

	(19) 대한민국특허청(KR) (12) 공개특허공보(A)	(11) 공개번호 10-2019-0048100 (43) 공개일자 2019년05월09일
(51) 국제특허분류(Int. Cl.) G06K 9/62 (2006.01)	(71) 출원인 인하대학교 산학협력단 인천광역시 미추홀구 인하로 100(용현동, 인하대학교)	
(52) CPC특허분류 G06K 9/6211 (2013.01) G06K 9/6215 (2013.01)	(72) 발명자 심정섭 인천광역시 남동구 남동대로 860, 120동 601호(간석동, 간석래미안자이아파트) 최지효 인천광역시 남동구 평온로6번길 26, 3층(간석동) 김영호 인천광역시 남구 낙섬동로 6, 102동 301호(용현동, 한국아파트)	
(21) 출원번호 10-2017-0142629 (22) 출원일자 2017년10월30일 심사청구일자 2017년10월30일	(74) 대리인 양정보	

전체 청구항 수 : 총 10 항

(54) 발명의 명칭 사각망 순열패턴매칭을 위한 병렬 알고리즘

(57) 요약

사각망 순열패턴매칭을 위한 병렬 알고리즘이 개시된다. 사각망 순열패턴매칭을 위한 방법은, 상수 시간에 비교 가능한 문자 집합을 나타내는 Σ 상의 텍스트 $T(|T|=n)$ 와 패턴 $P(|P|=m)$ 가 주어짐에 따라, 길이가 m 인 배열 tmp 을 정렬하고 정렬된 배열 tmp 을 이용하여 $\pi(P)$ 와 $\pi^{-1}(P)$ 을 계산하는 전처리단계; 및 상기 텍스트 T 에 대하여 단계 $i(1 \leq i \leq n-m+1)$ 의 각 스텝 $j(i \leq j \leq n)$ 에서 j 를 증가시키면서 상기 패턴 P 와 순위동형인 사각망 부분서열 $\hat{T}$ 을 찾는 탐색단계를 포함한다.

대표도

```

Algorithm 1
Input: Text  $T(|T|=n)$ , Pattern  $P(|P|=m)$ 
Output: Boundary  $R$  of all boxed subsequence  $\hat{T}$  of  $T$  that are order-isomorphic to  $P$ 
1: Preprocessing( $P$ )                                     // Preprocessing
2: parallel for  $i = 1$  to  $n-m+1$                              // Searching
3:   Assign two arrays  $A1_i, A2_i$  of length  $m+1$  for boxed subsequence  $\hat{T}$ 
4:   for  $j = i$  to  $n$ 
5:      $A1_i.insert((T[j],j)), A2_i.insert((j,T[j]))$ 
6:     if  $A1_i.rank((T[i],i)) > \pi(P)[1]$ 
7:       then  $k = A1_i.min().second$ 
8:        $A1_i.delete((T[k],k)), A2_i.delete((k,T[k]))$ 
9:     else if  $A1_i.size() > m$ 
10:      then  $k' = A1_i.max().second$ 
11:       $A1_i.delete((T[k'],k')), A2_i.delete((k',T[k']))$ 
12:     if  $A1_i.find((T[j],j)))$  and  $A1_i.size() == m$ 
13:       if  $\hat{T} \approx P$  then print boundary  $R$  of  $\hat{T}$ 

```

이 발명을 지원한 국가연구개발사업

과제고유번호 2016944565

부처명 미래창조과학부

연구관리전문기관 한국연구재단

연구사업명 핵심연구

연구과제명 [Ezbaro] 시계열데이터 패턴분석을 위한 순위패턴매칭 알고리즘 개발

기 여 율 1/1

주관기관 인하대학교 산학협력단

연구기간 2016.11.01 ~ 2017.10.31

명세서

청구범위

청구항 1

사각망 순열패턴매칭(boxed-mesh permutation pattern matching)을 위한 방법에 있어서,

상수 시간에 비교 가능한 문자 집합을 나타내는 Σ 상의 텍스트 $T(|T|=n)$ 와 패턴 $P(|P|=m)$ 가 주어짐에 따라,

길이가 m 인 배열 tmp 을 정렬하고 정렬된 배열 tmp 을 이용하여 $\pi(P)$ 와 $\pi^{-1}(P)$ 을 계산하는 전처리단계; 및

상기 텍스트 T 에 대하여 단계 $i(1 \leq i \leq n-m+1)$ 의 각 스텝 $j(i \leq j \leq n)$ 에서 j 를 증가시키면서 상기 패턴 P 와 순위동형인 사각망 부분서열 $\hat{T}$ 을 찾는 탐색단계를 포함하는 사각망 순열패턴매칭을 위한 방법.

청구항 2

제1항에 있어서,

상기 전처리단계는,

m 개의 스레드를 이용하여 의사코드 전처리(Preprocessing)을 수행하는 것으로,

상기 배열 tmp 에 짝 $(P[i], i)$ 형태로 저장하고 상기 배열 tmp 을 상기 짝의 첫 번째 원소를 기준으로 병렬 바이트닉 정렬(bitonic sort)을 이용하여 정렬한 후 정렬된 배열 tmp 을 이용하여 상기 $\pi(P)$ 와 상기 $\pi^{-1}(P)$ 을 계산하는 것

을 특징으로 하는 사각망 순열패턴매칭을 위한 방법.

청구항 3

제1항에 있어서,

상기 탐색단계는,

서로 독립적인 각 단계 i 마다 스레드 i 를 이용하여 스텝 j 를 계산하는 것

을 특징으로 하는 사각망 순열패턴매칭을 위한 방법.

청구항 4

제3항에 있어서,

상기 탐색단계는,

상기 사각망 부분서열 $\hat{T}$ 을 관리하기 위해 짝 $(T[j], j)$ 을 저장하는 배열 $A1_i(|A1_i|=m+1)$ 와 짝 $(j, T[j])$ 를 저장하는 배열 $A2_i(|A2_i|=m+1)$ 를 사용

하는 단계;

스텝 j 에서 상기 배열 $A1_i$, $A2_i$ 에 각각 $(T[j], j)$, $(j, T[j])$ 를 삽입하는 단계;

상기 사각망 부분서열 $\hat{T}$ 의 $T[i]$ 의 순위를 $P[1]$ 의 순위와 같도록 상기 사각망 부분서열 $\hat{T}$ 를 보정하는 단계; 및

상기 배열 $A2_i$ 와 상기 $\pi^{-1}(P)$ 에 대해 상기 사각망 부분서열 $\hat{T}$ 이 상기 패턴 P 와 순위동형인지 확인하는 단계

를 포함하는 사각망 순열패턴매칭을 위한 방법.

청구항 5

제4항에 있어서,

상기 보정하는 단계는,

상기 사각망 부분서열 $\hat{T}$ 에서 $T[i]$ 의 순위가 $P[1]$ 의 순위보다 높으면 상기 배열 $A1_i$ 에서 순위가 가장 낮은 원소 $T[k]$ 를 포함하는 짝 $(T[k], k)$ 와 $(k, T[k])$ 를 상기 배열 $A1_i$ 와 $A2_i$ 에서 각각 삭제하고, $T[i]$ 의 순위가 $P[1]$ 의 순위보다 높지 않으면 상기 배열 $A1_i$ 에서 순위가 가장 높은 원소 $T[k']$ 를 포함하는 짝 $(T[k'], k')$ 와 $(k', T[k'])$ 를 상기 배열 $A1_i$ 와 $A2_i$ 에서 각각 삭제하는 단계를 포함하는 사각망 순열패턴매칭을 위한 방법.

청구항 6

제1항에 있어서,

상기 탐색단계는,

$O(n)$ 개의 스레드 그룹 $i (1 \leq i \leq n - m + 1)$ 를 이용하여 단계 i 를 계산하는 것을 특징으로 하는 사각망 순열패턴매칭을 위한 방법.

청구항 7

제6항에 있어서,

상기 탐색단계는,

상기 사각망 부분서열 $\hat{T}$ 을 관리하기 위해 짝 $(T[j], j)$ 을 저장하는 배열 $A1_i (|A1_i| = m + 1)$ 와 짝 $(j, T[j])$ 를 저장하는 배열 $A2_i (|A2_i| = m + 1)$ 를 사용하는 단계;

상기 스레드 그룹 i 를 이용하여 상기 배열 $A1_i$ 와 $A2_i$ 에 $T[i..i + m - 1]$ 에 대한 각각의 짝 $(T[j], j)$, $(j, T[j]) (i \leq j \leq i + m - 1)$ 을 삽입하는 단계;

상기 배열 $A2_i$ 와 상기 $\pi^{-1}(P)$ 에 대해 상기 사각망 부분서열 $\hat{T}$ 이 상기 패턴 P 와 순위동형인지 확인하는 단계; 및

스텝 $j(i+m \leq j \leq n)$ 에서 j 를 증가시키면서 크기를 m 으로 유지하고 $T[i]$ 를 포함하는 상기 사각망 부분서열 $\hat{T}$ 을 탐색하는 단계를 포함하는 사각망 순열패턴매칭을 위한 방법.

청구항 8

제7항에 있어서,

상기 각각의 짝 $(T[j], j)$, $(j, T[j])$ ($i \leq j \leq i+m-1$)을 삽입하는 단계는,

상기 배열 $A1_i$ 의 첫 번째 원소에 대해 $A1_i[t].first > T[j]$ 인 스레드 t 는 임시 배열 $tmp1_i[t+1]$ 에 $A1_i[t]$ 을 저장하고, $A1_i[t].first > T[j]$ 인 스레드 중에서 $A1_i[t-1].first < T[j]$ 인 스레드 t 가 존재하면 $A1_i[t]$ 에 $(T[j], j)$ 를 저장하고, $A1_i[t].first > T[j]$ 인 스레드 중에서 $A1_i[t-1].first < T[j]$ 인 스레드 t 가 존재하지 않으면 $A1_i[m+1]$ 에 $(T[j], j)$ 를 저장하는 단계

를 포함하는 사각망 순열패턴매칭을 위한 방법.

청구항 9

제7항에 있어서,

상기 탐색단계는,

상기 사각망 부분서열 $\hat{T}$ 의 $T[i]$ 의 순위를 $P[1]$ 의 순위와 같도록 상기 사각망 부분서열 $\hat{T}$ 를 보정하는 단계

를 더 포함하는 사각망 순열패턴매칭을 위한 방법.

청구항 10

제9항에 있어서,

상기 보정하는 단계는,

상기 사각망 부분서열 $\hat{T}$ 에서 $T[i]$ 의 순위가 $P[1]$ 의 순위보다 높으면 상기 배열 $A1_i$ 에서 순위가 가장 낮은 원소 $T[k]$ 를 포함하는 짝 $(T[k], k)$ 와 $(k, T[k])$ 를 상기 배열 $A1_i$ 와 $A2_i$ 에서 각각 삭제하고, $T[i]$ 의 순위가 $P[1]$ 의 순위보다 높지 않으면 상기 배열 $A1_i$ 에서 순위가 가장 높은 원소 $T[k']$ 를 포함하는 짝 $(T[k'], k')$ 와 $(k', T[k'])$ 를 상기 배열 $A1_i$ 와 $A2_i$ 에서 각각 삭제하는 단계

를 포함하는 사각망 순열패턴매칭을 위한 방법.

발명의 설명

기술 분야

[0001] 아래의 설명은 사각망 순열패턴매칭을 위한 알고리즘에 관한 것이다.

배경 기술

[0002] 사각망 순열패턴매칭(boxed-mesh permutation pattern matching)은 순열패턴매칭(permutation pattern matching)의 한 분야로, 텍스트 $T(|T|=n)$ 와 패턴 $P(|P|=m)$ 가 주어졌을 때 P와 순위동형인 T의 모든 사각망 부분서열(boxed subsequence)을 찾는 문제이다. 순위동형(order-isomorphism)이란 대소 비교가 가능한 문자들로 구성된 두 문자열의 상대적인 순위가 같음을 말한다.

[0003] 사각망 순열(boxed-mesh permutation)은 순열패턴매칭에서 처음으로 정의되었고 패턴회피(pattern avoidance) 측면에서 연구되었다. 사각망 순열패턴매칭 문제에 대해 $O(n^3)$ 시간에 해결하는 알고리즘이 먼저 제시되고 이후 $O(n^2m)$ 시간에 해결하는 알고리즘과 $O(n^2 \log m)$ 시간에 해결하는 알고리즘이 제시되었다.

[0004] 예컨대, 한국등록특허 제10-1626721호(등록일 2016년 05월 26일)에는 순위통계트리(order-statistics tree)를 이용하여 종래의 결과보다 개선된 시간에 사각망 순열패턴매칭 문제를 해결하는 알고리즘이 개시되어 있다.

[0005] 최근에는 $O(n^2)$ 시간과 $O(n)$ 의 공간을 이용하여 사각망 순열패턴매칭 문제를 해결하는 최적 알고리즘과, $O(n^2 \log m)$ 시간과 $O(m)$ 공간을 이용하여 사각망 순열패턴매칭 문제를 해결하는 알고리즘이 제시되었다.

[0006] 한편, GPU(graphic process unit)의 성능이 향상되면서 GPU를 범용 계산(general propose)에도 활용한 연구가 진행되고 있다. GPU 범용 계산의 일례로는 문서 유사도 계산을 병렬화하는 방법이다. 다른 예로는 편집거리 계산을 위한 공간 효율적인 4-러시안 알고리즘 기반 병렬 알고리즘이 알려져 있다. 상기한 병렬 알고리즘들이 순차 알고리즘보다 빠른 수행 시간을 보이고 있다.

[0007] 최근 순열패턴매칭에서는 연속된 순열패턴(consecutive permutation pattern)에 대한 문제를 SSE 4.2(streaming SIMD extensions)와 AVX(Advance vector extensions) 명령어를 이용하여 해결하는 병렬 알고리즘이 알려져 있다.

발명의 내용

해결하려는 과제

[0008] 사각망 순열패턴매칭 문제를 해결하기 위한 병렬 알고리즘을 제공한다.

[0009] 사각망 순열패턴매칭 문제에 대해 $O(n)$ 개의 스레드를 이용하여 $O(nm)$ 시간에 해결하는 병렬 알고리즘을 제공한다.

[0010] 사각망 순열패턴매칭 문제에 대해 $O(nm)$ 개의 스레드를 이용하여 $O(n)$ 시간에 해결하는 병렬 알고리즘을 제공한다.

과제의 해결 수단

[0011] 상수 시간에 비교 가능한 문자 집합을 나타내는 Σ 상의 텍스트 $T(|T|=n)$ 와 패턴 $P(|P|=m)$ 가 주어짐

에 따라, 길이가 m 인 배열 tmp 을 정렬하고 정렬된 배열 tmp 을 이용하여 $\pi(P)$ 와 $\pi^{-1}(P)$ 을 계산하는 전처리단계; 및 상기 텍스트 T 에 대하여 단계 $i(1 \leq i \leq n-m+1)$ 의 각 스텝 $j(i \leq j \leq n)$ 에서 j 를 증가시키면서 상기 패턴 P 와 순위동형인 사각망 부분서열 $\hat{T}$ 을 찾는 탐색단계를 포함하는 사각망 순열패턴매칭을 위한 방법을 제공한다.

[0012] 일 측면에 따르면, 상기 전처리단계는, m 개의 스레드를 이용하여 의사코드 전처리(Preprocessing)을 수행하는 것으로, 상기 배열 tmp 에 짝 $(P[i], i)$ 형태로 저장하고 상기 배열 tmp 을 상기 짝의 첫 번째 원소를 기준으로 병렬 바이토닉 정렬(bitonic sort)을 이용하여 정렬한 후 정렬된 배열 tmp 을 이용하여 상기 $\pi(P)$ 와 상기 $\pi^{-1}(P)$ 을 계산할 수 있다.

[0013] 다른 측면에 따르면, 상기 탐색단계는, 서로 독립적인 각 단계 i 마다 스레드 i 를 이용하여 스텝 j 를 계산할 수 있다.

[0014] 또 다른 측면에 따르면, 상기 탐색단계는, 상기 사각망 부분서열 $\hat{T}$ 을 관리하기 위해 짝 $(T[j], j)$ 을 저장하는 배열 $A1_i(|A1_i|=m+1)$ 와 짝 $(j, T[j])$ 를 저장하는 배열 $A2_i(|A2_i|=m+1)$ 를 사용하는 단계; 스텝 j 에서 상기 배열 $A1_i$, $A2_i$ 에 각각 $(T[j], j)$, $(j, T[j])$ 를 삽입하는 단계; 상기 사각망 부분서열 $\hat{T}$ 의 $T[i]$ 의 순위를 $P[1]$ 의 순위와 같도록 상기 사각망 부분서열 $\hat{T}$ 를 보정하는 단계; 및 상기 배열 $A2_i$ 와 상기 $\pi^{-1}(P)$ 에 대해 상기 사각망 부분서열 $\hat{T}$ 이 상기 패턴 P 와 순위동형인지 확인하는 단계를 포함할 수 있다.

[0015] 또 다른 측면에 따르면, 상기 보정하는 단계는, 상기 사각망 부분서열 $\hat{T}$ 에서 $T[i]$ 의 순위가 $P[1]$ 의 순위보다 높으면 상기 배열 $A1_i$ 에서 순위가 가장 낮은 원소 $T[k]$ 를 포함하는 짝 $(T[k], k)$ 와 $(k, T[k])$ 를 상기 배열 $A1_i$ 와 $A2_i$ 에서 각각 삭제하고, $T[i]$ 의 순위가 $P[1]$ 의 순위보다 높지 않으면 상기 배열 $A1_i$ 에서 순위가 가장 높은 원소 $T[k']$ 를 포함하는 짝 $(T[k'], k')$ 와 $(k', T[k'])$ 를 상기 배열 $A1_i$ 와 $A2_i$ 에서 각각 삭제하는 단계를 포함할 수 있다.

[0016] 또 다른 측면에 따르면, 상기 탐색단계는, $O(n)$ 개의 스레드 그룹 $i(1 \leq i \leq n-m+1)$ 를 이용하여 단계 i 를 계산할 수 있다.

[0017] 또 다른 측면에 따르면, 상기 탐색단계는, 상기 사각망 부분서열 $\hat{T}$ 을 관리하기 위해 짝 $(T[j], j)$ 을 저장하는 배열 $A1_i(|A1_i|=m+1)$ 와 짝 $(j, T[j])$ 를 저장하는 배열 $A2_i(|A2_i|=m+1)$ 를 사용하는 단계; 상기 스레드 그룹 i 를 이용하여 상기 배열 $A1_i$ 와

$A2_i$ 에 $T[i..i+m-1]$ 에 대한 각각의 짝 $(T[j], j)$, $(j, T[j])$ ($i \leq j \leq i+m-1$)을 삽입하는 단계; 상기 배열 $A2_i$ 와 상기 $\pi^{-1}(P)$ 에 대해 상기 사각망 부분서열 $\hat{T}$ 이 상기 패턴 P 와 순위동형인지 확인하는 단계; 및 스텝 $j(i+m \leq j \leq n)$ 에서 j 를 증가시키면서 크기를 m 으로 유지하고 $T[i]$ 를 포함하는 상기 사각망 부분서열 $\hat{T}$ 을 탐색하는 단계를 포함할 수 있다.

[0018]

또 다른 측면에 따르면, 상기 각각의 짝 $(T[j], j)$, $(j, T[j])$ ($i \leq j \leq i+m-1$)을 삽입하는 단계는, 상기 배열 $A1_i$ 의 첫 번째 원소에 대해 $A1_i[t].first > T[j]$ 인 스레드 t 는 임시 배열 $tmp1_i[t+1]$ 에 $A1_i[t]$ 을 저장하고, $A1_i[t].first > T[j]$ 인 스레드 중에서 $A1_i[t-1].first < T[j]$ 인 스레드 t 가 존재하면 $A1_i[t]$ 에 $(T[j], j)$ 를 저장하고, $A1_i[t].first > T[j]$ 인 스레드 중에서 $A1_i[t-1].first < T[j]$ 인 스레드 t 가 존재하지 않으면 $A1_i[m+1]$ 에 $(T[j], j)$ 를 저장하는 단계를 포함할 수 있다.

[0019]

또 다른 측면에 따르면, 상기 탐색단계는, 상기 사각망 부분서열 $\hat{T}$ 의 $T[i]$ 의 순위를 $P[1]$ 의 순위와 같도록 상기 사각망 부분서열 $\hat{T}$ 를 보정하는 단계를 더 포함할 수 있다.

[0020]

또 다른 측면에 따르면, 상기 보정하는 단계는, 상기 사각망 부분서열 $\hat{T}$ 에서 $T[i]$ 의 순위가 $P[1]$ 의 순위보다 높으면 상기 배열 $A1_i$ 에서 순위가 가장 낮은 원소 $T[k]$ 를 포함하는 짝 $(T[k], k)$ 와 $(k, T[k])$ 를 상기 배열 $A1_i$ 와 $A2_i$ 에서 각각 삭제하고, $T[i]$ 의 순위가 $P[1]$ 의 순위보다 높지 않으면 상기 배열 $A1_i$ 에서 순위가 가장 높은 원소 $T[k']$ 를 포함하는 짝 $(T[k'], k')$ 와 $(k', T[k'])$ 를 상기 배열 $A1_i$ 와 $A2_i$ 에서 각각 삭제하는 단계를 포함할 수 있다.

발명의 효과

[0021]

본 발명의 실시예들에 따르면, $O(n)$ 개의 스레드를 이용하여 $O(nm)$ 시간에 해결하는 병렬 알고리즘을 통해 사각망 순열패턴매칭 문제를 해결할 수 있다.

[0022]

본 발명의 실시예들에 따르면, $O(nm)$ 개의 스레드를 이용하여 $O(n)$ 시간에 해결하는 병렬 알고리즘을 통해 사각망 순열패턴매칭 문제를 해결할 수 있다.

도면의 간단한 설명

[0023]

도 1은 본 발명의 일실시예에 있어서 사각망 순열패턴을 설명하기 위한 예시 도면이다.

도 2는 본 발명의 일실시예에 있어서 사각망 순열패턴매칭 문제를 해결하기 위한 병렬 알고리즘의 전체리단계를

도시한 것이다.

도 3은 본 발명의 일실시예에 있어서 사각망 순열패턴매칭 문제를 해결하기 위한 병렬 알고리즘의 탐색단계의 일례를 도시한 것이다.

도 4는 본 발명의 일실시예에 있어서 사각망 순열패턴매칭 문제를 해결하기 위한 병렬 알고리즘의 탐색단계의 다른 예를 도시한 것이다.

도 5는 본 발명의 일실시예에 있어서 탐색단계에서의 삽입 연산을 설명하기 위한 예시 도면이다.

도 6은 본 발명의 일실시예에 있어서 탐색단계에서의(또는 탐색단계의 보정하는 단계에서의) 삭제 연산을 설명하기 위한 예시 도면이다.

발명을 실시하기 위한 구체적인 내용

[0024] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0026] 사각망 순열패턴매칭은 텍스트 $T(|T|=n)$ 와 패턴 $P(|P|=m)$ 가 주어졌을 때, P와 순위동형인 T의 모든 사각망 부분서열을 찾는 문제이다. 본 실시예들은 사각망 순열패턴매칭 문제를 해결하는 두 가지 병렬 알고리즘을 제공한다. 즉, 이 문제에 대해 $O(n)$ 개의 스레드를 사용하여 $O(nm)$ 시간에 해결하는 병렬 알고리즘과 $O(nm)$ 개의 스레드를 사용하여 $O(n)$ 시간에 해결하는 병렬 알고리즘을 제시한다.

[0027] 먼저, 관련 용어와 문제를 정의한다.

[0028] Σ 는 상수시간에 대소 비교가 가능한 문자집합을 나타낸다. Σ 로 구성된 문자열들의 집합을 Σ^* 로 표기한다. 문자열 $x(\in \Sigma^*)$ 의 길이를 $|x|$ 로, x 의 i 번째 문자를 $x[i](1 \leq i \leq |x|)$ 로 표기한다. 문자열 x 안에 포함된 모든 문자들은 서로 다르다고 가정한다. 문자열 x 의 i 번째부터 j 번째까지의 부분문자열을 $x[i..j](1 \leq i \leq j \leq |x|)$ 로 표기한다. $x[1..i](1 \leq i \leq |x|)$ 를 x 의 i 번째 접두사(prefix)라 하고, $x[i..|x|](1 \leq i \leq |x|)$ 를 x 의 i 번째 접미사(suffix)라 한다.

[0029] 문자열 $x(\in \Sigma^*)$ 가 주어졌을 때, x 의 각 문자의 순위(rank)를 저장한 서열을 순열(permutation) $\pi(x)$, x 의 순위별 위치를 저장한 서열을 $\pi^{-1}(x)$ 라 하자. 즉, $\pi(x)[i]=|\{k:x[k] \leq x[i], 1 \leq k \leq |x|\}|$ 로, $\pi(x)[i]=j$ 일 때 $\pi^{-1}(x)[j]=i$ 로 정의된다. 예를 들어, $x=(7,1,4,5,3,8)$ 일 때, $\pi(x)=(5,1,3,4,2,6)$, $\pi^{-1}(x)=(2,5,3,4,1,6)$ 이다. $\pi(x)$ 는 x 의 문자들을 정렬하거나 순위통계트리(order-statistics tree)를 이용하여 $O(|x|\log|x|)$ 시간에 계산될 수 있으며, $\pi^{-1}(x)$ 는 $\pi(x)$ 를 이용하여 $O(|x|)$ 시간에 계산될 수 있다.

[0030] 길이가 같은 두 문자열 x , $y(\in \Sigma^*)$ 가 주어졌을 때, $x[i] \leq x[j] \Leftrightarrow y[i] \leq y[j] (1 \leq i, j \leq |x|)$ 를 만족하면 x 와 y 는 순위동형이라고, $x \approx y$ 로 표기한다. 이는 $\pi^{-1}(x)$ 가 주어졌을 때, 다음의 수학적 1을 만족하는지 확인함으로써 $O(|x|)$ 시간에 판단할 수 있다.

[0031] [수학적 1]

$$[0032] y[\pi^{-1}(x)[i-1]] < y[\pi^{-1}(x)[i]] (1 < i \leq |x|)$$

[0033] 순열패턴에서는 주어진 문자열을 좌표평면에 점으로 표시할 수 있다. 문자열 $x(\in \Sigma^*)$ 의 i 번째 문자인 $x[i]$ 의 순위 $\pi(x)[i]$ 를 c 라 할 때, 2차원 평면에서 점 (i, c) 로 표현될 수 있다. 문자열 x 를 나타낸 좌표평면 상에서 축에 평행한 직사각형 영역을 $R = (left, right, bottom, top)$ 이라 하자. 이때, left, right, bottom, top은 각각 영역 R 의 왼쪽, 오른쪽, 아래, 위의 경계선을 나타낸다. 문자열 x 의 사각망 순열은 $B = (\pi(x), R)$ 로 정의된다. 이때 영역 R 은 문자열 x 에 대한 2차원 평면상에 나타낸 점들을 모두 포함하는 가장 작은 직사각형 영역이다. 그리고 사각망 순열의 크기는 $|B| = |x|$ 로, 영역 R 에 포함된 점의 수이다. 문자열 x 의 임의의 부분서열 x' 중에서 사각망 순열 $B = (\pi(x'), R)$ 이 정의될 때, x' 을 x 의 사각망 부분서열(boxed subsequence)이라 한다. 두 사각망 순열 $B = (\pi(x), R)$ 와 $B' = (\pi(y), R')$ 이 주어졌을 때, $\pi(x) = \pi(y)$ 이면, 두 사각망 순열은 일치한다고 한다. 즉, 문자열 x 와 y 가 서로 순위동형이면 두 사각망 순열은 일치한다.

[0034] 사각망 순열패턴매칭 문제는 텍스트 $T(\in \Sigma^*)$ 와 패턴 $P(\in \Sigma^*)$ 가 주어졌을 때, 패턴 P 와 순위동형인 텍스트 T 의 사각망 부분서열을 모두 찾는 것이다. 예를 들어, 도 1에서 $P = (3, 1, 4, 2)$ 에 대한 사각망 순열은 $((3, 1, 4, 2), (1, |P|, 1, |P|))$ 이다. T 를 표현한 좌표평면에서 영역 $(1, 6, 1, 4)$ 와 $(5, 8, 2, 9)$ 에 포함된 점들의 순열은 $(3, 1, 4, 2)$ 로 $\pi(P)$ 와 일치한다. 도 1에서 회색으로 표시된 영역은 P 의 사각망 순열과 일치하는 T 의 영역인 $(1, 6, 1, 4)$, $(5, 8, 2, 9)$ 를 나타낸다.

[0035] 기존 알고리즘은 전처리단계와 탐색단계로 구성된다. 전처리단계에서는 수학적 1로 P 와 T 의 사각망 부분서열이 순위동형인지 판단할 수 있도록, 순위통계트리를 이용하여 $\pi(P)$ 와 $\pi^{-1}(P)$ 를 계산한다. 사각망 순열의 영역 R 의 left의 인덱스를 i , right의 인덱스를 $j(i \leq j)$ 라 하자. 영역 R 의 좌우 양 끝인 $i, j(i \leq j)$ 로 결정되었을 때, R 에 의해서 결정되는 T 의 사각망 부분서열 $\hat{T}$ 은 P 와 많아야 1개가 순위동형이기 때문에, 탐색단계에서는 T 의 모든 i, j 쌍의 사각망 부분서열에 대해 P 와 순위동형인지 계산한다. 즉,

단계 $j (m \leq j \leq n)$ 에서 j 를 증가시키면서 $T[1..j]$ 에서 $T[j]$ 를 가장 오른쪽 원소로 포함하고 크기가 m 이 넘지 않는 모든 사각망 부분서열 $\hat{T}$ 에 대해 처리한다. 이를 위해 단계 j 의 각 스텝 $i (1 \leq i \leq j)$ 에서 i 를 감소시키면서 가장 왼쪽 원소로 $T[i]$ 와 가장 오른쪽 원소로 $T[j]$ 를 포함하고 P 와 순위동형인 사각망 부분서열 $\hat{T}$ 을 탐색한다. 이때 사각망 부분서열 $\hat{T}$ 을 위한 2개의 순위통계트리 OST1과 OST2를 이용한다. OST1과 OST2는 각각 $T[i]$ 와 i 를 원소로 하는 짝(pair) $(T[i], i), (i, T[i])$ 형태로 저장하고 각각 짝의 첫 번째 원소를 기준으로 정렬된다. 세부적으로, 단계 j 의 각 스텝 $i (1 \leq i \leq j)$ 에서 먼저 OST1과 OST2에 $(T[i], i)$ 와 $(i, T[i])$ 를 각각 삽입한다. 그 다음 사각망 부분서열 $\hat{T}$ 의 $T[j]$ 의 순위를 $P[m]$ 의 순위와 같도록 사각망 부분서열 $\hat{T}$ 을 보정한다. 즉, OST1에서 $(T[j], j)$ 를 탐색하여 $T[j]$ 의 순위가 $P[m]$ 의 순위보다 높으면, OST1과 OST2에서 각각 순위가 가장 낮은 원소 $T[k]$ 를 포함하는 짝 $(T[k], k)$ 와 $(k, T[k])$ 를 각각 제거하고, $T[j]$ 의 순위가 $P[m]$ 의 순위보다 높지 않고 원소의 개수가 m 보다 크면 OST1과 OST2에서 각각 순위가 가장 높은 원소 $T[k']$ 를 포함하는 짝 $(T[k'], k')$ 와 $(k', T[k'])$ 를 각각 제거한다. OST1의 크기가 m 이고 $T[i]$ 를 원소로 가지고 있을 때, OST2를 중위순회(inorder traversal)하여 위치 순으로 정렬된 $\hat{T}$ 을 얻고, $\hat{T}$ 과 $\pi^{-1}(P)$ 에 대해 수학적 1을 이용하여 $\hat{T}$ 과 패턴 P 가 순위동형인지 확인한다. 만약 $\hat{T} \approx P$ 이면 $\hat{T}$ 의 영역 R 을 출력한다.

[0036] 상기한 알고리즘의 시간복잡도는 다음과 같다. 전처리단계에서는 순위통계트리를 이용하여 $\pi(P)$ 와 $\pi^{-1}(P)$ 를 $O(m \log m)$ 시간에 계산할 수 있다. 탐색단계에서는 사각망 부분서열 $\hat{T}$ 을 위한 순위통계트리를 유지하기 때문에, 각 스텝마다 한 원소에 대한 삽입, 삭제, 순위를 찾는 연산은 $O(\log m)$ 시간에 수행된다. $\hat{T}$ 과 P 가 순위동형인지는 $O(m)$ 시간에 판단할 수 있다. 한편, 최악의 경우 $O(n^2)$ 개의 사각망 부분서열 $\hat{T}$ 과 패턴 P 가 순위동형인지를 판단해야 하기 때문에 총 시간복잡도는 $O(n^2 m)$ 이다.

[0037] 이하에서는 사각망 순열패턴매칭 문제를 해결하기 위한 병렬 알고리즘을 구체적으로 설명한다.

[0038] 본 발명에서는 상기한 알고리즘에서 제시된 아이디어를 바탕으로 사각망 순열패턴매칭 문제를 해결하는 두 가지 병렬 알고리즘을 제시한다. 기존 알고리즘의 탐색단계에서는 각 $T[1..j] (m \leq j \leq n)$ 에 대해 마지막 문자인 $T[j]$ 를 고정시키고 역방향으로 탐색을 진행하였다. 한편, $T[i..n] (1 \leq i \leq n - m + 1)$ 에 대해 시작 문자인 $T[i]$ 를 고정시키고 순방향으로 탐색해도 사각망 순열패턴매칭 문제를 해결할 수 있다.

- [0039] 본 발명에 따른 병렬 알고리즘에서는 순방향으로 탐색한다. 단계 i 의 각 스텝 $j(i \leq j \leq n)$ 에서 j 를 증가시키면서 패턴 P 와 순위동형이고 가장 왼쪽 원소로 $T[i]$, 가장 오른쪽 원소로 $T[j]$ 를 포함하는 사각망 부분서열 $\hat{T}$ 을 찾는다. 그리고 사각망 부분서열 $\hat{T}$ 을 관리하기 위해 순위통계트리 대신 배열과 정렬알고리즘을 사용한다.
- [0040] 병렬 알고리즘 1
- [0041] 스레드 수가 $O(n)$ 일 때, 사각망 순열패턴매칭 문제를 $O(nm)$ 시간에 해결하는 병렬 알고리즘을 제시한다. 전처리단계에서는 m 개의 스레드를 이용하여 의사코드 전처리(Preprocessing)을 수행한다(도 2 참조). 먼저 길이가 m 인 배열 tmp 에 짝 $(P[i], i)$ 형태로 저장한다. 이때 짝의 첫 번째 원소를 $tmp.first$, 두 번째 원소를 $tmp.second$ 로 표기한다. tmp 를 $tmp.first$ 를 기준으로 병렬 바이토닉 정렬(bitonic sort)를 이용하여 정렬하고, 정렬된 tmp 를 이용하여 $\pi(P)$ 와 $\pi^{-1}(P)$ 를 계산한다.
- [0042] 도 3을 참조하면, 탐색단계에서는 각 단계 $i(1 \leq i \leq n-m+1)$ 가 서로 독립적이므로, 단계 i 마다 스레드 i 를 이용하여 스텝 $j(i \leq j \leq n)$ 를 계산한다(2번째 줄부터 13번째 줄). 각 스레드 i 는 $T[i..n]$ 에서 나타나는 사각망 부분서열 $\hat{T}$ 을 관리하기 위해 $T[j]$ 와 인덱스 j 를 원소로 하는 짝 $(T[j], j)$ 을 저장하는 배열 $A1_i(|A1_i|=m+1)$ 와 짝 $(j, T[j])$ 를 저장하는 배열 $A2_i(|A2_i|=m+1)$ 를 사용한다(3번째 줄). 배열 $A1_i$ 와 $A2_i$ 는 각각 짝의 첫 번째 원소를 기준으로, 오름차순으로 정렬된 상태를 유지한다. 세부적으로, 스텝 j 에서 먼저 배열 $A1_i, A2_i$ 에 각각 $(T[j], j), (j, T[j])$ 를 삽입한다(5번째 줄). $A1_i$ 와 $A2_i$ 를 정렬된 상태로 유지하기 위해 삽입 연산을 삽입정렬(insertion sort)과 같은 방법으로 수행한다. 그 후 사각망 부분서열 $\hat{T}$ 의 $T[i]$ 의 순위를 $P[1]$ 의 순위와 같도록 사각망 부분서열 $\hat{T}$ 를 보정한다. 즉, 사각망 부분서열 $\hat{T}$ 에서 $T[i]$ 의 순위가 $P[1]$ 의 순위보다 높으면 $A1_i$ 에서 순위가 가장 낮은 원소 $T[k]$ 를 포함하는 짝 $(T[k], k)$ 와 $(k, T[k])$ 를 $A1_i, A2_i$ 에서 각각 제거하고, $T[i]$ 의 순위가 $P[1]$ 의 순위보다 높지 않고 원소의 개수가 m 보다 크면 $A1_i$ 에서 순위가 가장 높은 원소 $T[k']$ 를 포함하는 짝 $(T[k'], k')$ 와 $(k', T[k'])$ 를 $A1_i, A2_i$ 에서 각각 제거한다(6번째 줄부터 11번째 줄). 삭제 연산은 삭제될 짝 $(T[k], k)(1 \leq k \leq m)$ 에 대해 오른쪽에 있는 짝들을 차례대로 왼쪽으로 한 칸씩 이동시키면 된다. $|\hat{T}|=m$ 이고 $\hat{T}$ 의 원소로 $T[j]$ 를 가지고 있을 때, $\hat{T}$ 과 P 가 순위동형인지는 $A2_i$ 와 $\pi^{-1}(P)$ 에 대해 수학적 식 1을 이용하여 확인한다. 만약 $\hat{T} \approx P$ 이면 $\hat{T}$ 의 영역

$R = (i, j, A1_i[1].first, A1_i[m].first)$ 을 출력한다(12번째 줄부터 13번째 줄).

[0043] 상기한 병렬 알고리즘 1의 시간 복잡도는 다음과 같다. 전처리단계에서는 m 개의 스레드와 병렬 바이토닉 정렬을 사용하여 $O(\log^2 m)$ 시간에 길이가 m 인 배열을 정렬할 수 있고, 이 배열을 이용하여 $\pi(P)$ 와 $\pi^{-1}(P)$ 를 $O(1)$ 시간에 계산할 수 있다. 탐색단계에서 각 스텝은 배열 $A1_i$ 와 $A2_i$ 에 하나의 원소에 대한 삽입, 삭제, 순위를 찾는 연산, 한 원소가 존재하는지 확인하는 연산과 $\hat{T}$ 과 P 가 순위동형인지 확인하는 연산을 수행한다. 삭제 연산은 $A1_i$ 에서 가장 작은 원소 또는 가장 큰 원소를 포함하는 짝만 삭제한다. 이때 배열 $A1_i$ 가 항상 정렬되어 있기 때문에, 이를 각각 $O(1)$ 시간에 탐색할 수 있다(7번째 줄과 10번째 줄). 삽입 연산과 삭제 연산은 $O(m)$ 시간에 수행될 수 있다. 그리고 $T[i]$ 의 순위를 찾는 연산(6번째 줄)과 $T[j]$ 의 존재여부를 판단하는 연산(12번째 줄)은 $A1_i$ 에서 이진탐색(binary search)를 이용하여 $O(\log m)$ 시간에 계산할 수 있다. 마지막으로 $\hat{T}$ 과 P 가 순위동형인지는 $A2_i$ 와 $\pi^{-1}(P)$ 에 대해 수학적 1을 이용하므로 $O(m)$ 시간이 소요된다. 각 스레드가 최대 $O(n)$ 개의 스텝을 탐색하고 각 스텝은 최악의 경우 $O(m)$ 시간이 소요되므로 병렬 알고리즘 1의 시간복잡도는 $O(nm)$ 이다.

[0044] 병렬 알고리즘 2

[0045] 스레드의 수가 $O(nm)$ 일 때, 사각망 순열패턴매칭 문제를 $O(n)$ 시간에 해결하는 병렬 알고리즘을 제시한다. m 개의 스레드를 스레드 그룹(thread group)이라 하자. 병렬 알고리즘 2는 $O(n)$ 개의 스레드 그룹을 사용하여 스레드 그룹 $i (1 \leq i \leq n - m + 1)$ 가 단계 i 를 계산한다.

[0046] 전처리단계는 병렬 알고리즘 1과 동일하게 수행된다(1번째 줄). 도 4를 참조하면, 탐색단계에서는 사각망 부분서열 $\hat{T}$ 을 위한 길이가 $m+1$ 인 배열 $A1_i$ 와 $A2_i$ 를 사용한다. 각 단계 $i (1 \leq i \leq n - m + 1)$ 에서 먼저 스레드 그룹 i 를 이용하여 $A1_i$ 와 $A2_i$ 에 $T[i..i+m-1]$ 에 대한 각각의 짝 $(T[j], j)$, $(j, T[j]) (i \leq j \leq i+m-1)$ 을 m 개의 스레드를 이용하여 삽입한다. 그리고 패턴 P 와 사각망 부분서열 $\hat{T}$ 이 순위동형인지 $A2_i$ 와 $\pi^{-1}(P)$ 에 대해 수학적 1을 이용하여 확인한다(3번째 줄부터 5번째 줄). 이후 스텝 $j (i+m \leq j \leq n)$ 에서 j 를 증가시키면서 크기를 m 으로 유지하고 $T[i]$ 를 포함하는 사각망 부분서열 $\hat{T}$ 을 탐색한다. 세부적으로, $(T[j], j)$ 와 $(j, T[j])$ 를 각각 배열 $A1_i$ 와 $A2_i$ 에 삽입한다(7번째 줄). 삽입 연산은 하나의 스레드 그룹을 이용하여 다음과 같이 수행된다. 정렬된 배열 $A1_i$ 는 $T[j]$ 보다 작은 원소들과 큰 원소들로 나눌 수 있다(도 5 참조). 도 5에서 $A1_i[t].first (1 \leq t \leq m)$ 가 $T[j]$ 보다 작은 원소들은 수직선 해칭으로, 큰 원소들을 수평선 해

칭으로 표시한다. $A1_i[t].first > T[j]$ 인 각 스레드 t 는 임시 배열 $tmp1_i[t+1]$ 에 $A1_i[t]$ 을 저장하고 $A1_i[t].first > T[j]$ 인 스레드 중에서 $A1_i[t-1].first < T[j]$ 인 스레드 t 가 존재하면 $A1_i[t]$ 에 $(T[j], j)$ 를 저장한다. 만약 그러한 스레드 t 가 존재하지 않으면, $A1_i[m+1]$ 에 $(T[j], j)$ 를 저장한다. 그 후 $tmp1_i[t+1].first > T[j]$ 인 각 스레드 t 는 $A1_i[t+1]$ 에 $tmp_i[t+1]$ 을 저장한다. $A2_i$ 에 대해서도 유사한 방법으로 수행한다. 사각망 부분서열 $\hat{T}$ 이 패턴 P 와 순위동형이 되려면 $T[i]$ 와 $P[1]$ 의 순위가 같고 길이가 같아야 한다. 따라서 $T[i]$ 의 순위가 $P[1]$ 의 순위보다 높으면 배열 $A1_i$ 에서 순위가 가장 낮은 원소 $T[k]$ 를 포함하는 짝 $(T[k], k)$ 와 $(k, T[k])$ 를 각각 $A1_i$, $A2_i$ 에서 삭제하고, 그렇지 않으면 배열 $A1_i$ 에서 순위가 가장 높은 원소 $T[k']$ 를 포함하는 짝 $(T[k'], k')$ 와 $(k', T[k'])$ 를 각각 $A1_i$, $A2_i$ 에서 삭제한다 (9번째 줄부터 11번째 줄). 도 6은 $A2_i$ 에서의 사선 해칭으로 표시한 하나의 원소를 삭제하는 연산을 보여준다. 이때 삭제연산은 하나의 스레드 그룹을 이용하여 각 스레드 $t (1 \leq t \leq m)$ 가 삭제할 원소보다 큰 원소를 가진 t 에 대해서 $A1_i[t+1]$ 를 $A1_i[t]$ 로 옮기면 된다. $A1_i$ 에 대해서도 유사한 방법으로 수행한다. 그리고 $T[i]$ 의 순위를 찾는 연산은 각 스레드 t 가 배열 $A1_i[t]$ 의 원소와 $(T[i], i)$ 가 일치하는 인덱스 t 를 반환하면 된다. 이후에 $T[j]$ 가 사각망 부분서열 $\hat{T}$ 에 존재하는지만 확인한다. 만약 앞의 조건이 참이면 m 개의 스레드를 이용하여 $A2_i$ 와 $\pi^{-1}(P)$ 에 대해 수학식 1을 병렬적으로 확인하고, $\hat{T} \approx P$ 이면 $\hat{T}$ 의 영역 $R = (i, j, A1_i[1].first, A1_i[m].first)$ 를 출력한다(13번째 줄부터 14번째 줄).

[0047] 상기한 병렬 알고리즘 2의 시간 복잡도를 분석하면 다음과 같다. 전처리단계는 병렬 알고리즘 1과 같다. 탐색 단계에서는 처음 $T[i..i+m-1]$ 에 대한 m 개의 짝을 배열 $A1_i$ 와 $A2_i$ 에 삽입하는 것은 m 개의 스레드와 바이토닉 정렬을 사용하여 $O(\log^2 m)$ 시간에 계산할 수 있다. 이후 스택 j 에서 하나의 원소에 대한 삽입, 삭제, 순위를 찾는 연산은 m 개의 스레드를 이용하여 $O(1)$ 시간에 처리할 수 있다. 또한 $\hat{T}$ 과 P 의 순위동형 판단은 $A2_i$, $\pi^{-1}(P)$ 에 대한 수학식 1과 m 개의 스레드를 이용하여 $O(1)$ 시간에 계산할 수 있다. 따라서 하나의 스택을 $O(1)$ 시간에 처리할 수 있다. 각 단계는 최대 $O(n)$ 개의 스택이 존재하므로, 병렬 알고리즘 2의 시간복잡도는 $O(n)$ 이다.

[0048] 이처럼 본 발명의 실시예들에 따르면, $O(n)$ 개의 스레드를 이용하여 $O(nm)$ 시간에 사각망 순열패턴매칭 문제를 해결할 수 있다. 또한, 본 발명의 실시예들에 따르면, $O(nm)$ 개의 스레드를 이용하여 $O(n)$ 시간

에 사각망 순열패턴매칭 문제를 해결할 수 있다.

- [0049] 본 발명에 따른 사각망 순열패턴매칭 문제를 해결하기 위한 병렬 알고리즘은 도 1 내지 도 6을 통해 설명한 상세 내용을 바탕으로 둘 이상의 동작들을 포함할 수 있다. 상기한 알고리즘은 컴퓨터 시스템을 통해 실행될 수 있으며, 컴퓨터 시스템은 사각망 순열패턴매칭 문제를 해결하기 위한 병렬 알고리즘을 실행하기 위한 구성요소로서, 일례로 프로세서, 메모리, 영구 저장 장치, 버스, 입출력 인터페이스 및 네트워크 인터페이스를 포함할 수 있다.
- [0050] 프로세서는 명령어들의 시퀀스를 처리할 수 있는 임의의 장치를 포함하거나 그의 일부일 수 있다. 프로세서는 예를 들어 컴퓨터 프로세서, 이동 장치 또는 다른 전자 장치 내의 프로세서 및/또는 디지털 프로세서를 포함할 수 있다. 프로세서는 예를 들어, 서버 컴퓨팅 디바이스, 서버 컴퓨터, 일련의 서버 컴퓨터들, 서버 팜, 클라우드 컴퓨터, 콘텐츠 플랫폼, 이동 컴퓨팅 장치, 스마트폰, 태블릿, 셋톱 박스 등에 포함될 수 있다. 프로세서는 버스를 통해 메모리에 접속될 수 있다.
- [0051] 메모리는 컴퓨터 시스템에 의해 사용되거나 그에 의해 출력되는 정보를 저장하기 위한 휘발성 메모리, 영구, 가상 또는 기타 메모리를 포함할 수 있다. 예를 들어, 메모리는 랜덤 액세스 메모리(RAM: random access memory) 및/또는 동적 RAM(DRAM: dynamic RAM)을 포함할 수 있다. 메모리는 컴퓨터 시스템의 상태 정보와 같은 임의의 정보를 저장하는 데 사용될 수 있다. 메모리는 예를 들어 사각망 순열패턴매칭 문제를 해결하기 위한 명령어들을 포함하는 컴퓨터 시스템의 명령어들을 저장하는 데에도 사용될 수 있다. 컴퓨터 시스템은 필요에 따라 또는 적절한 경우에 하나 이상의 프로세서를 포함할 수 있다.
- [0052] 버스는 컴퓨터 시스템의 다양한 컴포넌트들 사이의 상호작용을 가능하게 하는 통신 기반 구조를 포함할 수 있다. 버스는 컴퓨터 시스템의 컴포넌트들 사이에, 예를 들어 프로세서와 메모리 사이에 데이터를 운반할 수 있다. 버스는 컴퓨터 시스템의 컴포넌트들 간의 무선 및/또는 유선 통신 매체를 포함할 수 있으며, 병렬, 직렬 또는 다른 토폴로지 배열들을 포함할 수 있다.
- [0053] 영구 저장 장치는 소정의 연장된 기간 동안 데이터를 저장하기 위해 컴퓨터 시스템에 의해 사용되는 바와 같은 메모리 또는 다른 영구 저장 장치와 같은 컴포넌트들을 포함할 수 있다. 영구 저장 장치는 컴퓨터 시스템 내의 프로세서에 의해 사용되는 바와 같은 비휘발성 메인 메모리를 포함할 수 있다. 예를 들어, 영구 저장 장치는 플래시 메모리, 하드 디스크, 광 디스크 또는 다른 컴퓨터 판독 가능 매체를 포함할 수 있다.
- [0054] 입출력 인터페이스는 키보드, 마우스, 마이크, 카메라, 디스플레이 또는 다른 입력 또는 출력 장치에 대한 인터페이스들을 포함할 수 있다. 구성 명령들 및/또는 사각망 순열패턴매칭 문제를 해결하기 위한 알고리즘의 입력이 입출력 인터페이스를 통해 수신될 수 있다.
- [0055] 네트워크 인터페이스는 근거리 네트워크 또는 인터넷과 같은 네트워크들에 대한 하나 이상의 인터페이스를 포함할 수 있다. 네트워크 인터페이스는 유선 또는 무선 접속들에 대한 인터페이스들을 포함할 수 있다. 구성 명령들은 네트워크 인터페이스를 통해 수신될 수 있다. 그리고, 사각망 순열패턴매칭 문제를 해결하기 위한 알고리즘의 정보들은 네트워크 인터페이스를 통해 수신 또는 송신될 수 있다.
- [0056] 또한, 다른 실시예들에서 컴퓨터 시스템은 상기한 구성요소들보다 더 많은 구성요소들을 포함할 수도 있다. 그러나, 대부분의 종래기술적 구성요소들을 명확하게 도시할 필요성은 없다. 예를 들어, 컴퓨터 시스템은 상술한 입출력 인터페이스와 연결되는 입출력 장치들 중 적어도 일부를 포함하도록 구현되거나 또는 트랜시버(transceiver), GPS(Global Positioning System) 모듈, 카메라, 각종 센서, 데이터베이스 등과 같은 다른 구성요소들을 더 포함할 수도 있다.
- [0057] 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, 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 어플리케이션을 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 장치

는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 컨트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0058] 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, 컴퓨터 저장 매체 또는 장치에 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.

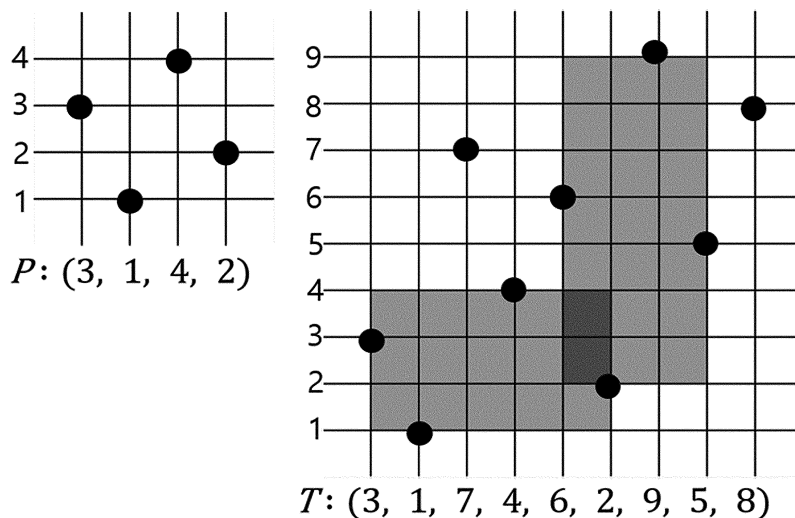
[0059] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 이때, 매체는 컴퓨터로 실행 가능한 프로그램을 계속 저장하거나, 실행 또는 다운로드를 위해 임시 저장하는 것일 수도 있다. 또한, 매체는 단일 또는 수 개의 하드웨어가 결합된 형태의 다양한 기록수단 또는 저장수단일 수 있는데, 어떤 컴퓨터 시스템에 직접 접속되는 매체에 한정되지 않고, 네트워크 상에 분산 존재하는 것일 수도 있다. 매체의 예시로는, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM 및 DVD와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical medium), 및 ROM, RAM, 플래시 메모리 등을 포함하여 프로그램 명령어가 저장되도록 구성된 것이 있을 수 있다. 또한, 다른 매체의 예시로, 어플리케이션을 유통하는 앱 스토어나 기타 다양한 소프트웨어를 공급 내지 유통하는 사이트, 서버 등에서 관리하는 기록매체 내지 저장매체도 들 수 있다.

[0060] 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.

[0061] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

도면1



```

Preprocessing
Input: String  $P$ 
Output:  $\pi(P), \pi^{-1}(P)$ 
1: Assign an array  $tmp$  of length  $m$ 
2: parallel for  $i = 1$  to  $m$ 
3:    $tmp[i] = (P[i], i)$ 
4: parallel_bitonic_sort( $tmp$ ) using  $m$  threads
5: parallel for  $i = 1$  to  $m$ 
6:    $\pi(P)[tmp[i].second] = i$ 
7:    $\pi^{-1}(P)[i] = tmp[i].second$ 

```

도면2

Algorithm 1

Input: Text $T(|T|=n)$, Pattern $P(|P|=m)$

Output: Boundary R of all boxed subsequence $\hat{T}$ of T that are order-isomorphic to P

```

1: Preprocessing( $P$ )
2: parallel for  $i = 1$  to  $n - m + 1$ 
3:   Assign two arrays  $A1_i$ ,  $A2_i$  of length  $m + 1$  for boxed subsequence  $\hat{T}$ 
4:   for  $j = i$  to  $n$ 
5:      $A1_i.insert((T[j], j))$ ,  $A2_i.insert((j, T[j]))$ 
6:     if  $A1_i.rank((T[i], i)) > \pi(P)[1]$ 
7:       then  $k = A1_i.min().second$ 
8:        $A1_i.delete((T[k], k))$ ,  $A2_i.delete((k, T[k]))$ 
9:     else if  $A1_i.size() > m$ 
10:      then  $k = A1_i.max().second$ 
11:       $A1_i.delete((T[k], k))$ ,  $A2_i.delete((k, T[k]))$ 
12:      if  $A1_i.find((T[j], j))$  and  $A1_i.size() == m$ 
13:        if  $\hat{T} \approx P$  then print boundary  $R$  of  $\hat{T}$ 

```

도면3

Algorithm 2

Input: Text $T(|T| = n)$, Pattern $P(|P| = m)$

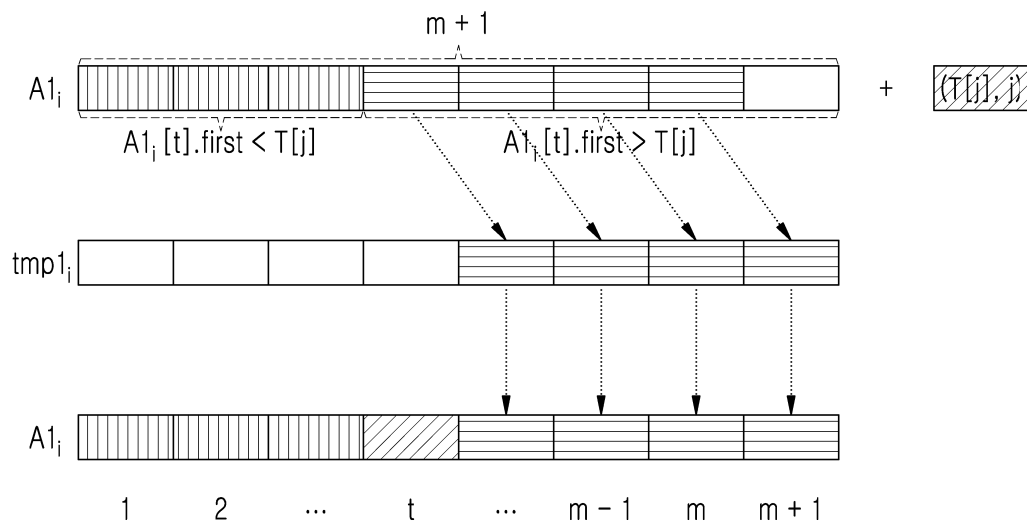
Output: Boundary R of all boxed subsequence $\hat{T}$ of T that are order-isomorphic to P

```

1: Preprocessing( $P$ )
2: parallel for  $i = 1$  to  $n - m + 1$  (using thread group  $i$ )    // Searching
3:   Assign two arrays  $A1_i, A2_i$  of length  $m + 1$  for boxed subsequence  $\hat{T}$ 
4:   Insert  $(T[j], j), (j, T[j])$  ( $i \leq j \leq i + m - 1$ ) to  $A1_i, A2_i$ 
      and sort  $A1_i, A2_i$  using parallel bitonic sort, respectively
5:   if  $\hat{T} \approx P$  then output boundary  $R$  of  $\hat{T}$ 
6:   for  $j = i + m$  to  $n$ 
7:      $A1_i.p\_insert((T[j], j)), A2_i.p\_insert((j, T[j]))$ 
8:     sync
9:     if  $A1_i.p\_rank((T[i], i)) > \pi(P)[1]$ 
10:       then  $k = A1_i.min().second, A1_i.p\_delete((T[k], k)), A2_i.p\_delete((k, T[k]))$ 
11:       else  $k' = A1_i.max().second, A1_i.p\_delete((T[k'], k')), A2_i.p\_delete((k', T[k']))$ 
12:       sync
13:       if  $A1_i.p\_find((T[j], j)))$ 
14:         if  $\hat{T} \approx P$  then print boundary  $R$  of  $\hat{T}$ 

```

도면5



도면6

