



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0071533
(43) 공개일자 2025년05월22일

- | | |
|---|--|
| <p>(51) 국제특허분류(Int. Cl.)
G06F 16/903 (2019.01) G06F 16/9032 (2019.01)
G16B 30/10 (2019.01)</p> <p>(52) CPC특허분류
G06F 16/90335 (2019.01)
G06F 16/9032 (2019.01)</p> <p>(21) 출원번호 10-2023-0158138
(22) 출원일자 2023년11월15일
심사청구일자 2023년11월15일</p> | <p>(71) 출원인
인하대학교 산학협력단
인천광역시 미추홀구 인하로 100(용현동, 인하대학교)</p> <p>(72) 발명자
심정섭
인천광역시 남동구 남동대로 860, 120동 601호 (간석동, 간석래미안자이아파트)</p> <p>신현준
경기도 고양시 덕양구 토당로 93, 102동 1809호 (토당동, 한라비발디아파트)</p> <p>(74) 대리인
양성보</p> |
|---|--|

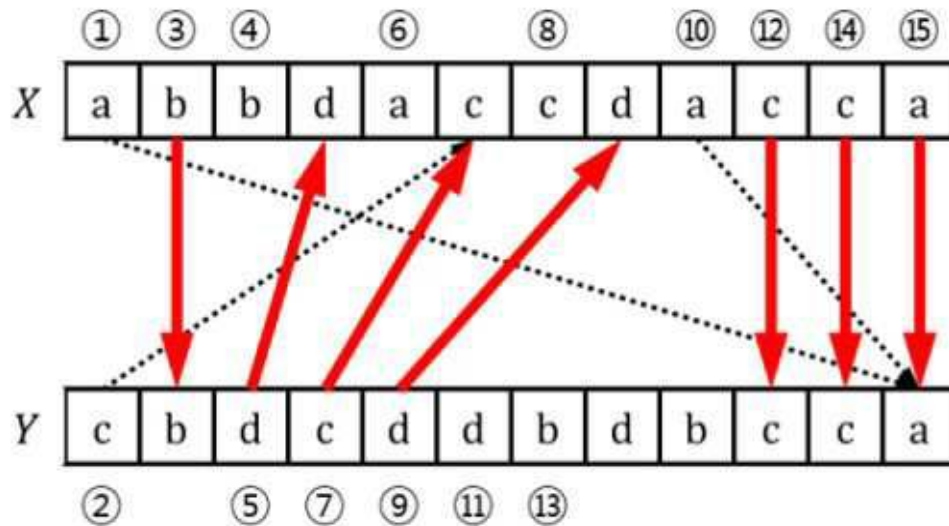
전체 청구항 수 : 총 8 항

(54) 발명의 명칭 극대공통부분서열을 찾는 개선된 알고리즘

(57) 요약

극대공통부분서열을 찾는 개선된 알고리즘이 개시된다. 일 실시예에 따른 서열 탐색 시스템에 의해 수행되는 극대공통부분서열 탐색 방법은, 제1 문자열(X), 제2 문자열(Y)의 각각의 부분문자열에 대해 탐색된 복수 개의 공통 문자들 중 공통문자를 선택하는 단계; 및 상기 선택된 공통문자를 이용하여 극대공통부분서열(Maximal Common Subsequence, MCS)을 계산하는 단계를 포함할 수 있다.

대표도 - 도4



(52) CPC특허분류

G16B 30/10 (2019.02)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711179347
과제번호	00155915
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	인공지능융합혁신인재양성
연구과제명	[Ezbaro][정부+민간] 인공지능융합혁신인재양성(2차년도)
기 여 율	1/1
과제수행기관명	인하대학교 산학협력단
연구기간	2023.01.01 ~ 2023.12.31

명세서

청구범위

청구항 1

서열 탐색 시스템에 의해 수행되는 극대공통부분서열 탐색 방법에 있어서,

제1 문자열(X), 제2 문자열(Y)의 각각의 부분문자열에 대해 탐색된 복수 개의 공통문자들 중 공통문자를 선택하는 단계; 및

상기 선택된 공통문자를 이용하여 극대공통부분서열(Maximal Common Subsequence, MCS)을 계산하는 단계를 포함하는 극대공통부분서열 탐색 방법.

청구항 2

제1항에 있어서,

상기 탐색하는 단계는,

제1 문자열(X), 제2 문자열(Y)의 각각의 부분문자열($X[x..x']$, $Y[y..y']$)이 주어졌을 때, 각 특정 부분문자열($X[x..g-1]$, $Y[y..h-1]$)의 길이를 각각 d_g , d_h 라고 정의할 때, 서로소인 각 특정 부분문자열의 길이 모두 고정 상수 미만인 공통문자의 위치 쌍을 선택하고, 위치 쌍이 존재하지 않을 경우 각 특정 부분문자열의 길이 중 하나가 고정 상수 미만인 위치 쌍을 선택하는 단계

를 포함하는 극대공통부분서열 탐색 방법.

청구항 3

제2항에 있어서,

상기 탐색하는 단계는,

상기 각각의 부분문자열에 기초하여 각 특정 부분문자열이 서로소인 위치 쌍을 탐색하는 탐색 문제를 해결하는 단계

를 포함하는 극대공통부분서열 탐색 방법.

청구항 4

제3항에 있어서,

상기 탐색하는 단계는,

각각의 부분문자열을 $X[x..x']$ 와 $Y[y..y']$ 라고 할 때, $X[x..x']$ 의 문자와 $Y[y..y']$ 의 문자에 대해 앞에서부터 next 질의를 번갈아 수행하고, i번째 문자에 대해 i값에 따라 서로소인 위치 쌍을 다른 방식으로 탐색하는 단계를 포함하는 극대공통부분서열 탐색 방법.

청구항 5

제4항에 있어서,

상기 탐색하는 단계는,

i값이 고정 상수 이하인 경우, 제1 문자열(X)의 부분 문자열의 i번째 문자에 대한 next 질의값을 획득하고, 특정 부분문자열의 길이인 d_h 가 고정 상수값 미만이면 i번째 문자와 i번째 문자에 대한 next 질의값을 포함하는 위

치 쌍을 선택하고, 특정 부분문자열의 길이인 d_k 가 제1 문자열(X)의 부분 문자열의 길이보다 작으면 번째 문자와 i 번째 문자에 대한 next 질의값을 포함하는 위치 쌍을 집합에 추가하는 단계
를 포함하는 극대공통부분서열 탐색 방법.

청구항 6

제4항에 있어서,

상기 탐색하는 단계는,

i 값이 고정 상수+1인 경우, 집합에 저장된 모든 위치 쌍 중에서 알고리즘 P에서 제시된 기준에 따라 가장 좋은 위치 쌍을 선택하는 단계

를 포함하고,

상기 알고리즘 P는, 제1 문자열(X)의 문자와 제2 문자열(Y)의 문자에 대하여 동시에 공통문자의 위치 쌍을 계산하고, 두 문자에 대한 공통문자가 모두 존재한다면 이후에 더 많은 공통문자를 찾을 가능성이 큰 위치 쌍을 선택하는 것을 특징으로 하는 극대공통부분서열 탐색 방법.

청구항 7

제4항에 있어서,

상기 탐색하는 단계는,

i 값이 고정 상수보다 큰 경우, 각각의 부분문자열과 시작 위치는 동일하지만 끝 위치는 다른 또 다른 부분문자열에 대해 집합을 공유하는 방식으로 동일한 문자에 대한 next 질의의 중복 수행을 회피하는 단계

를 포함하는 극대공통부분서열 탐색 방법.

청구항 8

서열 탐색 시스템에 있어서,

제1 문자열(X), 제2 문자열(Y)의 각각의 부분문자열에 대해 탐색된 복수 개의 공통문자들 중 공통문자를 선택하는 공통문자 선택부; 및

상기 선택된 공통문자를 이용하여 극대공통부분서열(Maximal Common Subsequence, MCS)을 계산하는 MCS 계산부를 포함하는 서열 탐색 시스템.

발명의 설명

기술 분야

[0001] 아래의 설명은 극대공통부분서열을 탐색하는 기술에 관한 것이다.

배경 기술

[0003] 문자열 X의 부분서열(Subsequence)은 X의 일부 문자들이 X에서의 순서대로 나열된 서열이다. 예를 들어, X=ABB의 부분서열은 A, B, AB, BB, ABB이다. 어떤 서열이 X와 Y의 부분서열이라면 그 서열을 X와 Y의 공통부분서열(Common Subsequence)이라고 하며, 공통부분서열 중에서 길이가 가장 긴 것을 최장공통부분서열(Longest Common Subsequence, LCS)이라고 한다. 예를 들어, X=ABB, Y=BBA의 공통부분서열은 A, B, BB이고, 공통부분서열 중에서 길이가 가장 긴 BB가 X와 Y의 최장공통부분서열이다. 길이가 n 인 두 문자열의 최장공통부분서열을 동적프로그래밍을 이용하여 $O(n^2)$ 시간에 계산할 수 있다. 최장공통부분서열은 생물정보학, 자연어처리 모델 평가 등에서 문자열 간의 유사도를 측정하는 기준으로 활용되고 있다.

[0004] 두 문자열의 극대공통부분서열(Maximal Common Subsequence, MCS)은 어떤 문자를 삽입하여도 더 긴 공통부분서열을 만들 수 없는 공통부분서열이다. 예를 들어, X=ABB, Y=BBA의 공통부분서열 A, B, BB 중에서 A, BB는 어떤

문자를 삽입하여도 X와 Y의 더 긴 공통부분서열을 만들 수 없으므로 X와 Y의 극대공통부분서열이다.

[0005] 극대공통부분서열의 정의에 의해 길이가 가장 긴 극대공통부분서열이 최장공통부분서열이며, 짧은 극대공통부분서열이 존재할 수 있다. 예를 들어, $X=ABB...BB$, $Y=BB...BBA$ 의 최장공통부분서열은 $BB...BB$ 지만, 길이가 1인 극대공통부분서열 A가 존재한다. 앞에서 언급한 바와 같이, 최장공통부분서열이 서열 간의 유사도 계산에 많이 사용되지만 계산하는데 매우 긴 시간이 소요된다. 극대공통부분서열은 최장공통부분서열보다 빠르게 계산할 수 있으므로 되도록 긴 극대공통부분서열을 찾는 문제는 중요하다.

발명의 내용

해결하려는 과제

[0007] 고정 상수 k가 주어졌을 때, $O(kn)$ 공간을 이용하여 $O(kn \sqrt{\log n / \log \log n})$ 시간에 극대공통부분서열을 계산하는 방법 및 시스템을 제공할 수 있다.

[0008] Sakai 알고리즘을 수정하여 고정 상수 k에 대해 $O(k)$ 개의 문자를 확인하고, 적절한 공통문자를 반복적으로 선택하여 극대공통부분서열을 계산하는 방법 및 시스템을 제공할 수 있다.

과제의 해결 수단

[0010] 서열 탐색 시스템에 의해 수행되는 극대공통부분서열 탐색 방법은, 제1 문자열(X), 제2 문자열(Y)의 각각의 부분문자열에 대해 탐색된 복수 개의 공통문자들 중 공통문자를 선택하는 단계; 및 상기 선택된 공통문자를 이용하여 극대공통부분서열(Maximal Common Subsequence, MCS)을 계산하는 단계를 포함할 수 있다.

[0011] 상기 탐색하는 단계는, 제1 문자열(X), 제2 문자열(Y)의 각각의 부분문자열($X[x..x']$, $Y[y..y']$)이 주어졌을 때, 각 특정 부분문자열($X[x..g-1]$, $Y[y..h-1]$)의 길이를 각각 d_g , d_h 라고 정의할 때, 서로소인 각 특정 부분문자열의 길이 모두 고정 상수 미만인 공통문자의 위치 쌍을 선택하고, 위치 쌍이 존재하지 않을 경우 각 특정 부분문자열의 길이 중 하나가 고정 상수 미만인 위치 쌍을 선택하는 단계를 포함할 수 있다.

[0012] 상기 탐색하는 단계는, 상기 각각의 부분문자열에 기초하여 각 특정 부분문자열이 서로소인 위치 쌍을 탐색하는 탐색 문제를 해결하는 단계를 포함할 수 있다.

[0013] 상기 탐색하는 단계는, 각각의 부분문자열을 $X[x..x']$ 와 $Y[y..y']$ 라고 할 때, $X[x..x']$ 의 문자와 $Y[y..y']$ 의 문자에 대해 앞에서부터 next 질의를 번갈아 수행하고, i번째 문자에 대해 i값에 따라 서로소인 위치 쌍을 다른 방식으로 탐색하는 단계를 포함할 수 있다.

[0014] 상기 탐색하는 단계는, i값이 고정 상수 이하인 경우, 제1 문자열(X)의 부분 문자열의 i번째 문자에 대한 next 질의값을 획득하고, 특정 부분문자열의 길이인 d_h 가 고정 상수값 미만이면 i번째 문자와 i번째 문자에 대한 next 질의값을 포함하는 위치 쌍을 선택하고, 특정 부분문자열의 길이인 d_h 가 제1 문자열(X)의 부분 문자열의 길이보다 작으면 번째 문자와 i번째 문자에 대한 next 질의값을 포함하는 위치 쌍을 집합에 추가하는 단계를 포함할 수 있다.

[0015] 상기 탐색하는 단계는, i값이 고정 상수+1인 경우, 집합에 저장된 모든 위치 쌍 중에서 알고리즘 P에서 제시된 기준에 따라 가장 좋은 위치 쌍을 선택하는 단계를 포함하고, 상기 알고리즘 P는, 제1 문자열(X)의 문자와 제2 문자열(Y)의 문자에 대하여 동시에 공통문자의 위치 쌍을 계산하고, 두 문자에 대한 공통문자가 모두 존재한다면 이후에 더 많은 공통문자를 찾을 가능성이 큰 위치 쌍을 선택하는 것일 수 있다.

[0016] 상기 탐색하는 단계는, i값이 고정 상수보다 큰 경우, 각각의 부분문자열과 시작 위치는 동일하지만 끝 위치는 다른 또 다른 부분문자열에 대해 집합을 공유하는 방식으로 동일한 문자에 대한 next 질의의 중복 수행을 회피하는 단계를 포함할 수 있다.

[0017] 서열 탐색 시스템은, 제1 문자열(X), 제2 문자열(Y)의 각각의 부분문자열에 대해 탐색된 복수 개의 공통문자들 중 공통문자를 선택하는 공통문자 선택부; 및 상기 선택된 공통문자를 이용하여 극대공통부분서열(Maximal Common Subsequence, MCS)을 계산하는 MCS 계산부를 포함할 수 있다.

발명의 효과

[0019] 실시예에서 제안된 방법을 통해 계산된 극대공통부분서열의 길이가 다른 알고리즘들로 계산한 극대공통부분서열의 길이보다 더 길다.

도면의 간단한 설명

[0021] 도 1은 알고리즘 P를 이용하여 문제 1을 해결하는 동작을 설명하기 위한 예이다.
 도 2는 일 실시예에 있어서, 서열 탐색 시스템의 구성을 설명하기 위한 블록도이다.
 도 3은 일 실시예에 있어서, 극대공통부분서열 탐색 방법을 설명하기 위한 흐름도이다.
 도 4는 일 실시예에 있어서, 극대공통부분서열 탐색 방법을 이용하여 문제 1을 해결하는 동작을 설명하기 위한 예이다.

발명을 실시하기 위한 구체적인 내용

[0022] 이하, 실시예를 첨부한 도면을 참조하여 상세히 설명한다.

[0024] 도 1은 알고리즘 P를 이용하여 문제 1을 해결하는 동작을 설명하기 위한 예이다.

[0025] 실시예에서는 극대공통부분서열 탐색 동작의 설명의 이해를 돕기 위해 비특허문헌 1 <Y. Sakai, "Maximal common subsequence algorithms," Theoretical Computer Science, Vol. 793, pp. 132-139, 2019>에 개시된 Sakai 알고리즘 및 비특허문헌 2 <D. Lee and J. C. Na, "An Improved Algorithm for Finding a Longer Maximal Common Subsequence," Journal of KIISE, Vol. 49, No. 7, pp. 507-513, 2022.(in Korean)>에 개시된 알고리즘 A와 P에 대해 설명하기로 한다.

[0026] 문자들의 집합 Σ 의 문자들로 구성된 문자열 X에 대해, $|X|$ 는 X의 길이를 나타낸다. $X[i] (1 \leq i \leq |X|)$ 는 X의 i번째 문자를 나타내며, $X[i..j] (1 \leq i \leq j \leq |X|)$ 는 X의 i번째 문자부터 j번째까지의 문자들로 구성된 X의 부분문자열을 나타낸다. 어떤 두 문자열 간의 공통문자가 존재하지 않는다면 이 두 문자열은 서로소라 정의한다. 문자열 X에서의 위치 i와 문자 c가 주어졌을 때, $next_X(i, c)$ 는 $X[i..|X|]$ 에서 c가 처음 나타나는 위치이고, 문자 c가 존재하지 않는다면 $next_X(i, c)$ 는 Δ 로 정의한다. 길이가 n인 문자열 X에 대한 next 질의는 $O(n)$ 시간의 전처리 이후에 $O(\sqrt{\log n / \log \log n})$ 시간에 수행된다.

[0027] Sakai 알고리즘은 극대공통부분서열(MCS)의 문자들을 찾기 위해 아래 문제 1을 정의하고, 이를 이용하여 극대공통부분서열을 찾는다.

[0028] 문제 1: 문자열 X, Y의 부분문자열 $X[x..x']$, $Y[y..y']$ 이 주어졌을 때, $X[g] = Y[h]$ 이고 $X[x..g-1]$ 과 $Y[y..h-1]$ 이 서로소인 위치 쌍 (g, h) ($x \leq g \leq x'$, $y \leq h \leq y'$)를 찾는다.

[0029] 문자열 X, Y가 주어졌을 때, Sakai 알고리즘은 X와 Y의 문자들에 대하여 앞에서부터 next 질의를 번갈아 수행하는 방식으로 문제 1을 반복적으로 해결한다. 먼저, $next_Y(y, X[x])$ 를 수행하고 그 결과가 y' 이하라면, 즉

$X[x]$ 가 $Y[y..y']$ 에 존재한다면 (g, h) 는 $(x, next_Y(y, X[x]))$ 이다. $next_Y(y, X[x])$ 가 y' 보다 크다면, 즉 $X[x]$ 가 $Y[y..y']$ 에 존재하지 않으면 $Y[y]$, $X[x+1]$, $Y[y+1]$, ...에 대해 차례로 유사한 과정을 수행하여 위치 쌍 (g, h) 을 찾는다.

[0030] Sakai 알고리즘이 문제 1을 반복적으로 수행하는 경우는 두 가지이다. 첫 번째는 $X[x..x']$, $Y[y..y']$ 와 시작 위치는 동일하지만 끝 위치는 다른 부분문자열 $X[x..x'']$ 와 $Y[y..y'']$ 에 대해 문제 1을 수행하는 경우이다. 이 경우, 기존에 $X[x..x']$ 와 $Y[y..y']$ 에 대해 next 질의를 수행한 문자들에 대해서는 next 질의를 수행하지 않는 방식으로 중복 계산을 피하여 문제 1을 효율적으로 해결한다.

[0031] 두 번째는 문제 1의 해가 존재하여 이후 문자열 $X[g+1..x']$ 과 $Y[h+1..y']$ 에 대해서 문제 1을 반복적으로 수행하는 경우이다. 따라서, 남은 문자열의 길이가 길수록 더 많은 공통문자를 찾을 가능성이 높아진다. 하지만, Sakai 알고리즘은 항상 문자 $X[x]$ 를 가장 먼저 조사하기 때문에 Y쪽의 길이가 급격히 줄어드는 문제점이 있다.

[0032] Sakai 알고리즘의 문제 1을 해결하는 과정을 개선하여 더 긴 극대공통부분서열(MCS)를 계산하는 알고리즘 A와 알고리즘 P이 제시되었다. 알고리즘 A는 X와 Y의 공통문자를 찾을 때마다 next 질의의 반복 수행을 시작하는 문자열을 바꿔서 $X[g+1..x']$ 와 $Y[h+1..y']$ 의 길이가 불균형하게 감소할 가능성을 줄인다. 알고리즘 P는 X의 문자와 Y의 문자에 대하여 동시에 공통문자의 위치 쌍을 계산하고, 두 문자에 대한 공통문자가 모두 존재한다면 이후에 더 많은 공통문자를 찾을 가능성이 큰 위치 쌍을 선택한다.

[0033] 도 1은 문자 집합 $\Sigma = \{a, b, c, d\}$ 로 구성된 문자열 $X = abbdaccdacca$ 와 $Y = cbdcddbdbcca$ 가 주어졌을 때, 알고리즘 P에서 문제 1을 반복적으로 해결하는 과정을 보여준다. next 질의가 수행되는 순서를 원 문자로 표시하였고, 문제 1에서 찾은 위치 쌍 (g, h) 을 빨간 화살표로 연결하여 표시하였다. $X[1..12]$ 와 $Y[1..12]$ 에 대한 문제 1을 해결할 때, $X[1]$ 과 $Y[1]$ 에 대해 각각 next 질의를 수행하여 대응하는 공통문자 $Y[12]$ 와 $X[6]$ 을 찾는다. l 을 $X[g+1..x']$ 과 $Y[h+1..y']$ 중 짧은 문자열의 길이라고 정의하기로 한다. 위치 쌍 $(1, 12)$ 를 선택했을 때의 l 은 0이고 $(6, 1)$ 을 선택했을 때의 l 은 6이다. l 이 더 클수록 공통문자를 찾을 가능성이 크므로, 위치 쌍 $(6, 1)$ 을 $X[1..12]$ 과 $Y[1..12]$ 에 대한 문제 1의 해로 선택한다. 다음으로, $X[7..12]$ 과 $Y[2..12]$ 에 대한 문제 1의 해로 $(7, 4)$ 를 선택한다. 이러한 과정을 반복하여 문제 1을 반복적으로 해결한 결과, 4개의 공통문자가 계산된다.

[0034] 도 2는 일 실시예에 있어서, 서열 탐색 시스템의 구성을 설명하기 위한 블록도이고, 도 3은 일 실시예에 있어서, 극대공통부분서열 탐색 방법을 설명하기 위한 흐름도이다.

[0035] 서열 탐색 시스템(100)의 프로세서는 공통문자 선택부(210) 및 MCS 계산부(220)를 포함할 수 있다. 이러한 프로세서의 구성요소들은 서열 탐색 시스템에 저장된 프로그램 코드가 제공하는 제어 명령에 따라 프로세서에 의해 수행되는 서로 다른 기능들(different functions)의 표현들일 수 있다. 프로세서 및 프로세서의 구성요소들은 도 3의 극대공통부분서열 탐색 방법이 포함하는 단계들(310 내지 320)을 수행하는 서열 탐색 시스템을 제어할 수 있다. 이때, 프로세서 및 프로세서의 구성요소들은 메모리가 포함하는 운영체제의 코드와 적어도 하나의 프로그램의 코드에 따른 명령(instruction)을 실행하도록 구현될 수 있다.

[0036] 프로세서는 극대공통부분서열 탐색 방법을 위한 프로그램의 파일에 저장된 프로그램 코드를 메모리에 로딩할 수 있다. 예를 들면, 서열 탐색 시스템에서 프로그램이 실행되면, 프로세서는 운영체제의 제어에 따라 프로그램의 파일로부터 프로그램 코드를 메모리에 로딩하도록 서열 탐색 시스템을 제어할 수 있다. 이때 프로세서는 공통문자 선택부(210) 및 MCS 계산부(220) 각각에 메모리에 로딩된 프로그램 코드 중 대응하는 부분의 명령을 실행하여 이후 단계들(310 내지 320)을 실행하기 위한 프로세서의 서로 다른 기능적 표현들일 수 있다.

[0037] 단계(310)에서 공통문자 선택부(210)는 제1 문자열(X), 제2 문자열(Y)의 각각의 부분문자열에 대해 탐색된 복수

개의 공통문자들 중 공통문자를 선택할 수 있다. 공통문자 선택부(210)는 제1 문자열(X), 제2 문자열(Y)의 각각의 부분문자열($X[x..x']$, $Y[y..y']$)이 주어졌을 때, 각 특정 부분문자열의 길이를 각각 d_g , d_h 라고 정의할 때, 서로소인 각 특정 부분문자열($X[x..g-1]$, $Y[y..h-1]$)의 길이 모두 고정 상수 미만인 공통문자의 위치 쌍을 선택하고, 위치 쌍이 존재하지 않을 경우 각 특정 부분문자열의 길이 중 하나가 고정 상수 미만인 위치 쌍을 선택할 수 있다. 공통문자 선택부(210)는 각각의 부분문자열에 기초하여 각 특정 부분문자열이 서로소인 위치 쌍을 탐색하는 탐색 문제를 해결할 수 있다. 공통문자 선택부(210)는 각각의 부분문자열을 $X[x..x']$ 와 $Y[y..y']$ 라고 할 때, $X[x..x']$ 의 문자와 $Y[y..y']$ 의 문자에 대해 앞에서부터 next 질의를 번갈아 수행하고, i번째 문자에 대해 i값에 따라 서로소인 위치 쌍을 다른 방식으로 탐색할 수 있다. 공통문자 선택부(210)는 i값이 고정 상수 이하인 경우, 제1 문자열(X)의 부분 문자열의 i번째 문자에 대한 next 질의값을 획득하고, 특정 부분문자열의 길이인 d_h 가 고정 상수값 미만이면 i번째 문자와 i번째 문자에 대한 next 질의값을 포함하는 위치 쌍을 선택하고, 특정 부분문자열의 길이인 d_h 가 제1 문자열(X)의 부분 문자열의 길이보다 작으면 번째 문자와 i번째 문자에 대한 next 질의값을 포함하는 위치 쌍을 집합에 추가할 수 있다. 공통문자 선택부(210)는 i값이 고정 상수+1인 경우, 집합에 저장된 모든 위치 쌍 중에서 알고리즘 P에서 제시된 기준에 따라 가장 좋은 위치 쌍을 선택할 수 있다. 이때, 알고리즘 P는, 제1 문자열(X)의 문자와 제2 문자열(Y)의 문자에 대하여 동시에 공통문자의 위치 쌍을 계산하고, 두 문자에 대한 공통문자가 모두 존재한다면 이후에 더 많은 공통문자를 찾을 가능성이 큰 위치 쌍을 선택하는 것일 수 있다. 공통문자 선택부(210)는 i값이 고정 상수보다 큰 경우, 각각의 부분문자열과 시작 위치는 동일하지만 끝 위치는 다른 또 다른 부분문자열에 대해 집합을 공유하는 방식으로 동일한 문자에 대한 next 질의의 중복 수행을 회피할 수 있다.

[0038] 단계(320)에서 MCS 계산부(220)는 선택된 공통문자를 이용하여 극대공통부분서열(Maximal Common Subsequence, MCS)을 계산할 수 있다. 예를 들면, MCS 계산부(220)는 선택된 공통문자를 극대공통부분서열에 추가할 수 있다.

[0039] 도 4는 일 실시예에 있어서, 극대공통부분서열 탐색 방법을 이용하여 문제 1을 해결하는 동작을 설명하기 위한 예이다.

[0040] 실시예에서는 $O(k)$ 개의 문자를 확인하여 MCS를 계산하는 알고리즘을 제시한다. 문제 1의 조건에서 서로소인 $X[x..g-1]$ 과 $Y[y..h-1]$ 의 길이를 각각 d_g , d_h 라고 정의하기로 한다. 실시예에서 제시하는 알고리즘은 d_g , d_h 모두 주어진 고정 상수 k 미만인 위치 쌍 (g, h)를 우선적으로 선택하고, 그러한 위치 쌍이 존재하지 않으면 d_g , d_h 중 하나가 k 미만인 위치 쌍을 선택한다. d_g , d_h 모두 k 이상인 위치 쌍은 가장 우선순위가 낮다.

[0041] 문제 1을 해결하기 위한 구체적인 알고리즘은 다음과 같다. $X[x..x']$ 의 문자와 $Y[y..y']$ 의 문자에 대하여 앞에서부터 next 질의를 X의 문자와 Y의 문자가 번갈아 수행한다. i번째 문자에 대한 처리는 i값에 따라 다음과 같이 세 단계로 나뉜다. 이때, 기술을 간략히 하기 위해 $X[x..x']$ 과 $Y[y..y']$ 모두 k보다 길다고 가정하고, X의 문자를 기준으로만 설명하기로 한다.

[0042] A. $i \leq k$ 인 경우: $X[x..x']$ 의 i번째 문자 $X[x+i]$ 에 대한 next 질의 값을 h라 하자. 즉, $h = \text{next}_Y(y, X[x+i])$ 이다. d_h 가 k 미만이면, 위치 쌍 $(x+i, h)$ 을 문제 1의 해로 선택하고 종료 한다. d_h 가 $X[x..x']$ 의 길이보다 작으면, 위치 쌍 $(x+i, h)$ 를 집합 $L_{x,y}$ 에 추가한다. 그렇지 않으면 해당 위치 쌍은 무시한다.

[0043] B. $i=k+1$ 인 경우: 집합 $L_{x,y}$ 에 저장된 모든 위치 쌍 중에서 알고리즘 P에서 제시된 기준에 따라 가장 좋은 위치 쌍을 문제 1의 해로 선택하고 종료한다. 만약 유효한 위치 쌍이 존재하지 않으면, 단계 C를 수행한다

[0044] C. $i > k$ 인 경우: Sakai 알고리즘과 같은 방식으로 진행한다. 앞서 설명한 바와 같이, Sakai 알고리즘은

$X[x..x']$, $Y[y..y']$ 과 시작 위치는 동일하지만 끝 위치는 다른 또 다른 부분문자열 $X[x..x'']$ 와 $Y[y..y'']$ 에 대해 문제 1을 수행하는 경우가 발생한다. 이 경우, next 질의를 처음부터 다시 수행하지 않고, 집합 $L_{x,y}$ 를 공유하는 방식으로 같은 문자에 대한 next 질의의 중복 수행을 회피한다.

[0045] 도 4는 도 1의 문자열 X와 Y에 대해 k가 2일 때, 실시예에서 제시하는 알고리즘이 문제 1을 반복적으로 해결하는 과정을 보여준다. 우선, $X[1..12]$ 와 $Y[1..12]$ 에 대해 문제 1을 해결하는 과정을 살펴본다. 먼저, X[1]을 기준으로 공통문자 위치 쌍 (1, 12)을 찾는다. 이때, d_h 는 11로 k이상이므로 문제 1의 해로 선택하지 않는다. 문자 Y[1]을 기준으로 찾은 공통문자 위치 쌍 (6, 1)도 조건에 부합하지 않는다. 여기서, 도 4에서 해로 선택되지 않은 두 위치 쌍은 점선으로 표시하였다.

[0046] 다음으로 찾은 위치 쌍 (2, 2)는 d_g 와 d_h 모두 k 미만이므로 문제 1의 해로 선택된다. 유사한 과정으로 문제 1을 반복적으로 해결하여 위치 쌍 (4, 3), (6, 4), (8, 5)를 찾을 수 있다

[0047] 다음으로, $X[9..12]$ 과 $Y[6..12]$ 에 대한 문제 1을 해결하는 과정을 고려해보기로 한다. 단계 A에서 X[9], Y[6], X[10], Y[7]에 대해 next 질의를 수행하여 찾은 위치 쌍 (9, 6), (∞ , 6), (10, 10), (∞ , 7)는 모두 d_g 또는 d_h 가 k 이상이므로, 단계 B에서 앞서 찾은 위치 쌍 중에서 l이 가장 큰 위치 쌍 (10, 10)을 문제 1의 해로 선택한다. 이러한 과정을 반복하여 실시예에서 제시하는 알고리즘에서는 7개의 공통문자가 계산되며, 이는 알고리즘 P로 계산된 공통문자의 개수보다 3개 더 많다.

[0048] 실시예에서 제시하는 알고리즘의 시간 복잡도와 공간 복잡도를 분석한다. 단계 C는 Sakai 알고리즘과 동일하게 수행되므로, 실시예에서 제시하는 알고리즘에서는 Sakai 알고리즘에 비해 단계 A, B가 추가적으로 수행된다. 하나의 문제 1을 해결하는 과정에서 $L_{x,y}$ 에 최대 $2k$ 개의 위치 쌍이 저장된다. 따라서, 단계 A에서 next 질의를 이용하여 $L_{x,y}$ 를 생성하는데 $O(k\sqrt{\log n/\log \log n})$ 시간이 필요하다. 단계 B에서 하나의 위치 쌍에 대해 l을 계산하는데 상수 시간이 걸리므로, 단계 B는 $O(k)$ 시간이 필요하다. Sakai 알고리즘에 의하면 문제 1은 $O(n)$ 번 발생하므로 전체 알고리즘 수행 과정에서 생성되는 집합 $L_{x,y}$ 의 개수는 $O(nk)$ 이다. 그러므로, 실시예에서 제시하는 알고리즘은 $O(kn)$ 공간을 이용하여 $O(kn\sqrt{\log n/\log \log n})$ 시간에 수행된다.

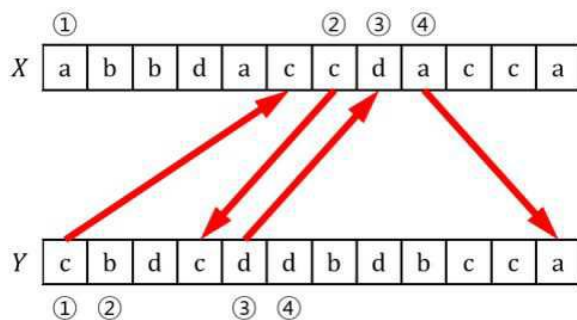
[0049] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 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.

[0050] 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, 가상장치(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.

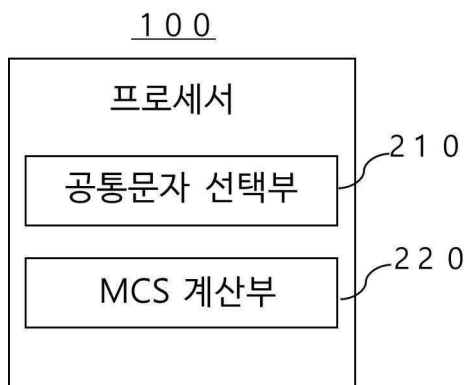
- [0051] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic 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.
- [0052] 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.
- [0053] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

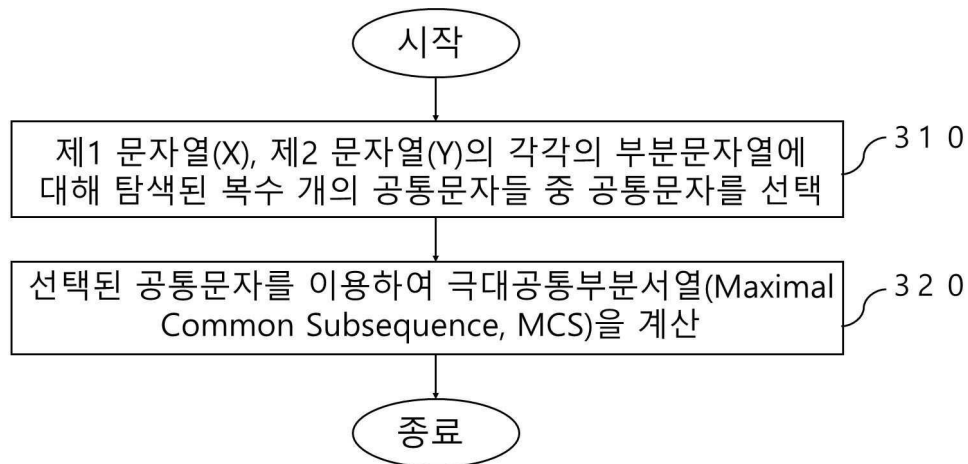
도면1



도면2



도면3



도면4

