문자열에 대하여 이진인코딩을 적용한 인코딩된 문자열을 이용하여 이진법에 따라 변형된 핑거프린트 함수를 계산하고 패턴과 순위동형인 텍스트의 부분문자열을 탐색하는 단계를 포함한다.

대표도 - 도1



이 발명을 지원한 국가연구개발사업

과제고유번호	1711083273(세부과제번호: 2017R1E1A1A03070867)
부처명	과학기술정보통신부
과제관리(전문)기관명	한국연구재단
연구사업명	전략과제(산업수학)
연구과제명	[Ezbaro] 대용량 시계열 데이터의 반복성 검출과 패턴 예측을 위한 새로운 모델 및
효율적인 알고리즘 개발	
기 여 율	1/1
과제수행기관명	인하대학교
연구기간	2019.03.01 ~ 2020.02.29

명세서

청구범위

청구항 1

이진인코딩을 이용하여 텍스트(T)와 패턴(P)의 연속된 문자열을 구하기 위해 패턴에 대해 이동테이블과 위치테이블을 생성하는 전처리 단계; 및

획득된 연속된 문자열에 대하여 이진인코딩을 적용한 인코딩된 문자열을 이용하여 이진법에 따라 변형된 핑거프린트 함수를 계산하고 패턴과 순위동형인 텍스트의 부분문자열을 탐색하는 단계

를 포함하는 순위패턴매칭 방법.

청구항 2

제1항에 있어서,

이진인코딩을 이용하여 텍스트(T)와 패턴(P)의 연속된 문자열을 구하기 위해 패턴에 대해 이동테이블과 위치테이블을 생성하는 전처리 단계는,

크기가 2^q 인 배열을 할당하고, 각 원소를 최대 이동거리로 초기화한 후 이동테이블(D)을 계산하고, 여기서 q 는 연속된 문자의 수인

순위패턴매칭 방법.

청구항 3

제1항에 있어서,

획득된 연속된 문자열에 대하여 이진인코딩을 적용한 인코딩된 문자열을 이용하여 이진법에 따라 변형된 핑거프린트 함수를 계산하고 패턴과 순위동형인 텍스트의 부분문자열을 탐색하는 단계는,

순위동형 검증횟수를 줄이기 위해 2차 연속된 문자열의 핑거프린트를 이용하여 패턴과 순위동형인 텍스트의 부분문자열을 검색하는

순위패턴매칭 방법.

청구항 4

제3항에 있어서,

2차 연속된 문자열의 핑거프린트도 일치한다면, 위치테이블을 이용하여 패턴과 해당 텍스트가 순위동형인지 검증하고, 순위동형일 경우 상응하는 위치를 출력하는

순위패턴매칭 방법.

청구항 5

이진인코딩을 이용하여 텍스트(T)와 패턴(P)의 연속된 문자열을 구하기 위해 패턴에 대해 이동테이블과 위치테이블을 생성하는 전처리부; 및

획득된 연속된 문자열에 대하여 이진인코딩을 적용한 인코딩된 문자열을 이용하여 이진법에 따라 변형된 핑거프린트 함수를 계산하고 패턴과 순위동형인 텍스트의 부분문자열을 탐색하는 탐색부

를 포함하는 순위패턴매칭 장치.

청구항 6

제5항에 있어서,

전처리부는,

크기가 2^q 인 배열을 할당하고, 각 원소를 최대 이동거리로 초기화한 후 이동테이블(D)을 계산하고, 여기서 q 는 연속된 문자의 수인

순위패턴매칭 장치.

청구항 7

제5항에 있어서,

탐색부는,

순위동형 검증횟수를 줄이기 위해 2차 연속된 문자열의 핑거프린트를 이용하여 패턴과 순위동형인 텍스트의 부분문자열을 검색하고,

2차 연속된 문자열의 핑거프린트도 일치한다면, 위치테이블을 이용하여 패턴과 해당 텍스트가 순위동형인지 검증하고, 순위동형일 경우 상응하는 위치를 출력하는

순위패턴매칭 장치.

발명의 설명

기술 분야

[0001] 본 발명은 이진인코딩을 이용한 호스폴 알고리즘 기반 순위패턴매칭 방법 및 장치에 관한 것이다.

배경 기술

[0002] 정수로 구성된 두 문자열 x , y 에 대해 길이가 같고 문자들의 상대적 순위가 모든 위치에서 동일하면 x 와 y 는 순위동형(order-isomorphic)이라 한다. 예를 들어, $x = (26, 4, 47, 13)$ 과 $y = (37, 15, 68, 31)$ 은 길이가 같고 각 문자의 순위가 (3, 1, 4, 2)로 같으므로 서로 순위동형이다. 순위패턴매칭문제는 길이가 각각 n , m 인 텍스트 T 와 패턴 P 가 주어질 때, P 와 순위동형인 T 의 모든 부분문자열을 찾는 문제이다. 이는 주가지수, 악보의 멜로디와 같은 시계열 데이터 분석에 활용될 수 있다. 최근 순위패턴매칭문제에 대한 연구가 활발히 진행되고 있다. KMP 알고리즘을 기반으로, [2]에서는 패턴 P 를 $O(m)$ 시간에 정렬할 수 있을 때 $O(n + m)$ -시간 알고리즘을, [1]에서는 일반적인 알파벳 Σ 에 대해 $O(n + m \log m)$ -시간 알고리즘을 제시하였다. [3]에서는 호스폴(Horspool) 알고리즘[7]을 기반으로 q 개의 연속된 문자인 q -그램을 이용하여 $O(nm + nq \log q + q!)$ 시간과 $O(m + q!)$ 공간에 이 문제를 해결하는 알고리즘을 제시하였다. 또한 호스폴 알고리즘과 KMP 알고리즘을 결합하여 최악수행 시간을 $O((m + n) q \log q + q!)$ 으로 개선하였다. [4]에서는 [3]의 변형된 호스폴 알고리즘을 기반으로 순위동형 검증횟수를 줄이기 위해 2차 q -그램의 핑거프린트를 활용한 알고리즘을 제시 하였다. [5]에서는 이진인코딩을 이용하여 평균적으로 선형이하 시간에 수행되는 알고리즘을 제시하였다.

[0003] 본 발명에서는 순위패턴매칭문제를 해결하기 위해 [5]의 이진 인코딩을 활용하여 [4]에서 제시된 알고리즘을 개선한다.

발명의 내용

해결하려는 과제

[0004] 본 발명이 이루고자 하는 기술적 과제는 순위패턴매칭문제를 해결하기 위해 이진인코딩을 이용한 호스폴 알고리즘 기반 순위패턴매칭 방법 및 장치를 제공하는데 있다.

과제의 해결 수단

[0005] 일 측면에 있어서, 본 발명에서 제안하는 이진인코딩을 이용한 호스폴 알고리즘 기반 순위패턴매칭 방법은 이진인코딩을 이용하여 텍스트(T)와 패턴(P)의 연속된 문자열을 구하기 위해 패턴에 대해 이동테이블과 위치테이블을 생성하는 전처리 단계 및 획득된 연속된 문자열에 대하여 이진인코딩을 적용한 인코딩된 문자열을 이용하여 이진법에 따라 변형된 핑거프린트 함수를 계산하고 패턴과 순위동형인 텍스트의 부분문자열을 탐색하는 단계를

포함한다.

[0006] 이진인코딩을 이용하여 텍스트(T)와 패턴(P)의 연속된 문자열을 구하기 위해 패턴에 대해 이동테이블과 위치테이블을 생성하는 전처리 단계는 크기가 2^q 인 배열을 할당하고, 각 원소를 최대 이동거리로 초기화한 후 이동테이블(D)을 계산하고, 여기서 q 는 연속된 문자의 수이다.

[0007] 획득된 연속된 문자열에 대하여 이진인코딩을 적용한 인코딩된 문자열을 이용하여 이진법에 따라 변형된 핑거프린트 함수를 계산하고 패턴과 순위동형인 텍스트의 부분문자열을 탐색하는 단계는 순위동형 검증횟수를 줄이기 위해 2차 연속된 문자열의 핑거프린트를 이용하여 패턴과 순위동형인 텍스트의 부분문자열을 검색한다.

[0008] 2차 연속된 문자열의 핑거프린트도 일치한다면, 위치테이블을 이용하여 패턴과 해당 텍스트가 순위동형인지 검증하고, 순위동형일 경우 상응하는 위치를 출력한다.

[0009] 또 다른 일 측면에 있어서, 본 발명에서 제안하는 이진인코딩을 이용한 호스폴 알고리즘 기반 순위패턴매칭 장치는 이진인코딩을 이용하여 텍스트(T)와 패턴(P)의 연속된 문자열을 구하기 위해 패턴에 대해 이동테이블과 위치테이블을 생성하는 전처리부 및 획득된 연속된 문자열에 대하여 이진인코딩을 적용한 인코딩된 문자열을 이용하여 이진법에 따라 변형된 핑거프린트 함수를 계산하고 패턴과 순위동형인 텍스트의 부분문자열을 탐색하는 탐색부를 포함한다.

발명의 효과

[0010] 본 발명의 실시예들에 따른 이진인코딩을 이용한 호스폴 알고리즘 기반 순위패턴매칭 방법 및 장치를 통해 순위패턴매칭문제를 해결할 수 있고, 다우존스지수 데이터를 이용하여 실험한 결과, $n = 34685$, $m = 20$, $q = 6$ 일 때, 제안하는 알고리즘은 기존의 알고리즘보다 최대 약 45% 빠르게 수행되었다.

도면의 간단한 설명

[0011] 도 1은 본 발명의 일 실시예에 따른 이진인코딩을 이용한 호스폴 알고리즘 기반 순위패턴매칭 방법을 설명하기 위한 흐름도이다.

도 2는 본 발명의 일 실시예에 따른 이진인코딩을 이용한 호스폴 알고리즘 기반 순위패턴매칭 장치의 구성을 나타내는 도면이다.

도 3은 본 발명의 실시예에 따른 다우존스지수 데이터에 대한 호스폴 알고리즘 기반 알고리즘들의 가장 빠른 수행시간을 비교한 그래프이다.

발명을 실시하기 위한 구체적인 내용

[0012] 본 발명의 실시예에 따르면, 기존의 $O(m + q!)$ 공간을 $O(m + 2^q)$ 공간으로 줄이고, $O(nm + nq \log q + q!)$ 시간을 $O(m + 2^q)$ 시간으로 개선한 알고리즘을 제시한다. 다우존스지수 데이터를 이용하여 실험한 결과, $n = 34685$, $m = 20$, $q = 6$ 일 때, 제안하는 알고리즘은 기존의 알고리즘보다 최대 약 45% 빠르게 수행되었다. 이하, 본 발명의 실시 예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0013] Σ 는 정수문자로 이루어진 집합이다. Σ 로 구성된 길이가 m 인 문자열들의 집합을 Σ^m 으로 표기한다. 문자열 x 가 주어졌을 때, x 의 길이를 $|x|$ 로 나타낸다. x 의 i ($0 \leq i < |x|$) 번째 문자는 $x[i]$ 로 표기하고, x 의 i 번째부터 j ($0 \leq i \leq j < |x|$) 번째까지의 부분문자열을 $x[i..j]$ 로 표기한다.

[0014] 두 문자열 x, y ($\in \Sigma^m$)에 대해 $x[i] \leq x[j] \Leftrightarrow y[i] \leq y[j]$ ($0 \leq i, j < m$) 이면 x 와 y 는 순위동형이라 하고 $x \approx y$ 로 표기한다. 순위동형 판단을 위해 점두사표현을 이용한 점두사테이블과 최근점이 윗표현을 이용한 위치테이블을 사용할 수 있다. 문자열 x 에 대한 점두사테이블 μ_x 는 다음과 같이 정의된다.

[0015]
$$\mu_x[i] = |\{j : x[j] \leq x[i], 0 \leq j < i\}|$$

[0016] 만약 $\mu_x = \mu_y$ 이면 $x \approx y$ 이다. 접두사테이블 μ_x 는 순위통계트리를 이용하여 $O(m \log m)$ 시간에 계산할 수 있다. 문자열 x 에 대한 위치테이블 $LMax_x, LMin_x$ 는 다음과 같이 정의된다.

$$LMax_x[i] = j \text{ if } x[j] = \max\{x[k] : k \in [0, i-1], x[k] \leq x[i]\}$$

$$LMin_x[i] = j \text{ if } x[j] = \min\{x[k] : k \in [0, i-1], x[k] \geq x[i]\}$$

[0017]

[0018] 즉, $LMax_x[i]$ 는 $x[0..i-1]$ 에서 $x[i]$ 보다 작거나 같은 값 중 최댓값의 위치 j ($0 \leq j < i$) 이고, $LMin_x[i]$ 는 $x[0..i-1]$ 에서 $x[i]$ 보다 크거나 같은 값 중 최솟값의 위치 j 이다. 만약 그러한 j 가 2 개 이상이면, 이 중 가장 큰 j 를 저장하고, 그러한 j 가 없다면 -1을 저장한다. 두 문자열 x, y ($\in \Sigma^m$) 와 위치테이블 $LMax_x, LMin_x$ 가 주어졌을 때, p 를 조건 $y[LMax_x[i]] < y[i]$, q 를 조건 $y[i] < y[LMin_x[i]]$ 라 하자. 모든 i ($1 \leq i < m$) 에 대해 $(p \wedge q) \vee (\neg p \wedge \neg q)$ 를 만족하면, $x \approx y$ 이다. 위치테이블 $LMax_x, LMin_x$ 는 정렬알고리즘을 이용하여 $O(m \log m)$ 시간 에 계산할 수 있고, 이를 이용하여 x 와 y 의 순위동형 여부를 $O(m)$ 시간에 검증할 수 있다. 표 1은 문자열 $x = (21, 69, 63, 57, 59, 67)$ 에 대한 접두사테이블 μ_x , 위치테이블 $LMax_x$ 와 $LMin_x$ 가 이진수로 인코딩된 x' 을 보여준다.

[0019] <표 1>

i	0	1	2	3	4	5
x	21	69	63	57	59	67
μ_x	0	1	1	1	2	4
$LMax_x$	-1	0	0	0	3	2
$LMin_x$	-1	-1	1	2	2	1
x'	1	0	0	1	1	

[0020]

[0022] 순위패턴매칭문제는 텍스트 $T (\in \Sigma^n)$, 패턴 $P (\in \Sigma^m)$ 가 주어졌을 때, $P \approx T[i-m+1..i] (m-1 \leq i < n)$ 의 모든 위치 $i-m+1$ 을 찾는 문제이다.

[0023] 순위패턴매칭문제에서는 한 문자씩 비교할 때, 문자의 순위가 항상 0으로 일치하게 된다. 따라서 순위패턴매칭 문제에 일반적인 패턴매칭문제에 대한 호스폴(Horspool) 알고리즘을 그대로 활용 할 수 없다. [3]에서는 순위패턴매칭문제에 호스폴 알고리즘의 오문자규칙(bad-character heuristic)을 적용하기 위해, 하나의 문자가 아닌, 연속된 q 개의 문자인 q -그램을 사용하고, 이를 하나의 정수로 사상하는 핑거프린트를 이용한다. 이때 임의의 q -그램 x 에 대한 핑거프린트 $f(x)$ 는 다음과 같이 정의된다.

$$f(x) = \sum_{k=0}^{q-1} \mu_x[k] \times k!$$

[0024]

[0025] $f(x)$ 는 계승수체계를 사용하므로 범위가 $0 \sim q!-1$ 이다. 예를 들어, q -그램 $x = (21, 69, 63, 57)$ 에 대

해 $\mu_x = (0, 1, 1, 1)$, 핑거프린트 $f(x) = 9$ 이다. [3]에서 제시된 알고리즘은 전처리단계와 탐색단계로 구성된다. 전처리단계에서는 패턴 P 에 대해 이동테이블과 위치테이블을 생성하고, 탐색단계에서는 이동테이블과 위치테이블을 활용 하여 호스폴 알고리즘을 기반으로 P 와 순위동형인 T 의 모든 부분문자열을 검색한다. 전처리단계에서 이동테이블 D 는 다음과 같이 계산된다. 먼저 D 에 대해 크기가 $q!$ 인 배열을 할당하고, 각 원소를 최대 이동거리인 $m - q + 1$ 로 초기화한다. 이후 P 에 대해 다음의 수식으로 D 를 계산한다.

$$t = \max\{i : \mu_{P[i-q+1..i]} = \mu_x, q-1 \leq i < m-1\}$$

$$D[f(x)] = \min(m-q+1, m-t-1)$$

위치테이블 $LMax_p$, $LMin_p$ 는 앞서 설명한 방법으로 계산한다. 탐색단계는 스텝 i ($m-1 \leq i < n$)가 존재하며, 각 스텝 i 에서는 $P \approx T[i-m+1..i]$ 인지 확인한다. 구체적으로, 각 스텝 i 에 서는 $f(P[m-q..m-1])$ 를 계산하여 미리 계산한 $f(P[m-q..m-1])$ 과 같은지 비교한다. 만약 두 값이 다르다면, $T[i-m+1..i]$ 와 P 는 순위동형일 수 없으므로 i 를 $D[f(T[i-q+1..i])]$ 만큼 증가시킨다. 만약 $f(T[i-q+1..i])$ 와 $f(P[m-q..m-1])$ 이 같다면, $T[i-m+1..i]$ 와 P 는 순위동형일 수 있기 때문에 위치테이블을 이용하여 순위동형임을 검증한다. 만약 순위동형이면 위치 $i - m + 1$ 을 출력한다. 이후 순위동형의 여부와 관계없이 i 를 $D[f(T[i-q+1..i])]$ 만큼 증가시키고, $i < n$ 인 경우 이 과정을 반복한다.

[3]에서 제시된 알고리즘의 복잡도는 다음과 같다. 전처리단계에서 위치테이블은 $O(m)$ 공간과 $O(m \log m)$ 시간에 계산할 수 있고, 이동테이블은 $O(q!)$ 공간과 $O(q! + mq \log q)$ 시간에 계산할 수 있다. 탐색단계에서는 모든 스텝 i 에서 $f(T[i-q+1..i])$ 계산과 순위동형 검증을 할 수 있으므로 $O(nq \log q + nm)$ 시간에 수행된다. 따라서 순위패턴매칭문제를 $(nm + nq \log q + q!)$ 시간과 $O(m + q!)$ 공간에 해결할 수 있다. [4]에서는 [3]의 알고리즘을 기반으로 순위동형 검증횟수를 줄이기 위해 2차 q -그램의 핑거프린트를 이용한 알고리즘을 제시하였다. 즉, 스텝 i 에서 $f(T[i-q+1..i])$ 와 $f(P[m-q..m-1])$ 이 같다면, T 와 P 의 2차 q -그램의 핑거프린트인 $f(T[i-q..i-1])$ 과 $f(P[m-q-1..m-2])$ 를 한 번 더 비교하여 순위동형후보를 줄이는 전략을 사용하였다. [4]의 알고리즘의 복잡도는 [3]의 알고리즘의 복잡도와 동일하다.

[5]에서는 순위패턴매칭문제를 해결하기 위해 문자열을 이진수로 인코딩한다. 문자열 x ($\in \Sigma^m$)에 대한 인코딩된 문자열 x' ($|x'| = m-1$)은 i ($0 \leq i < m-1$)에 대해 다음과 같이 정의된다. 편의상, x 에서 x' 으로의 변환을 이진인코딩(binary encoding)으로 표기한다.

$$x'[i] = \begin{cases} 1 & \text{if } x[i] < x[i+1], \\ 0 & \text{otherwise.} \end{cases}$$

x' 은 x 의 인접한 두 원소들의 대소비교를 통해 $O(m)$ 시간에 계산할 수 있다(표 1 참조).

도 1은 본 발명의 일 실시예에 따른 이진인코딩을 이용한 호스폴 알고리즘 기반 순위패턴매칭 방법을 설명하기 위한 흐름도이다.

[0034] 제안하는 이진인코딩을 이용한 호스폴 알고리즘 기반 순위패턴매칭 방법은 이진인코딩을 이용하여 텍스트(T)와 패턴(P)의 연속된 문자열을 구하기 위해 패턴에 대해 이동테이블과 위치테이블을 생성하는 전처리 단계(110) 및 획득된 연속된 문자열에 대하여 이진인코딩을 적용한 인코딩된 문자열을 이용하여 이진법에 따라 변형된 핑거프린트 함수를 계산하고 패턴과 순위동형인 텍스트의 부분문자열을 탐색하는 단계(120)를 포함한다.

[0035] 단계(110)에서, 이진인코딩을 이용하여 텍스트(T)와 패턴(P)의 연속된 문자열을 구하기 위해 패턴에 대해 이동테이블과 위치테이블을 생성한다. 먼저, 크기가 2^q 인 배열을 할당하고, 각 원소를 최대 이동거리로 초기화한 후 이동테이블(D)을 계산하고, 여기서 q 는 연속된 문자의 수이다.

[0036] 단계(120)에서, 획득된 연속된 문자열에 대하여 이진인코딩을 적용한 인코딩된 문자열을 이용하여 이진법에 따라 변형된 핑거프린트 함수를 계산하고 패턴과 순위동형인 텍스트의 부분문자열을 탐색한다. 순위동형 검증횟수를 줄이기 위해 2차 연속된 문자열의 핑거프린트를 이용하여 패턴과 순위동형인 텍스트의 부분문자열을 검색한다. 2차 연속된 문자열의 핑거프린트도 일치한다면, 위치테이블을 이용하여 패턴과 해당 텍스트가 순위동형인지 검증하고, 순위동형일 경우 상응하는 위치를 출력한다.

[0037] 제안하는 알고리즘의 기본 아이디어는 점두사 테이블을 이용하는 대신 이진인코딩을 이용하여 T 와 P 의 q -그램을 구하는 것이다. 그리고 계산한 q -그램에 대하여 핑거프린트는 계승수체계가 아닌 이진법을 이용하여 계산한다. 즉, μ_x 를 이용하는 것이 아니라 아래의 식과 같이 이진인코딩을 적용한 x' 을 이용하여 이진법에 따라 변형된 핑거프린트 함수 $f'(x')$ 를 계산한다.

$$f'(x') = \sum_{k=0}^{q-1} x'[k] \times 2^{q-k-1}$$

[0038] 예를 들어, 문자열 $x = (21, 69, 63, 57)$ 에 대해 인코딩된 문자열 $x' = (1, 0, 0)$, $f'(x') = 4$ 이다. $f'(x') = 4$ 는 계승수체계를 사용하지 않으며, $0 \sim 2^q - 1$ 의 범위를 갖는다.

[0040] 본 발명에서 제안하는 알고리즘은 전처리단계(110) 및 탐색단계(120)를 포함한다. 전처리단계(110)에서는 P 에 대해 이동테이블과 위치테이블을 생성한다. 위치테이블 $LMax_p$, $LMin_p$ 는 [3]에서 제시된 방법으로 계산한다. 이동테이블 D 는 먼저 크기가 2^q 인 배열을 할당하고, 각 원소를 최대 이동거리인 $m - q$ 로 초기화한다. 이후 P 에 대해 다음의 수식으로 D 를 계산한다.

$$t = \max\{i : P'[i - q + 1..i] = x', q - 1 \leq i < m - 2\}$$

$$D[f'(x')] = \min(m - q, m - t - 1)$$

[0041] 탐색단계(120)에서는 순위동형 검증횟수를 줄이기 위해 [4]와 같이 2차 q -그램의 핑거프린트를 이용하여 P 와 순위동형인 T 의 부분문자열을 검색한다. 이때 1차 q -그램의 핑거프린트가 일치할 확률은 $1/2^q$ 이고, 2차 q -그램의 핑거프린트까지 일치할 확률은 $1/2^{q+1}$ 이다. 한편, 실험 결과, 3차 q -그램부터는 성능향상이 미미하여 2차 q -그램까지만 이용한다. 구체적으로, 스텝 $i(m - 1 \leq i < n)$ 가 존재하며, 각 스텝 i 에서 $f'(T'[i - q..i - 1])$ 과 $f'(P'[m - q - 1..m - 2])$ 를 비교한다. 만약 두 값이 일치하면 2차 q -그램의 핑거프린트인 $f'(T'[i - q - 1..i - 2])$ 와 $f'(P'[m - q - 2..m - 3])$ 를 비교한다. 이때 2차 q -그램의 핑거프린트는 1차 q -그램으로부터 라빈-카프(Rabin-Karp) 방법[10]을 이용하여 $O(1)$ 시간에 계산할 수 있다. 만약 2차 q -그램의 핑거프린트도 일치한다면, 위치테이블을 이용하여 P 와 $T[i - m + 1..i]$ 가 순위동형인지 검증하고, 순위동형이면 위치 $i - m + 1$ 을 출력한다. 이후 순위동형 여

부에 관계없이 i 를 $D[f'(T'[i-q..i-1])]$ 만큼 증가시키고 $i < n$ 이면 이 과정을 반복한다.

- [0043] 본 발명에서 제안하는 알고리즘의 복잡도는 다음과 같다. 전처리단계는 위치테이블과 이동테이블을 각각 $O(m \log m)$ 시간, $O(2^q + m)$ 시간에 계산할 수 있다. 이동테이블의 공간을 $O(q!)$ 에서 $O(2^q)$ 으로 변경하였으며, 이는 스티어링(Stirling)의 근사로부터 더 낮은 증가율을 보임을 알 수 있다. 탐색단계에서는 모든 스텝 i 에서 핑거프린트 계산과 순위동형 검증을 할 수 있으므로 $O(nm)$ 시간에 수행할 수 있다. 따라서 본 발명에서 제안하는 알고리즘은 $O(m + 2^q)$ 공간을 사용하여 $O(nm + 2^q)$ 시간에 순위패턴매칭문제를 해결할 수 있다.
- [0045] 도 2는 본 발명의 일 실시예에 따른 이진인코딩을 이용한 호스폴 알고리즘 기반 순위패턴매칭 장치의 구성을 나타내는 도면이다.
- [0046] 이진인코딩을 이용한 호스폴 알고리즘 기반 순위패턴매칭 장치(200)는 전처리부(210) 및 탐색부(220)를 포함한다.
- [0047] 전처리부(210) 및 탐색부(220)는 도 1의 단계들(110~120)을 수행하기 위해 구성될 수 있다.
- [0048] 전처리부(210)는 이진인코딩을 이용하여 텍스트(T)와 패턴(P)의 연속된 문자열을 구하기 위해 패턴에 대해 이동테이블과 위치테이블을 생성한다. 먼저, 크기가 2^q 인 배열을 할당하고, 각 원소를 최대 이동거리로 초기화한 후 이동테이블(D)을 계산하고, 여기서 q 는 연속된 문자의 수이다.
- [0049] 탐색부(220)는 획득된 연속된 문자열에 대하여 이진인코딩을 적용한 인코딩된 문자열을 이용하여 이진법에 따라 변형된 핑거프린트 함수를 계산하고 패턴과 순위동형인 텍스트의 부분문자열을 탐색한다. 탐색부(220)는 순위동형 검증횟수를 줄이기 위해 2차 연속된 문자열의 핑거프린트를 이용하여 패턴과 순위동형인 텍스트의 부분문자열을 검색한다. 2차 연속된 문자열의 핑거프린트도 일치한다면, 위치테이블을 이용하여 패턴과 해당 텍스트가 순위동형인지 검증하고, 순위동형일 경우 상응하는 위치를 출력한다.
- [0051] 도 3은 본 발명의 실시예에 따른 다우존스지수 데이터에 대한 호스폴 알고리즘 기반 알고리즘들의 가장 빠른 수행시간을 비교한 그래프이다($n = 34685$, $m = \{5, 10, 15, 20\}$, $3 \leq q \leq 6$).
- [0052] 실험환경은 다음과 같다. OS는 Windows 10(64bit), CPU는 Intel Core i7-6700(3.4GHz), RAM은 32GB의 크기를, 개발 툴은 Visual Studio 2015, 개발 언어는 C++을 사용하였다. 텍스트 T 는 1885년 2월 16일부터 2010년 12월 31까지의 다우존스산업평균지수의 일별 종가 ($n = 34685$)로 생성한 다우존스지수 데이터를 이용하였다. 패턴 P 는 T 의 부분문자열 중 무작위로 추출하였다. 패턴의 길이 m 은 5, 10, 15, 20, q 는 3, 4, 5, 6에 대해 실험하였으며, 알고리즘의 수행시간은 각각 10,000 번 실험한 수행시간의 합으로 측정하였다. 편의상, [3]에서 제시된 호스폴 알고리즘 기반 알고리즘을 OHq, [4]에서 제시된 알고리즘을 OHSq, 본 발명에서 제시하는 알고리즘을 OHESq로 표기한다. 한편, [3]에서의 호스폴 알고리즘과 KMP 알고리즘을 결합한 알고리즘은 OHq보다 실제 수행시간이 느리기 때문에, 본 발명에서는 OHq와 비교한다.
- [0053] 표 2는 다우존스지수 데이터에 대해 OHq, OHSq, OHESq의 수행시간을 보여준다.

<표 2>

알고리즘	m	수행시간(단위: 초)			
		$q=3$	$q=4$	$q=5$	$q=6$
OHq	5	1.574	1.773	-	-
OHSq		1.695	2.412	-	-
OHESq		1.49	2.597	-	-
OHq	10	0.993	0.771	0.9	1.153
OHSq		1.016	1.054	1.377	1.924
OHESq		0.751	0.639	0.589	0.615
OHq	15	0.822	0.565	0.573	0.639
OHSq		0.879	0.741	0.848	1.08
OHESq		0.632	0.458	0.381	0.337
OHq	20	0.779	0.478	0.44	0.458
OHSq		0.776	0.607	0.64	0.768
OHESq		0.592	0.4	0.302	0.254

표 2에서 각 파라미터마다 가장 빠른 알고리즘의 수행시간을 진하게 표시하였다. $m = 5$, $q = 4$ 인 경우를 제외한 다른 모든 경우에서 OHESq가 가장 빠르게 수행되었다. OHESq는 OHq와 OHSq 중 더 빠른 알고리즘보다 $m = 5$, $q = 3$ 일 때 약 5%, $m = 10$, $q = 4$ 일 때 약 17%, $m = 15$, $q = 5$ 일 때 약 34%, $m = 20$, $q = 6$ 일 때 약 45% 빠르게 수행되었다. 도 3은 q 에 상관없이 m 에 따라 각 알고리즘들의 가장 빠른 수행시간을 보여준다.

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.

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 장치에 구체화(embodiment)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.

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.

[0061] 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.

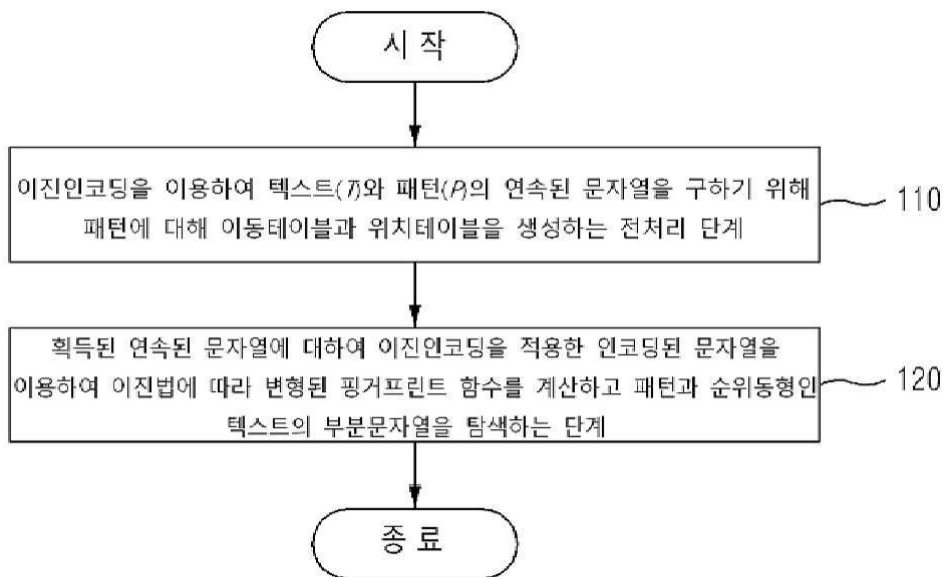
[0062] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

[0064] <참고문헌>

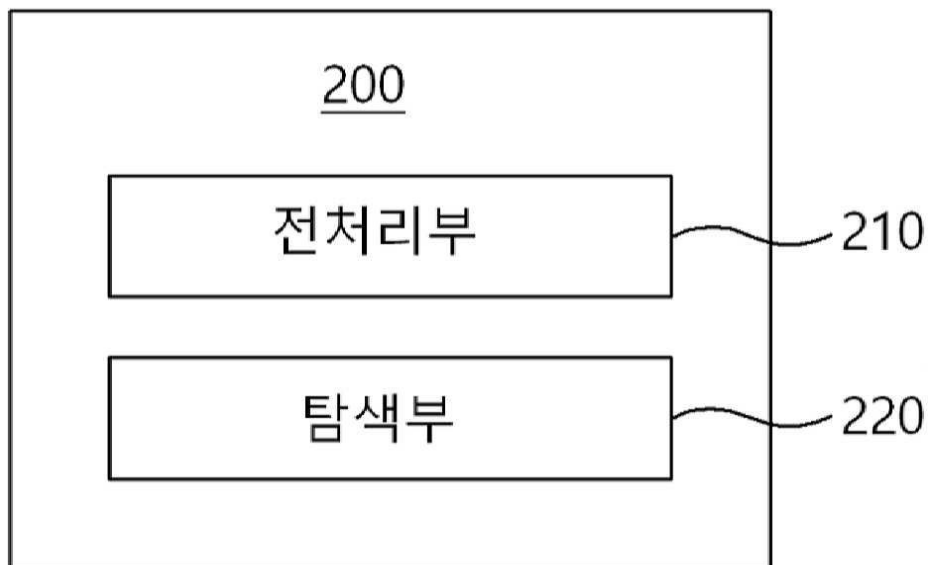
- [0065] [1] J. Kim, P. Eades, R. Fleischer, S. Hong, C. S. Iliopoulos, K. Park, S. J. Puglisi, T. Tokuyama, "Order-preserving matching," Theoretical Computer Science, Vol. 525, pp. 68-79, 2014.
- [0066] [2] M. Kubica, T. Kulczynski, J. Radoszewski, W. Rytter, T. Walen, "A linear time algorithm for consecutive permutation pattern matching," Information Processing Letters, Vol. 113, pp. 430-433, 2013.
- [0067] [3] S. Cho, J. Na, K. Pa, J. Sim, "A fast algorithm for order-preserving pattern matching," Information Processing Letters, Vol. 115, pp. 397-402, 2015.
- [0068] [4] G. Yoo, Y. Kim, J. Sim, "A Horspool-based order-preserving pattern matching algorithm using fingerprints of secondary -grams," KSC 2017, pp. 2186-2188, 2017.
- [0069] [5] T. Chhabra, J. Tariho, "A filtration method for order-preserving matching," Information Processing Letters, Vol. 116, pp. 71-74, 2016.
- [0070] [6] D. E. Knuth, J. H. Morris, V. R. Pratt, "Fast pattern matching in strings," SIAM journal on computing, Vol. 6, no. 2, pp. 323-350, 1977.
- [0071] [7] R. N. Horspool, "Practical fast searching in strings," Software: Practice and Experience, Vol. 10, No. 6, pp. 501-506, 1980.
- [0072] [8] D. E. Knuth, The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, 3rd Ed., Addison-Wesley, 1997.
- [0073] [9] M. Mares, M. Straka, "Linear-time ranking of permutations," European Symposium on Algorithms, Vol. 4698, pp. 187-193, 2007.
- [0074] [10] H. Cormen, E. Leiserson, L. Rivest, C. Stein, "Introduction to Algorithms," 3rd Ed., The MIT press, 2009.

도면

도면1



도면2



도면3

