



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2018-0065496
(43) 공개일자 2018년06월18일

(51) 국제특허분류(Int. Cl.)
G06F 17/10 (2006.01)

(52) CPC특허분류
G06F 17/10 (2013.01)

(21) 출원번호 10-2016-0166451

(22) 출원일자 2016년12월08일

심사청구일자 2016년12월08일

(71) 출원인

인하대학교 산학협력단

인천광역시 남구 인하로 100, 인하대학교 (용현동)

(72) 발명자

심정섭

인천광역시 남동구 문화로 161, 102동 502호(구월동, 대우재래아파트)

김영호

인천광역시 남구 인하로77번길 28, 가동 406호(용현동, 새한아파트)

(74) 대리인

양성보

전체 청구항 수 : 총 6 항

(54) 발명의 명칭 정수문자집합에 대한 문자열의 δ -근사주기와 γ -근사주기 계산 방법

(57) 요약

정수문자집합에 대한 문자열의 δ -근사주기와 γ -근사주기 계산 방법이 개시된다. 일 실시예에 따른 근사주기 계산 방법은, 정수문자로 구성된 길이가 n 인 문자열(T)에 대하여, 길이가 $n/2$ 인 배열을 생성하는 단계; 상기 길이가 m ($1 \leq m \leq n/2$)인 문자열(T)의 δ -근사주기 또는 γ -근사주기를 계산하는 단계; 및 상기 δ -근사주기 또는 γ -근사주기와 관련된 문자열(P)을 획득하는 단계를 포함할 수 있다.

대표도 - 도3

• $T = (17, 12, 21, 16, 14, 20, 23, 7, 25, 19, 15), m = 3$

	T_1			T_2			T_3			T_4	
T	17	12	21	16	14	20	23	7	25	19	15

T_1	17	12	21
T_2	16	14	20
T_3	23	7	25
T_4	19	15	

A	16	7	20
B	23	15	25

$A[i] = \min\{T_k[i], 1 \leq k \leq [n/m]\}$

$B[i] = \max\{T_k[i], 1 \leq k \leq [n/m]\}$

P	19.5	11	22.5
---	------	----	------

$P[i] = (A[i] + B[i])/2$

$\therefore P = (19, 11, 22), (19, 11, 23), (20, 11, 22), (20, 11, 23)$

$$d = 4 (\max_{1 \leq i \leq m} \{ \max(|A[i] - P[i]|, |B[i] - P[i]|) \})$$

이 발명을 지원한 국가연구개발사업

과제고유번호 1711030765

부처명 미래창조과학부

연구관리전문기관 한국연구재단

연구사업명 중견연구자지원

연구과제명 시계열데이터 패턴분석을 위한 순위패턴매칭 알고리즘 개발

기 여 율 1/1

주관기관 인하대학교 산학협력단

연구기간 2015.11.01 ~ 2016.10.31

명세서

청구범위

청구항 1

근사주기 계산 방법에 있어서,

정수문자로 구성된 길이가 n 인 문자열(T)에 대하여, 길이가 $n/2$ 인 배열을 생성하는 단계;

상기 길이가 m ($1 \leq m \leq n/2$)인 문자열(T)의 δ -근사주기 또는 γ -근사주기를 계산하는 단계; 및

상기 δ -근사주기 또는 γ -근사주기와 관련된 문자열(P)을 획득하는 단계

를 포함하는 근사주기 계산 방법.

청구항 2

제1항에 있어서,

상기 문자열 $T(\in \Sigma^n)$ 와 $P(\in \Sigma^m, m \leq n/2)$ 가 주어졌을 때, $j(1 \leq j \leq m)$ 에 대해

$((i-1) \bmod m) + 1 = j$ 인 $i(1 \leq i \leq n)$ 의 집합을 E_j 라고 하면,

상기 $j(1 \leq j \leq m)$, $k(\in E_j)$ 에 대해 $|T[k] - P[j]| \leq d$ 이면, 상기 문자열(P)은 상기 문자열(T)에 대해 거리가 d 인 δ -근사주기이고,

상기 $j(1 \leq j \leq m)$, $k(\in E_j)$ 에 대해 $\sum_{j=1}^m \sum_{k \in E_j} |T[k] - P[j]| \leq d$ 를 만족하면, 상기 문자열(P)은

상기 문자열(T)에 대해 거리가 d 인 γ -근사주기인

것을 특징으로 하는 근사주기 계산 방법.

청구항 3

제1항에 있어서,

상기 길이가 m ($1 \leq m \leq n/2$)인 문자열(T)의 δ -근사주기 또는 γ -근사주기를 계산하는 단계는,

상기 길이가 m ($1 \leq m \leq n/2$)인 문자열(T)의 δ -근사주기들 중 최소 δ -근사주기와 관련된 문자열(P)과 거리(d)를 계산하거나 상기 길이가 m ($1 \leq m \leq n/2$)인 문자열(T)의 γ -근사주기들 중 최소 γ -근사주기와 관련된 문자열(P)과 거리(d)를 계산하는 단계

를 포함하고,

상기 문자열(T)에 대한 최소 δ -근사주기는, 상기 문자열(T)의 δ -근사주기들 중 거리가 최소이고 가장 짧은 문자열을 의미하고, 상기 문자열(T)에 대한 최소 γ -근사주기는, 상기 문자열(T)의 γ -근사주기들 중 거리가 최소

이고, 길이가 가장 짧은 문자열을 의미하는 것을 포함하는 근사주기 계산 방법.

청구항 4

제3항에 있어서,

상기 길이가 m ($1 \leq m \leq n/2$)인 문자열(T)의 δ -근사주기 또는 γ -근사주기를 계산하는 단계는,

상기 문자열 $T(|T|=n)$ 를 길이 m 으로 분할했을 때 생성되는 부분문자열들을 주기 블록이라 하고, 각 $T[(k-1)m+1] (1 \leq k \leq \lceil n/m \rceil)$ 에서 시작하는 k 번째 주기블록을 T_k 라고 할 때,

상기 δ -근사주기의 길이를 1부터 $n/2$ 까지 증가시켜가면서, ComputeDeltaAP 함수를 이용하여 상기 길이에 대한 δ -근사주기를 계산하는 단계를 포함하고,

상기 ComputeDeltaAP 함수는 상기 문자열(T)와 근사주기의 길이 m 이 주어지면, $T_k[j] (1 \leq k \leq \lceil n/m \rceil)$ 들의 오차를 최소화하기 위해 최소값과 최대값의 산술평균을 이용하여 길이가 m 일 때, 최소의 거리(δ_m)와 δ -근사주기를 계산하는, 근사주기 계산 방법.

청구항 5

제3항에 있어서,

상기 길이가 $m(1)$ 인 문자열(T)의 δ -근사주기 또는 γ -근사주기를 계산하는 단계는,

상기 문자열 $T(|T|=n)$ 를 길이 m 으로 분할했을 때 생성되는 부분문자열들을 주기 블록이라 하고, 각 $T[(k-1)m+1] (1 \leq k \leq \lceil n/m \rceil)$ 에서 시작하는 k 번째 주기블록을 T_k 라고 할 때,

상기 γ -근사주기의 길이를 1부터 $n/2$ 까지 증가시키면서, ComputeGammaAP 함수를 이용하여 상기 길이에 대한 γ -근사주기를 계산하는 단계를 포함하고,

상기 ComputeGammaAP 함수는, 상기 문자열(T)와 근사주기의 길이 m 이 주어지면, $T_k[i] (1 \leq k \leq \lceil n/m \rceil)$ 들의 오차의 합을 최소화하기 위해 중앙값(Median)을 이용하여, 길이가 m 일 때 최소의 거리(γ_m)와 γ -근사주기를 계산하는 근사주기 계산 방법.

청구항 6

제5항에 있어서,

상기 중앙값은 t 개의 정수들을 오름차순으로 정렬했을 때, t 가 홀수일 때 $\lfloor t/2 \rfloor + 1$ 번째의 값, t 가 짝수일 때 $t/2$ 번째와 $(t/2)+1$ 번째 값의 산술평균인 근사주기 계산 방법.

발명의 설명

기술 분야

[0001] 아래의 설명은 정수문자집합에 대한 문자열의 δ -근사주기와 γ -근사주기를 계산하는 기술에 관한 것이다.

배경 기술

[0002]반복적인 문자열에 대한 연구는 데이터압축, 생물정보학, 컴퓨터지원 음악분석 등 많은 분야에서 연구되어 왔다. 반복(Repetition)은 문자열 T 가 주어졌을 때, 연속적으로 나타나는 T 의 부분문자열을 말한다. 예를 들면, $T=aababab$ 라 하면, aa 와 $ababab$ 는 T 의 반복이다.

[0003]이때, $a^2(=aa)$ 은 스퀘어(Square), $(ab)^3(=ababab)$ 은 큐브(Cube)라 부른다. 문자열 P 를 r 번 반복적으로 연결시킨 문자열을 P^r 이라 하자. 문자열 T 가 주어졌을 때, $r>0$ 이고, P' 이 P 의 접두사(Prefix)일 때, $T=P^r P'$ 를 만족하면, P 를 T 의 주기(Period)라 한다. 예를 들면, $T=abcabcab$ 의 주기는 abc , $abcabc$, T 이다. 주기 중 가장 짧은 문자열을 최단 주기라 한다. 다시 말해서, abc 는 T 의 최단 주기이다. 만약 T 에 $|P| \leq |T|/2$ 인 주기 P 가 존재하면, T 는 주기적(Periodic)이라 한다.

[0004]목적함수로 거리함수를 이용하여 거리반경 기반 문자열의 근사주기를 정의하고, 세 개의 문제를 제안하는 기술이 존재한다. 상기 기술은 두 문제에 대해 다항시간에 해결하는 알고리즘을 제시하고, 세 번째 문제에 대해 NP-완전임을 증명하였다.

[0005]또 다른 기술에서는 거리함기반 문자열의 근사주기를 정의하고, 문자열 P 와 X 가 주어졌을 때, X 에 대한 P 의 최소 근사주기거리를 가중편집거리에 대해 $O(mn^2)$ 시간, 편집거리에 대해 $O(mn)$ 시간, 해밍거리에 대해 $O(n)$ 시간에 계산하는 알고리즘을 제시하였다.

[0006]한편, 정수문자로 구성된 문자열에 대한 근사문자열매칭(Approximate String Matching)의 연구도 진행되고 있다. 예를 들면, 음의 높이인 음고(Pitch)는 MIDI 숫자들로 표현될 수 있고, 음과 음 사이의 음높이 차이인 음정(Pitch Interval)은 반음의 수로 표현될 수 있기 때문에 음악서열을 정수문자집합에 대한 문자열로 볼 수 있다. 이와 비슷하게 주가도 정수문자집합에 대한 문자열로 볼 수 있다.

[0007]더 나아가, δ -근사패턴매칭(Approximate Pattern Matching)과 (δ, γ) -근사패턴매칭, δ -근사스퀘어(Approximate Square)와 (δ, γ) -근사스퀘어를 찾는 문제를 정의하고 이를 해결하는 알고리즘, δ -근사반복(Approximate Repetition)과 (δ, γ) -근사반복에 관한 문제들을 정의하고 각 문제를 해결하는 알고리즘, δ -근사패턴매칭문제에 대해 Boyer-Moore 알고리즘 기반의 알고리즘, (δ, γ) -매칭과 관련된 새로운 문제들을 정의하고 이에 대해 FFT(Fast Fourier Transform)를 적용한 알고리즘 및 무관문자(Don't Care Symbol)를 포함한 (δ, γ) -매칭 문제를 연구하는 기술이 제안되어 왔다.

발명의 내용

해결하려는 과제

[0008] 본 발명은 δ -근사와 γ -근사를 이용하여 정수문자집합에 대한 문자열의 δ -근사주기 및 γ -근사주기를 정의한다. 또한, 본 발명은 최소 δ -근사주기와 최소 γ -근사주기를 탐색하는 문제를 정의하고, 이를 각각 $O(n^2)$ 시간에 해결하는 알고리즘을 제공한다.

과제의 해결 수단

[0009] 근사주기 계산 방법은, 정수문자로 구성된 길이가 n 인 문자열(T)에 대하여, 길이가 $n/2$ 인 배열을 생성하는 단계; 상기 길이가 m ($1 \leq m \leq n/2$)인 문자열(T)의 δ -근사주기 또는 γ -근사주기를 계산하는 단계; 및 상기 δ -근사주기 또는 γ -근사주기와 관련된 문자열(P)을 획득하는 단계를 포함할 수 있다.

[0010] 상기 문자열 $T(\in \Sigma^n)$ 와 $P(\in \Sigma^m, m \leq n/2)$ 가 주어졌을 때, $j(1 \leq j \leq m)$ 에 대해 $((i-1) \bmod m) + 1 = j$ 인 $i(1 \leq i \leq n)$ 의 집합을 E_j 라고 하면, 상기 $j(1 \leq j \leq m)$, $k(\in E_j)$ 에 대해 $|T[k] - P[j]| \leq d$ 이면, 상기 문자열(P)은 상기 문자열(T)에 대해 거리가 d 인 δ -근사주기이고, 상기 $j(1 \leq j \leq m)$, $k(\in E_j)$ 에 대해 $\sum_{j=1}^m \sum_{k \in E_j} |T[k] - P[j]| \leq d$ 를 만족하면, 상기 문자열(P)은 상기 문자열(T)에 대해 거리가 d 인 γ -근사주기일 수 있다.

[0011] 상기 길이가 m ($1 \leq m \leq n/2$)인 문자열(T)의 δ -근사주기 또는 γ -근사주기를 계산하는 단계는, 상기 길이가 m ($1 \leq m \leq n/2$)인 문자열(T)의 δ -근사주기를 중 최소 δ -근사주기와 관련된 문자열(P)과 거리(d)를 계산하거나 상기 길이가 m ($1 \leq m \leq n/2$)인 문자열(T)의 γ -근사주기를 중 최소 γ -근사주기와 관련된 문자열(P)과 거리(d)를 계산하는 단계를 포함하고, 상기 문자열(T)에 대한 최소 δ -근사주기는, 상기 문자열(T)의 δ -근사주기를 중 거리가 최소이고 가장 짧은 문자열을 의미하고, 상기 문자열(T)에 대한 최소 γ -근사주기는, 상기 문자열(T)의 γ -근사주기를 중 거리가 최소이고, 길이가 가장 짧은 문자열을 의미할 수 있다.

[0012] 상기 길이가 m ($1 \leq m \leq n/2$)인 문자열(T)의 δ -근사주기 또는 γ -근사주기를 계산하는 단계는, 상기 문자열 $T(|T|=n)$ 를 길이 m 으로 분할했을 때 생성되는 부분문자열들을 주기 블록이라 하고, 각 $T[(k-1)m+1](1 \leq k \leq \lceil n/m \rceil)$ 에서 시작하는 k 번째 주기블록을 T_k 라고 할 때, 상기 δ -근사주기의 길이를 1부터 $n/2$ 까지 증가시켜가면서, ComputeDeltaAP 함수를 이용하여 상기 길이에 대한 δ -근사주기를 계산하는 단계를 포함하고, 상기 ComputeDeltaAP 함수는 상기 문자열(T)과 근사주기의 길이 m 이 주어지면, $T_k[j](1 \leq k \leq \lceil n/m \rceil)$ 들의 오차를 최소화하기 위해 최소값과 최대값의 산술평균을 이용하여 길이 m 일 때, 최소의 거리(δ_m)와 δ -근사주기를 계산할 수 있다.

[0013] 상기 길이가 $m(1)$ 인 문자열(T)의 δ -근사주기 또는 γ -근사주기를 계산하는 단계는, 상기 문자열 $T(|T|=n)$ 를 길이 m 으로 분할했을 때 생성되는 부분문자열들을 주기 블록이라 하고, 각 $T[(k-1)m+1] (1 \leq k \leq \lceil n/m \rceil)$ 에서 시작하는 k 번째 주기블록을 T_k 라고 할 때, 상기 γ -근사주기의 길이를 1부터 $n/2$ 까지 증가시키면서, ComputeGammaAP 함수를 이용하여 상기 길이에 대한 γ -근사주기를 계산하는 단계를 포함하고, 상기 ComputeGammaAP 함수는, 상기 문자열(T)과 근사주기의 길이 m 이 주어지면, $T_k[i] (1 \leq k \leq \lceil n/m \rceil)$ 들의 오차의 합을 최소화하기 위해 중앙값(Median)을 이용하여, 길이가 m 일 때 최소의 거리(γ_m)와 γ -근사주기를 계산할 수 있다.

[0014] 상기 중앙값은 t 개의 정수들을 오름차순으로 정렬했을 때, t 가 홀수일 때 $\lfloor t/2 \rfloor + 1$ 번째의 값, t 가 짝수일 때 $t/2$ 번째와 $(t/2)+1$ 번째 값의 산술평균일 수 있다.

발명의 효과

[0015] 본 발명은 정수문자집합에 대한 문자열의 δ -근사주기와 γ -근사주기를 정의하고, 최소 δ -근사주기와 최소 γ -근사주기를 찾는 문제와 각 문제를 $O(n^2)$ 시간에 해결할 수 있다.

[0016] 본 발명은 주어진 문자열에 무관문자가 포함된 경우, 본 발명에서 제안하는 알고리즘을 이용하여 최소 δ -근사주기와 최소 γ -근사주기를 찾을 수 있다.

[0017] 또한, 본 발명은 정수로 표현된 문자열에 대한 (δ, γ) -매칭은 음악 서열이나 주가 연구에 응용될 수 있다.

[0018] 또한, 본 발명은 실수로 구성된 문자열에 대해서도 보다 쉽게 적용할 수 있다.

도면의 간단한 설명

[0019] 도 1은 일 실시예에 따른 δ -근사주기를 계산하는 방법을 설명하기 위한 예이다.

도 2는 일 실시예에 따른 γ -근사주기를 계산하는 방법을 설명하기 위한 예이다.

도 3은 일 실시예에 따른 길이가 m 이고 거리 d 가 최소인 δ -근사주기를 계산하는 방법을 설명하기 위한 예이다.

도 4는 일 실시예에 따른 길이가 m 이고 거리 d 가 최소인 γ -근사주기를 계산하는 방법을 설명하기 위한 예이다.

도 5는 일 실시예에 따른 최소 δ -근사주기를 계산하는 방법을 설명하기 위한 흐름도이다.

도 6은 일 실시예에 따른 최소 γ -근사주기를 계산하는 방법을 설명하기 위한 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0020] 이하, 실시예를 첨부한 도면을 참조하여 상세히 설명한다.

[0021] 유한한 문자집합 Σ 의 문자들로 구성된 문자열들의 집합을 Σ^* 로 나타내며, 길이가 m 인 문자열들의 집합을 Σ^m 으로 나타낸다. 문자열 X 의 길이는 $|X|$ 로 표기하며, X 의 i 번째 ($1 \leq i \leq |X|$) 문자는 $X[i]$ 로 표기한다. X 의 i 번째 문자부터 j 번째 문자까지의 부분문자열인 $X[i] X[i+1] \dots X[j]$ ($1 \leq j \leq |X|$)는 $X[i..j]$ 로 표기하고, $X=WU$ 이면, W 를 X 의 접두사, U 를 X 의 접미사(Suffix)라 한다. $T[i..j]$ 가 $i > j$ 이거나 $T[0]$ 일 때 공백 문자열로 정의하고 이를 ϵ 로 표기한다. 무관문자는 자신을 포함한 모든 문자와 일치하는 문자이며, $\star$ 로 표기한다. 다시 말해서, $a \in \Sigma \cup \{\star\}$ 일 때, $\star = a$ 이다.

[0022] 문자열 X 와 P , 거리함수 d 가 주어졌을 때, 다양한 거리함수에 대한 근사주기를 다음과 같이 정의할 수 있다. $X = P_1 P_2 \dots P_r$ ($P_i \neq \epsilon$) 이고 P' 을 P 의 접두사라고 할 때, $d(P, P_i) \leq t$ ($1 \leq i < r$) 이고, $d(P', P_r) \leq t$ 이면, P 를 X 의 t -근사주기 또는 거리가 t 인 X 의 근사주기라고 한다. 종래의 기술에 따르면, 다양한 거리함수 d , 다시 말해서, 해밍거리, 편집거리, 메트릭 가중편집거리에 대해 P 가 X 의 t -근사주기를 만족하는 최소 근사주기거리 t 를 찾는 알고리즘이 제시되었다.

[0023] 정수문자집합 Σ 의 두 문자 a, b 와 정수 δ 가 주어졌을 때, $|a-b| \leq \delta$ 이면, a 와 b 는 δ -근사라고 하고(역도 성립), $a =_{\delta} b$ 문자열 X 와 Y ($|X|=|Y|$), 정수 δ, γ 가 주어졌을 때, $X[i] =_{\delta} Y[i]$ ($1 \leq i \leq |X|$) 이면, X 와 Y 는 δ -근사라고 하고(역도 성립), $X =_{\delta} Y$ 로 표기할 수 있다.

[0024] 만약, $\sum_{i=1}^{|X|} |X[i] - Y[i]| \leq \gamma$ 이면, X 와 Y 는 γ -근사라고 하고(역도 성립), $X =_{\gamma} Y$ 로 표기한다. 만약, X 와 Y 가 δ -근사이며 동시에 γ -근사이면, X 와 Y 는 (δ, γ) -근사라고 하고, $X =_{\delta, \gamma} Y$ 로 표기한다.

[0025] 도 1은 일 실시예에 따른 δ -근사주기를 계산하는 방법을 설명하기 위한 예이고, 도 2는 일 실시예에 따른 γ -근사주기를 계산하는 방법을 설명하기 위한 예이다.

[0026] 문자열 $T(\in \Sigma^n)$ 와 $P(\in \Sigma^m, m \leq n/2)$ 가 주어졌을 때, 각 $j(1 \leq j \leq m)$ 에 대해 $((i-1) \bmod m) + 1 = j$ 인 $i(1 \leq i \leq n)$ 의 집합을 E_j 라고 가정한다. 이때, 모든 $j(1 \leq j \leq m)$, $k(\in E_j)$ 에 대해 $|T[k] - P[j]| \leq d$ 이면, P 는 T 에 대해 거리가 d 인 δ -근사주기라고 정의한다. 또한, 모든 $j(1 \leq j \leq m)$, $k(\in E_j)$ 에 대해 $\sum_{j=1}^m \sum_{k \in E_j} |T[k] - P[j]| \leq d$ 를 만족하면, P 는 T 에 대해 거리가 d 인 γ -근사주기라 정의한다.

[0027] 도 1 및 도 2를 참고하면, 예를 들면, $T=(17, 12, 21, 16, 14, 20, 23, 7, 25, 19, 15)$ 가 주어질 수 있다. 이때, $P=(17, 13, 20)$ 은 T 에 대해 거리가 6인 δ -근사주기이며, $P=(19, 11, 22)$ 는 T 에 대해 거리가 4인 δ -근사주기이다.

- [0028] 또한, $P=(19, 11, 22)$ 는 T 에 대해 거리가 27인 γ -근사주기이며, $P=(18, 13, 21)$ 은 T 에 대해 거리가 24인 γ -근사주기이다. 문자열(T)만 주어졌을 때 문자열(T)에 대하여 복수의 δ -근사주기 및 γ -근사주기가 존재할 수 있다. 문자열(T)의 δ -근사주기들 중 거리가 최소이고, 가장 짧은 문자열을 T 에 대한 최소 δ -근사주기라고 정의하며, T 의 γ -근사주기들 중 거리가 최소이고, 길이가 가장 짧은 문자열을 T 에 대한 최소 γ -근사주기라고 정의한다.
- [0029] 도 3은 일 실시예에 따른 길이가 m 이고 거리 d 가 최소인 δ -근사주기를 계산하는 방법을 설명하기 위한 예이고, 도 4는 일 실시예에 따른 길이가 m 이고 거리 d 가 최소인 γ -근사주기를 계산하는 방법을 설명하기 위한 예이다.
- [0030] 아래의 설명에서는 문제 1 내지 4를 해결하는 알고리즘들을 제시할 수 있다. 단, Σ 는 유한한 정수문자집합으로 가정한다.
- [0031] 주어진 문자열 $T(|T|=n)$ 를 길이 m 으로 분할했을 때 생성되는 부분문자열들을 주기 블록이라 하고, 각 $T[(k-1)m+1](1 \leq k \leq \lceil n/m \rceil)$ 에서 시작하는 k 번째 주기블록을 T_k 라고 하자.
- [0032] 문제 1. 주어진 $T(\in \Sigma^*)$ 의 최소 δ -근사주기 $P(\in \Sigma^m)$ 찾기.
- [0033] 도 5를 참고하면, 최소 δ -근사주기를 계산하는 방법을 설명하기 위한 흐름도로서, 단계(510 내지 530)을 수행할 수 있다. 이때, 근사주기 계산 시스템에 의하여 최소 δ -근사주기가 계산될 수 있다.
- [0034] 단계(510)에서 근사주기 계산 시스템은 정수문자로 구성된 길이가 n 인 문자열(T)에 대하여 길이가 $n/2$ 인 배열 각각을 생성할 수 있다.
- [0035] 단계(520)에서 근사주기 계산 시스템은 길이가 $m(1 \leq m \leq n/2)$ 인 문자열(T)의 δ -근사주기들 중 최소 δ -근사주기와 관련된 문자열(P)의 길이와 거리(d)를 계산할 수 있다.
- [0036] 단계(530)에서 근사주기 계산 시스템은 문자열(T)의 최소 δ -근사주기와 관련된 문자열(P)를 획득할 수 있다.
- [0037] 단계(510 내지 단계530)에 대한 설명은 아래의 설명을 통하여 더욱 상세하게 설명하기로 한다.
- [0038] 문제 1은 알고리즘 1을 이용하여 해결할 수 있다. 알고리즘 1은 δ -근사주기의 길이를 1부터 $n/2$ 까지 증가시켜 가면서, ComputeDeltaAP 함수를 이용하여 상기 길이에 대한 δ -근사주기를 계산한다.

```

ComputeDeltaAP (T, A, B, P, n, m)
1   for j = 1 to m
2       A[j] = B[j] = T[j]
3       j = 0
4   for i = m + 1 to n
5       j = ((i - 1) mod m) + 1
6       if T[i] < A[j] then
7           A[j] = T[i]
8       if T[i] > B[j] then
9           B[j] = T[i]
10  for j = 1 to m
11      P[j] = (A[j] + B[j])/2
12  return  $\max_{1 \leq i \leq m} (|B[i] - \lfloor P[i] \rfloor|)$ 

```

```

Algorithm 1 (T)
1   n = T.length
2   let A[1..n/2], B[1..n/2], P[1..n/2] be ne arrays
3    $\delta' = \infty$  // 최소  $\delta$  - 근사주기의 거리
4    $m' = 0$  // 최소  $\delta$  - 근사주기의 길이
5    $\delta_m = 0$ 
6   for m = 1 to n/2
7        $\delta_m = \text{ComputeDeltaAP}(T, A, B, P, n, m)$ 
8       if  $\delta_m < \delta'$  then
9            $\delta' = \delta_m$ 
10           $m' = m$ 
11   $\delta_m = \text{ComputeDeltaAP}(T, A, B, P, n, m')$ 
12  return P

```

[0039]

[0040] δ -근사주기의 각 문자 $P[j]$ ($1 \leq j \leq m$)를 계산하기 위해, ComputeDeltaAP 함수는 문자열(T)와 근사주기의 길이 m이 주어지면, $T_k[j]$ ($1 \leq k \leq \lceil n/m \rceil$)들의 오차를 최소화하기 위해 최소값과 최대값의 산술평균을 이용하여 길이가 m일 때, 최소의 거리(δ_m)와 δ -근사주기를 계산한다.

[0041]

구체적으로, i ($1 \leq i \leq n$)에서 $T[i]$ (주기블록 $T_k[j]$, $j = ((i-1) \bmod m) + 1$, $k = \lfloor (i-1)/m \rfloor + 1$)로부터 최소값과 최대값을 각각 A[j], B[j]에 저장한다(1번째 줄부터 9번째 줄). 각 P[j]는 $(A[j] + B[j])/2$ 로, δ_m 는 $\max_{1 \leq i \leq m} (|B[i] - \lfloor P[i] \rfloor|)$ 로 계산될 수 있다(10번째 줄부터 12번째 줄).

[0042]

예를 들면, 도 3을 참고하면, T=(17, 12, 21, 16, 14, 20, 23, 7, 25, 19, 15)이고, m=3일 때, T₁=(17, 12, 21), T₂=(16, 14, 20), T₃=(23, 7, 25), T₄=(9, 15)이고, A=(16, 7, 20), B=(23, 15, 25)이고, δ -근사주기 P=(19.5, 11, 22.5), $\delta_3=4$ 가 된다. 알고리즘 1은 크기가 n/2인 배열 A, B, P를 할당하고, ComputeDeltaAP 함수

수를 이용하여 최소 δ -근사주기의 거리(δ')와 길이(m')를 계산한다(6번째 줄부터 10번째 줄). m' 이 계산됨에 따라 m' 을 이용하여 최소 δ -근사주기를 다시 계산한다(11번째 줄). 최소 δ -근사주기는 정수문자로 구성되어야 하므로 각 문자 $P[j]$ 의 소수부분을 올림하거나 내림한 정수로 변환하여 출력한다. 이때, 정수 조건이 없으면 변환하지 않은 상태로 출력한다. 앞서 설명한 바와 같이, $m'=3$ 이며, 최소 δ -근사주기는 정수 변환전(19.5, 11, 22.5)이며, 변환 후(19, 11, 22), (20, 11, 22), (19, 11, 23), (20, 11, 23) 중 적어도 하나가 출력될 수 있다. 알고리즘 1은 $O(n)$ 시간에 수행되는 ComputeDeltaAP함수를 $(n/2)+1$ 번 호출하기 때문에 $O(n^2)$ 시간에 수행될 수 있다.

[0043] 문제 2. 주어진 $T(\in \Sigma^*)$ 의 최소 γ -근사주기 $P(\in \Sigma^m)$ 찾기.

[0044] 도 6을 참고하면, 최소 γ -근사주기를 계산하는 방법을 설명하기 위한 흐름도로서, 단계(610 내지 630)을 수행할 수 있다. 이때, 근사주기 계산 시스템에 의하여 최소 γ -근사주기가 계산될 수 있다.

[0045] 단계(610)에서 근사주기 계산 시스템은 정수문자로 구성된 길이가 n 인 문자열(T)에 대하여 길이가 $n/2$ 인 배열을 생성할 수 있다.

[0046] 단계(620)에서 근사주기 계산 시스템은 길이가 m ($1 \leq m \leq n/2$)인 문자열(T)의 γ -근사주기를 중 최소 γ -근사주기와 관련된 문자열(P)의 길이와 거리(d)를 계산할 수 있다.

[0047] 단계(630)에서 근사주기 계산 시스템은 문자열(T)의 최소 γ -근사주기와 관련된 문자열(P)를 획득할 수 있다.

[0048] 단계(610 내지 단계630)에 대한 설명은 아래의 설명을 통하여 더욱 상세하게 설명하기로 한다.

[0049] 문제 2는 알고리즘 2를 이용하여 해결할 수 있다. 알고리즘 2는 γ -근사주기의 길이를 1부터 $n/2$ 까지 증가시키면서, ComputeGammaAP 함수를 이용하여 상기 길이에 대한 γ -근사주기를 계산할 수 있다.

ComputeMedian (T, n, m, offset)

```

1  k = ⌊n/m⌋
2  if (n mod m) ≥ offset then
3      k = k + 1
4  let A[1..k] be new arrays
5  j = 0
6  for i = 1 to k
7      A[i] = T[j + offset]
8      j = j + m
9  Sort(A, 1, k)
10 if k is even then
11     return ( A[k/2] + A[k/2 + 1])/2
12 else
13     return A[⌊k/2⌋ + 1]
```

ComputeGammaAP (T, P, n, m)

```

1  for i = 1 to m
2      P[i] = ComputeMedian (T, n, m, i)
3  return  $\sum_{i=1}^n \lceil |T[i] - P[(i-1) \bmod m] + 1| \rceil$ 
```

Algorithm 2 (T)

```

1  n = T.length
2  let a[1..n/2] be ne arrays
3   $\gamma' = \infty$  // 최소  $\gamma$ - 근사주기의 거리
4   $m' = 0$  // 최소  $\gamma$ - 근사주기의 길이
5   $\gamma_m = 0$ 
6  for m = 1 to n/2
7       $\gamma_m = \text{ComputeGammaAP}(T, A, B, P, n, m)$ 
8      if  $\gamma_m < \gamma'$  then
9           $\gamma' = \gamma_m$ 
10          $m' = m$ 
11  $\gamma_m = \text{ComputeGammaAP}(T, A, B, P, n, m')$ 
12 return P
```

[0050]

[0051]

γ -근사주기의 각 문자 $P[i]$ ($1 \leq i \leq m$)를 계산하기 위하여, ComputeGammaAP 함수는 문자열(T)와 근사주기의 길이 m이 주어지면, $T_k[i]$ ($1 \leq k \leq \lceil n/m \rceil$)들의 오차의 합을 최소화하기 위해 중앙값(Median)을 이용하여, 길이가 m일 때 최소의 거리(γ_m)와 γ -근사주기를 계산할 수 있다. 중앙값은 t개의 정수들을 오름차순으로 정렬했을 때, t가 홀수일 때는 $\lfloor t/2 \rfloor + 1$ 번째의 값, t가 짝수일 때는 t/2 번째와 (t/2)+1번째 값의 산술평균이다. t개의 실수들로 구성된 집합 S에 대해 x 가 S의 중앙값이면, $\sum_{s \in S} |s - x|$ 는 최소이다. 구체적으로 i ($1 \leq i \leq m$)에서 주기블록 $T_k[i]$ ($1 \leq k \leq \lceil n/m \rceil$)들의 중앙값을 계산한다. 각 $P[i]$ 는 ComputeMedian 함수를 통해 계산되며(1번째 줄부터 2번째 줄), γ_m 은 $\sum_{i=1}^n \lceil |T[i] - P[(i-1) \bmod m] + 1| \rceil$ 으로 계산될 수 있다(3번째 줄). 예를 들면, 도 4를 참고하면,

$T=(17, 12, 21, 16, 14, 20, 23, 7, 25, 19, 15)$ 이고 $m=3$ 일 때, $P[1]$ 은 $(17, 16, 23, 19)$ 의 중앙값인 18, $P[2]$ 는 $(12, 14, 7, 15)$ 의 중앙값인 13, $P[3]$ 은 $(21, 20, 25)$ 의 중앙값인 21이 된다. 이에 따라 γ -근사주기 $P=(18, 13, 21)$ 이고, $\gamma_3=24$ 가 된다.

[0052] 알고리즘 2는 γ -근사주기를 저장할 크기가 $n/2$ 인 배열 P 를 할당하고, ComputeGammaAP 함수를 이용하여 최소 γ -근사주기의 거리(γ')와 길이(m')를 계산한다(6번째 줄부터 10번째 줄). m' 이 계산됨에 따라 m' 를 이용하여 최소 γ -근사주기를 다시 계산한다(11번째 줄). 최소 γ -근사주기는 정수문자로 구성되어야 하므로 각 문자 $P[j]$ 의 소수부분을 올림하거나 내림한 정수로 변환하여 출력할 수 있다. 이때, 정수 조건이 없으면 변환하지 않고 그대로 출력하면 된다. 앞서 설명한 것과 같이 $m'=3$ 이고, 최소 γ -근사주기는 $(18, 13, 21)$ 이다.

[0053] 알고리즘 2의 복잡도를 분석하면 다음과 같다. ComputeMedian 함수에서 중앙값은 정렬알고리즘을 이용하여 $O((n/m)\log(n/m))$ 시간에 계산할 수 있다. ComputeGammaAP 함수의 시간복잡도는 ComputeMedian 함수를 m 번 호출하기 때문에 $O(n\log(n/m))$ 이다. 알고리즘 2는 m ($1 \leq m \leq n/2$)에 대해 ComputeGammaAP 함수를 $n/2$ 번 호출한다(6번째 줄부터 10번째 줄). 이에 따라 알고리즘 2의 시간복잡도는 $\sum_{i=1}^{n/2} O(n\log(n/i))$ 이며, 아래의 과정에 의하여 $O(n^2)$ 임을 알 수 있다.

[0054] 수학적 식 1:

$$\begin{aligned} n \sum_{i=1}^{n/2} \left(\log\left(\frac{n}{i}\right) \right) \\ &= n \left\{ \log\left(\frac{n}{1}\right) + \log\left(\frac{n}{2}\right) + \log\left(\frac{n}{3}\right) + \dots + \log\left(\frac{n}{(n/2)}\right) \right\} \\ &= n \left\{ \log\left(\frac{n^{n/2}}{(n/2)!}\right) \right\} = n \left\{ \frac{n}{2} \log n - \log\left(\frac{n}{2}\right)! \right\} \end{aligned}$$

[0055] [0056] stirling의 근사에 의해, $n! \geq \frac{n^n}{e^n}$ 이다. $\frac{n^n}{e^n} \geq \frac{n^n}{4^n}$ 이므로, $n! \geq \frac{n^n}{4^n}$ 이다.

[0057] 따라서, $\left(\frac{n}{2}\right)! \geq \frac{\left(\frac{n}{2}\right)^{n/2}}{4^{n/2}}$ 이므로 양변에 $\log$ 를 취하면, $\log\left(\left(\frac{n}{2}\right)!\right) \geq \frac{n}{2} \log\left(\frac{n}{2}\right) - n$ 이다.

[0058] 수학적 식 2:

$$\begin{aligned} n \left\{ \frac{n}{2} \log n - \log\left(\frac{n}{2}\right)! \right\} &\leq n \left\{ \frac{n}{2} \log n - \left(\frac{n}{2} \log\left(\frac{n}{2}\right) - n \right) \right\} \\ &= n \left\{ \frac{n}{2} \log n - \left(\frac{n}{2} \log n - \frac{n}{2} - n \right) \right\} = n \times \frac{3n}{2} \end{aligned}$$

[0059] [0060] 수학적 식 2를 수학적 식 1에 대입하면, 수학적 식 3으로 도출될 수 있다.

[0061] 수학적식 3:

$$n \sum_{i=1}^{n/2} \left(\log\left(\frac{n}{i}\right) \right) \leq \frac{3n^2}{2} \in O(n^2)$$

[0062]

[0063] 만약, ComputeMedian 함수에서 정렬알고리즘이 아닌, 선택알고리즘(Selection Algorithm)을 이용하면, 알고리즘 2의 시간복잡도가 $O(n^2)$ 임을 보다 쉽게 판단할 수 있다. ComputeMedian 함수에서 중앙값은 선택알고리즘을 이용할 경우, $O(n/m)$ 시간에 계산할 수 있다.

[0064]

이에 따라 ComputeGammaAP 함수의 시간복잡도는 $O(n)$ 이고, 알고리즘 2는 ComputeGammaAP 함수를 $(n/2)+1$ 번 호출하기 때문에, 시간복잡도는 $O(n^2)$ 이다.

[0065]

더 나아가, 문제 1 및 문제 2를 해결하는 알고리즘에 기반하여 시간복잡도의 증가없이 문제 3 및 문제 4를 해결할 수 있다.

[0066]

문제 3. 주어진 $T(\in (\Sigma \cup \{\star\})^*)$ 의 최소 δ -근사주기 $P(\in \Sigma^m)$ 찾기.

[0067]

문제 4. 주어진 $T(\in (\Sigma \cup \{\star\})^*)$ 의 최소 γ -근사주기 $P(\in \Sigma^m)$ 찾기.

[0068]

문제 3의 경우, ComputeDeltaAP 함수를 ComputeDeltaAPwithDC 함수로 변경하여 무관문자를 고려한 최소 δ -근사주기를 획득할 수 있다. 마찬가지로, 문제 4의 경우, T[i]가 $\star$ 인 경우, 중앙값 및 γ_m 계산에서 제외하기 위하여 ComputeMedian 함수와 ComputeGammaAP 함수를 ComputeMedianwithDC 함수와 ComputeGammaAPwithDC 함수로 변경하여 무관문자를 고려한 최소 γ -근사주기를 획득할 수 있다.

```

ComputeDeltaAPwithDC (T, A, B, P, n, m)
1  for j = 1 to m
2    A[j] = B[j] = T[j]
3    j = 0
4  for i = m + 1 to n
5    j = ((i - 1) mod m) + 1
6    if T[i] != * and T[i] ≤ A[j] then
7      A[j] = T[i]
8    if T[i] != * and T[i] ≥ B[j] then
9      B[j] = T[i]
10   δm = 0
11   δ* = δm
12   for j = 1 to m
13     if A[j] != * then
14       P[j] = (A[j] + B[j])/2
15       δ* = |B[j] - ⌊P[j]⌋|
16       if δ* > δm then
17         δm = δ*
18     else
19       P[j] = *
20   return δm

ComputeMedianwithDC (T, n, m, offset )
1  k = ⌊n/m⌋
2  if ( n mod m ) ≥ offset then
3    k = k + 1
4  let A[1..k] be new arrays
5  j = 0
6  for i = 1 to k
7    if T[j + offset] != * then
8      A[i] = T[j + offset]
9    else
10     k = k - 1
11     i = i - 1
12     j = j + m
13  if k = 0 then
14    return *
15  Sort(A, 1, k)
16  if k is even then
17    return (A[k/2] + A[k/2 + 1])/2
18  else
19    return A[⌊k/2⌋+1]

ComputeGammaAPwithDC (T, P, n, m)
1  for i = 1 to m
2    P[i] = ComputeMedianwithDC(T, n, m, i)
3  γm = 0
4  j = 0
5  for i = 1 to n
6    j = ((i - 1) mod m) + 1
7    if P[j] != * then
8      γm = γm + |T[i] - P[j]|
9  return γm

```

[0069]

[0070]

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.

[0071]

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 영구적으로, 또는 일시적으로 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.

[0072] 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.

[0073] 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.

[0074] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

도면1



도면2



도면3

- $T = (17, 12, 21, 16, 14, 20, 23, 7, 25, 19, 15)$, $m = 3$

	T_1			T_2			T_3			T_4	
T	17	12	21	16	14	20	23	7	25	19	15
	m										

T_1	17	12	21
T_2	16	14	20
T_3	23	7	25
T_4	19	15	

A	16	7	20
B	23	15	25

$$A[i] = \min\{T_k[i], 1 \leq k \leq \lfloor n/m \rfloor\}$$

$$B[i] = \max\{T_k[i], 1 \leq k \leq \lfloor n/m \rfloor\}$$

P	19.5	11	22.5
---	------	----	------

$$P[i] = (A[i] + B[i])/2$$

$$\therefore P = (19, 11, 22), (19, 11, 23), (20, 11, 22), (20, 11, 23)$$

$$d = 4 \left(\max_{1 \leq i \leq m} \{ \max(|A[i] - P[i]|, |B[i] - P[i]|) \} \right)$$

도면4

- $T = (17, 12, 21, 16, 14, 20, 23, 7, 25, 19, 15)$, $m = 3$

	T_1			T_2			T_3			T_4	
T	17	12	21	16	14	20	23	7	25	19	15
	m										

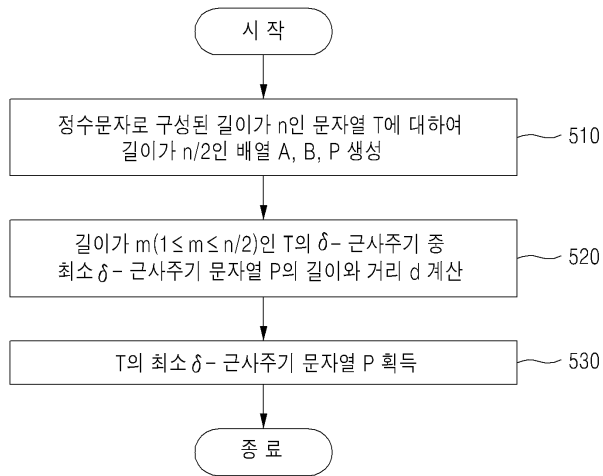
T_1	17	12	21
T_2	16	14	20
T_3	23	7	25
T_4	19	15	

P	18	13	21
---	----	----	----

$$P[i] = \text{median}\{T_k[i], 1 \leq k \leq \lfloor n/m \rfloor\}$$

$$d = 24 \left(\sum_{i=1}^n |T[i] - P[(i-1) \bmod m + 1]| \right)$$

도면5



도면6

