



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년11월30일
(11) 등록번호 10-1088290
(24) 등록일자 2011년11월23일

(51) Int. Cl.

G06F 17/30 (2006.01) G06F 9/44 (2006.01)

(21) 출원번호 10-2011-0042413

(22) 출원일자 2011년05월04일

심사청구일자 2011년05월04일

(56) 선행기술조사문헌

T.Jiang and V.G. Timkovsky, Shortest consistent superstrings computable in polynomial time, Theoretical Computer Science, vol.143, no.1, pp.113-122, 1995*

한국정보과학회- 한국컴퓨터

학술발표논문집(2010.11)

한국정보과학회-정보과학회논문지(2009)

Finding the LCNSS in linear time(2009.07.01)

*는 심사관에 의하여 인용된 문헌

(73) 특허권자

인하대학교 산학협력단

인천 남구 용현동 253 인하대학교

(72) 발명자

심정섭

인천광역시 남동구 간석2동 금호어울림마을 110동 1101호

조석현

인천광역시 남구 학익동 풍림아이원아파트 105동 501호

윤현철

인천광역시 남구 용현4동 84-6번지 신유성빌라 가동 102호

(74) 대리인

특허법인무한

전체 청구항 수 : 총 4 항

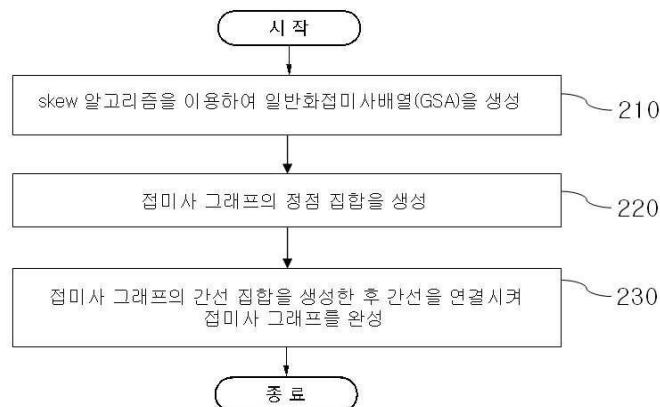
심사관 : 석상문

(54) 접미사 배열을 이용한 최장공통비상위문자열 검색 방법

(57) 요약

접미사 배열을 이용한 최장공통비상위문자열 검색 방법이 개시된다. 최장공통비상위문자열 검색 방법은 문자열 집합에 대하여 접미사배열(suffix array)을 이용하여 접미사 그래프를 생성하는 단계; 및 접미사 그래프를 이용하여 문자열 집합에서 가장 긴 문자열에 해당되는 최장공통비상위문자열을 검색하는 단계를 포함하며, 접미사 그래프를 생성하는 단계는 문자열 집합 내 각 문자열의 접미사를 사전 순으로 배열한 일반화접미사배열(generalized suffix array)을 생성하는 단계; 일반화접미사배열에 대하여 접미사 그래프의 정점 집합을 생성하는 단계; 및 일반화접미사배열에 대한 최근접접두사(nearest prefix)와 link 함수를 이용하여 접미사 그래프의 간선 집합을 생성하는 단계를 포함할 수 있다.

대표도 - 도2



이 발명을 지원한 국가연구개발사업

과제고유번호 2010-0028134

부처명 교육과학기술부

연구관리전문기관 한국연구재단

연구사업명 기본연구

연구과제명 문자열 포함 및 불포함 문제해결 알고리즘 및 응용 기술개발

기여율 1/1

주관기관 인하대학교 산학협력단

연구기간 2010.09.01 ~ 2011.08.31

특허청구의 범위

청구항 1

삭제

청구항 2

접미사 그래프 모델에 기반한 최장공통비상위문자열 검색 방법에 있어서,

문자열 집합에 대하여 접미사배열(suffix array)을 이용하여 접미사 그래프를 생성하는 단계; 및

상기 접미사 그래프를 이용하여 상기 문자열 집합에서 가장 긴 문자열에 해당되는 최장공통비상위문자열을 검색하는 단계

를 포함하고,

상기 접미사 그래프를 생성하는 단계는,

상기 문자열 집합 내 각 문자열의 접미사를 사전 순으로 배열한 일반화접미사배열(generalized suffix array)을 생성하는 단계;

상기 일반화접미사배열에 대하여 상기 접미사 그래프의 정점 집합을 생성하는 단계; 및

상기 일반화접미사배열에 대한 최근접접두사(nearest prefix)와 link 함수를 이용하여 상기 접미사 그래프의 간선 집합을 생성하는 단계

를 포함하는 최장공통비상위문자열 검색 방법.

청구항 3

제2항에 있어서,

상기 일반화접미사배열을 생성하는 단계는,

상기 문자열 집합에 속한 각 문자열 끝에 종결문자를 각각 붙인 문자열 집합을 이용하여 상기 일반화접미사배열을 생성하는 것

을 특징으로 하는 최장공통비상위문자열 검색 방법.

청구항 4

제2항에 있어서,

상기 접미사 그래프의 정점 집합을 생성하는 단계는,

상기 일반화접미사배열 상의 대표 접미사이면서 상기 문자열 집합에 속하지 않는 접미사에 대하여 정점을 생성하는 것

을 특징으로 하는 최장공통비상위문자열 검색 방법.

청구항 5

제2항에 있어서,

상기 접미사 그래프의 간선 집합을 생성하는 단계는,

상기 일반화접미사배열에 속한 접미사의 길이를 나타내는 배열인 Len 배열과 상기 일반화접미사배열에 대한 최장공통접두사(longest common prefix) 배열인 LCP 배열을 이용하여 상기 일반화접미사배열 상의 대표 접미사에 대한 최근접접두사 배열을 구한 후, 상기 최근접접두사 배열을 이용하여 상기 정점 집합에 속한 정점 간의 간선을 나타내는 링크 함수를 계산하는 것

을 특징으로 하는 최장공통비상위문자열 검색 방법.

명세서

기술분야

[0001] 본 발명의 실시예들은 접미사 배열을 이용하여 접미사 그래프 모델을 생성하고 이를 통해 최장공통비상위문자열을 찾는 알고리즘에 관한 것이다.

배경기술

[0002] 주어진 문자열 집합에 대한 포함/불포함 문제가 압축 알고리즘, 분자 생물학, 컴퓨터 보안 등 다양한 분야에서 필요성이 대두되어 진행되고 있다.

[0003] 문자열 포함 문제는 최장공통부분서열(Longest Common Subsequence), 최단공통상위서열(Shortest Common Supersequence), 최장공통부분문자열(Longest Common Substring)과 최단공통상위문자열(Shortest Common Superstring) 등이 연구되어 왔고, 문자열 불포함 문제는 최장공통비상위서열(Longest Common Non-Supersequence), 최단공통비부분서열(Shortest Common Non-Subsequence), 최장공통비상위문자열(Longest Common Non-Superstring)과 최단공통비부분문자열(Shortest Common Non-Substring) 등이 연구되어 왔다.

[0004] 이 중 최장공통비상위문자열 문자에 관하여 설명하면 다음과 같다.

[0005] 상수 크기 문자 집합에 대한 문자열 집합 $F = \{f_1, f_2, \dots, f_m\}$ 와 문자열 x 를 고려해 보자. 만약 x 가 모든 $f_i (1 \leq i \leq m)$ 에 대해서 상위문자열(superstring)이 아니라면, x 를 F 에 대한 공통비상위문자열(common non-superstring, 이하 CNSS로 표기)이라 한다. 만약 x 가 F 에 대한 공통비상위문자열 중 가장 긴 문자열이라면 x 를 F 에 대한 최장공통비상위문자열(longest common non-superstring, 이하 LCNSS로 표기)이라 한다. S 를 F 에 속한 문자열들의 접미사 집합이라 하자. Rubinov와 Timkovsky는 그래프 상의 경로가 F 의 CNSS와 대응되는 S 에 대한 유향 그래프 $\Gamma_S = (V, E)$ 를 정의했다. Γ_S 가 사이클이 존재할 경우, F 에 대한 LCNSS는 존재하지 않는다. 그렇지 않다면, 즉 Γ_S 가 사이클이 없는 유향 그래프일 경우, Γ_S 의 최장 경로는 F 에 대한 LCNSS를 나타내며, $O(|V| + |E|)$ 시간에 찾을 수 있다. 따라서, LCNSS 문제를 효율적으로 해결하기 위해서는 Γ_S 를 효율적으로 생성해야 한다.

발명의 내용

해결하려는 과제

[0006] LCNSS를 찾기 위하여 접미사 배열(suffix array)을 이용하여 접미사 그래프 모델 Γ_S 를 효율적으로 생성하는 알고리즘을 제공한다.

과제의 해결 수단

[0007] 문자열 집합에 대하여 접미사배열(suffix array)을 이용하여 접미사 그래프를 생성하는 단계; 및 접미사 그래프를 이용하여 문자열 집합에서 가장 긴 문자열에 해당되는 최장공통비상위문자열을 검색하는 단계를 포함하는 최장공통비상위문자열 검색 방법이 제공된다.

[0008] 일측에 따르면, 접미사 그래프를 생성하는 단계는 문자열 집합 내 각 문자열의 접미사를 사전 순으로 배열한 일반화접미사배열(generalized suffix array)을 생성하는 단계; 일반화접미사배열에 대하여 접미사 그래프의 정점 집합을 생성하는 단계; 및 일반화접미사배열에 대한 최근접접두사(nearest prefix)와 link 함수를 이용하여 접미사 그래프의 간선 집합을 생성하는 단계를 포함할 수 있다.

[0009] 다른 측면에 따르면, 일반화접미사배열을 생성하는 단계는 문자열 집합에 속한 각 문자열 끝에 종결문자를 각각 붙인 문자열 집합을 이용하여 일반화접미사배열을 생성할 수 있다.

[0010] 또 다른 측면에 따르면, 접미사 그래프의 정점 집합을 생성하는 단계는 일반화접미사배열 상의 대표 접미사이면서 문자열 집합에 속하지 않는 접미사에 대하여 정점을 생성할 수 있다.

[0011] 또 다른 측면에 따르면, 접미사 그래프의 간선 집합을 생성하는 단계는 일반화접미사배열에 속한 접미사의 길이를 나타내는 배열인 Len 배열과 일반화접미사배열에 대한 최장공통접두사(longest common prefix) 배열인 LCP

배열을 이용하여 일반화접미사배열 상의 대표 접미사에 대한 최근접접두사 배열을 구한 후, 최근접접두사 배열을 이용하여 정점 집합에 속한 정점 간의 간선을 나타내는 링크 함수를 계산할 수 있다.

발명의 효과

[0012] 접미사 배열(suffix array)을 이용하여 접미사 그래프 모델 Γ_S 를 생성함으로써 최장공통비상위문자열을 보다 빠르고 효율적으로 찾을 수 있다. 본 발명의 실시예에서 제시된 알고리즘은 문자열 집합 F 에 속한 모든 문자열의 길이의 합을 $\|F\|$ 라 했을 때, 상수 크기 문자 집합 Σ 에 대해, $O(\|F\|)$ 시간이 소요된다. 접미사배열을 이용한 알고리즘이 접미사트리를 이용한 알고리즘 등 보다 빠른 이유는 중간 자료구조인 접미사배열이 접미사트리보다 생성시간이 빠르기 때문이다. 따라서, 접미사배열을 이용할 경우, 보다 빠르게 접미사 그래프 Γ_S 를 생성할 수 있어 최장공통비상위문자열 문제 또한 더욱 신속하게 해결할 수 있다.

도면의 간단한 설명

[0013] 도 1은 본 발명의 실시예에 있어서, 접미사 그래프를 설명하기 위한 것으로 일례의 문자열 $F=\{aaa, bab, aab, bbb\}$ 에 대한 접미사 그래프를 도시한 것이다.

도 2는 본 발명의 실시예에 있어서, 접미사배열을 이용하여 LCNSS를 찾기 위한 접미사 그래프를 생성하는 알고리즘의 순서도를 도시한 것이다.

도 3과 4는 본 발명의 실시예에 있어서, 접미사 그래프의 간선 집합을 생성하기 위해 link 함수를 계산하는 알고리즘을 도시한 것이다.

발명을 실시하기 위한 구체적인 내용

[0014] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0015] 본 발명의 실시예에서는 접미사 배열을 이용하여 접미사 그래프 모델을 효율적으로 생성함으로써 LCNSS(최장공통비상위문자열)을 찾는 새로운 알고리즘을 제공한다. 특히, 본 발명의 실시예에 따른 최장공통비상위문자열 검색 방법은 컴퓨터보안 분야에서 패킷 내의 악성패턴(즉, 금지문자열)을 검출하는 침입탐지시스템에 적용될 수 있다.

[0016] 본 명세서에서 쓰이는 기호는 다음과 같다.

[0017] Σ 는 문자 집합을 나타내며 $|\Sigma|$ 는 Σ 의 크기를 나타낸다. 본 발명의 실시예에서는 $|\Sigma|$ 를 상수로 가정한다. $F=\{f_1, f_2, \dots, f_m\}$ 는 입력으로 주어지는 문자열 집합이며, m 은 문자열 집합 F 의 원소 개수를 나타낸다. $\|F\|$ 는 집합 F 에 속한 모든 문자열의 길이의 합을 나타낸다. 문자열 A 의 길이를 $|A|$ 로 표기하고, A 의 i 번째 문자는 $A[i]$ 로 표기한다. 또한, 문자열 A 의 i 번째부터 j 번째까지의 부분문자열을 $A[i..j]$ 로 표기한다. $A[i..|A|]$ 를 A 의 접미사(suffix)라 하고, 특히 $i>1$ 일 때, $A[i..|A|]$ 를 A 의 진접미사(proper suffix)라 한다. $A[i..j]$ 를 A 의 접두사(prefix)라 하고, 특히 $j<|A|$ 일 때, $A[1..j]$ 를 A 의 진접두사(proper prefix)라 한다.

[0018] LCNSS 문제를 정의하면 다음과 같다.

[0019] 정의 1: 최장공통비상위문자열 문제(LCNSS 문제)

[0020] 입력: Σ 에 대한 문자열 집합 $F=\{f_1, f_2, \dots, f_m\}$

[0021] 출력: 무한히 긴 길이의 CNSS를 생성할 수 있다면, LCNSS는 존재하지 않는다. 만약 LCNSS가 하나 이상 존재한다면 그 중 하나를 출력한다.

[0022] 본 발명의 실시예에서는 F 에 대해 다음 두 가지 조건을 가정한다.

[0023] (1) F 내의 어떤 문자열도 F 에 속한 다른 문자열의 부분문자열이 아니다: 만약 f_i 가 f_j 의 부분문자열이라면 f_i 의 비상위문자열은 f_j 의 비상위문자열이기도 하다.

[0024] (2) F 에는 Σ 에 속한 각 문자 σ 에 대해 $\sigma^k(k \geq 2)$ 인 형태의 문자열이 존재한다: 만약 σ^k 가 없다면 σ 를 무한

히 연결(concatenation)시킨 문자열이 CNSS가 되므로 LCNSS는 존재하지 않는다.

[0025] 접미사 그래프 모델 Γ_s 를 이용하여 LCNSS 문제를 모델링 할 수 있다. $\Gamma_s=(V,E)$ 는 다음과 같이 정의된다. 우선 $S=\{s_1, \dots, s_n\}$ 를 F 에 속한 각 문자열 f_i 의 진접미사들의 집합이라 하자.

[0026] 1) 정점 집합 V

[0027] 각 $s_i \in S(1 \leq i \leq n)$ 에 대해, v_i 가 정의되고 그 외의 정점은 V 내에 존재하지 않는다. 그러면 V 와 S 사이의 일대일 대응관계가 성립한다.

[0028] 2) 간선 집합 E

[0029] $\sigma s_i (s_i \in S, \sigma \in \Sigma)$ 인 문자열에 대한 집합

[0030] $P(\sigma s_i)=\{x|x \in SUF, x \text{는 } \sigma s_i \text{의 접두사}\}$ 를 정의한다. 그리고, $P(\sigma s_i)$ 에 속한 문자열 중 최장 문자열이 $s_j \in S$ 일 경우 $v_j(s_j \text{에 대응})$ 로부터 $v_i(s_i \text{에 대응})$ 로의 간선을 연결 해준다. 이때, 간선의 레이블은 σ 이다.

[0031] 이로부터 각 정점이 최대 $|\Sigma|$ 개수의 진입간선을 갖는다는 사실을 알 수 있다.

[0032] <예제 1>

[0033] $F=\{aaa, bab, aab, bbb\}$ 일 때, F 에 대한 진접미사 집합 $S=\{a, aa, ab, b, bb\}$ 이다. 따라서, Γ_s 는 5개의 정점으로 구성된다. $S_5(=bb)$ 를 고려해 보자. σ 가 a 일 때 $\sigma s_5(=abb)$ 의 접미사는 $\{a, ab, abb\}$ 이며 이 중 SUF 에 포함된 접미사는 $\{a, ab\}$ 이다. 따라서, $P(\sigma s_5)=\{a, ab\}$ 이고, $P(\sigma s_5)$ 에 속한 문자열 중 최장 문자열은 ab 이다. 그러므로 v_3 로부터 v_5 로의 간선을 연결해준다. σ 가 b 일 때, $\sigma s_5(=bbb)$ 의 접미사는 $\{b, bb, bbb\}$ 이며 이 중 SUF 에 포함된 접미사 역시 $\{b, bb, bbb\}$ 이다. 따라서, $P(\sigma s_5)=\{b, bb, bbb\}$ 이고, $P(\sigma s_5)$ 에 속한 문자열 중 최장 문자열은 bbb 이다. 하지만 bbb 는 S 내에 존재하지 않기 때문에 간선이 존재하지 않는다. $F=\{aaa, bab, aab, bbb\}$ 에 대한 접미사 그래프 Γ_s 는 도 1과 같다.

[0034] 위와 같은 방식으로 접미사 그래프 Γ_s 를 정의 했을 때, 접미사 그래프 Γ_s 와 LCNSS 문제 사이의 관계는 다음 보조정리 1과 같다.

[0035] 보조정리 1

[0036] 만약 Γ_s 그래프가 사이클이 없는 유향 그래프일 경우, F 에 대한 LCNSS는 Γ_s 의 최장 경로를 이용해 구할 수 있다. (1) 마지막 정점 전까지 최장 경로 상의 모든 정점들에 대해, 정점과 대응하는 각 문자열의 첫 번째 문자를 연속적으로 연결(concatenation)시킨다. (2) 최장 경로의 마지막 정점에 대응하는 문자열을 모두 연결(concatenation)시킨다.

[0037] <예제 2>

[0038] 도 1의 접미사 그래프 Γ_s 에 대한 최장 경로는 $\langle v_3, v_5, v_4, v_2, v_1 \rangle$ 과 같다. $F=\{aaa, bab, aab, bbb\}$ 에 대한 LCNSS는 상기한 보조정리 1에 의해 $abbbaa$ 이다.

[0039] 상기에서 정의한 LCNSS 문제는 접미사 그래프 Γ_s 를 이용하여 $O(\|F\|)$ 시간에 해결할 수 있다. 접미사 그래프를 이용하는 알고리즘에서는 접미사 그래프 Γ_s 를 생성하기 위하여 일반화접미사트리(generalized suffix tree)를 이용한다. 한편, 접두사 그래프 모델을 이용하여 LCNSS를 찾을 수도 있다. 여기서, 접두사 그래프는 Aho-Corasick 알고리즘을 이용하여 생성될 수 있으며, 접두사 그래프를 이용한 알고리즘은 접두사 그래프의 최장 경로를 구한 후 그 경로 상에 있는 간선이 나타내는 문자를 연결시킴으로써 $O(\|F\|)$ 시간에 LCNSS를 찾는다.

[0040] 본 발명의 실시예에서는 접미사 그래프 Γ_s 를 이용하여 LCNSS를 찾되, 접미사 그래프 Γ_s 를 생성하기 위하여 일반화접미사배열(generalized suffix arrays)을 이용한다. 따라서, 이하에서는 일반화접미사배열을 이용하여 접

미사 그래프 Γ_s 를 생성하는 새로운 알고리즘에 대해 설명하기로 한다.

- [0041] 본 발명의 실시예에서, 접미사 그래프 Γ_s 생성 알고리즘은 총 세 단계로 구성된다. 도 2를 참조하면, 접미사 그래프 Γ_s 생성 알고리즘의 첫 번째 단계에서는 skew 알고리즘을 이용하여 일반화접미사배열(이하, GSA로 표기)을 생성하고(210), 두 번째 단계에서는 접미사 그래프 Γ_s 의 정점 집합 V 를 생성한다(220). 마지막으로, 세 번째 단계에서는 접미사 그래프 Γ_s 의 간선 집합 E 를 생성하기 위해 필요한 최근접접두사(nearest prefix)와 $\text{link}(v_i, \sigma)$ 함수를 계산하고, 접미사 그래프 Γ_s 의 간선을 연결시켜 접미사 그래프 Γ_s 를 완성한다(230). skew 알고리즘은 <J. Karkkainen, and P. Sanders, Simple linear work suffix array construction, In Proc. 30th International Colloquium on Automata, Languages and Programming, 943-955, 2003.>에서 소개하고 있다.
- [0042] 이하에서, 접미사 그래프 Γ_s 생성 알고리즘의 세 단계를 구체적으로 설명하기로 한다.
- [0043] GSA를 생성하는 단계(210)
- [0044] GSA는 문자열 집합이 주어졌을 때, 각 문자열의 접미사들을 사전순(lexicographic order)으로 정렬한 배열이며 접미사들의 위치를 저장한다. GSA는 접미사 그래프 Γ_s 를 생성하는데 기반이 되는 정보를 제공하며, GSA를 생성하기 위해 F 에 속한 문자열 끝에 종결문자를 붙인 F' 을 이용한다.
- [0045] 종결 문자 $\$ (1 \leq i \leq m)$ 는 사전 순으로 Σ 내의 어떤 문자보다 빠르다고 가정한다. 또한 서로 다른 임의의 두 종결문자 $\$_i, \$_j$ 에 대해서 $i < j$ 일 경우 $\$_i$ 가 $\$_j$ 보다 빠르다고 가정한다. $F = \{f_1, f_2, \dots, f_m\}$ 일 때, $F' = \{f_1\$_1, f_2\$_2, \dots, f_m\$_m\}$ 이라 하자. 이 때, F' 에 대한 GSA는 두 정수의 쌍 $\langle i, j \rangle$ 를 저장하며, 이는 F' 의 문자열 f'_i 의 접미사 $f'_i[j..|f'_i|]$ 를 나타낸다.
- [0046] 접미사 배열은 문자열의 길이에 대해 선형시간에 구할 수 있다. 본 발명의 실시예에서는 skew 알고리즘을 이용하여 F' 에 대한 GSA를 $O(\|F\|)$ 시간에 구하고, GSA에 대한 최장공통접두사(longest common prefix, 이하 LCP로 표기) 배열을 $O(\|F\|)$ 시간에 구할 수 있다.
- [0047] 본 발명에서 F' 에 대한 GSA를 생성하는 이유는 skew 알고리즘을 적용하는 과정에서 마지막 문자가 유일해야 한다는 속성이 GSA를 재귀적으로 구하는 과정에서 중요하기 때문이다. 표 1은 $F' = \{aaa\$_1, bab\$_2, aab\$_3, bbb\$_4\}$ 에 대한 GSA와 LCP 배열이다.

표 1

index	GSA	F'	LCP	Len	npX	Γ_S
1	$\langle 1, 4 \rangle$	$\$1$	-	0	-	v_λ
2	$\langle 2, 4 \rangle$	$\$2$	0	0	-	
3	$\langle 3, 4 \rangle$	$\$3$	0	0	-	
4	$\langle 4, 4 \rangle$	$\$4$	0	0	-	
5	$\langle 1, 3 \rangle$	$a\$1$	0	1	1	v_1
6	$\langle 1, 2 \rangle$	$aa\$1$	1	2	5	v_2
7	$\langle 1, 1 \rangle$	$aaa\$1$	2	3	6	
8	$\langle 3, 1 \rangle$	$aab\$3$	2	3	6	
9	$\langle 2, 2 \rangle$	$ab\$2$	1	2	5	v_3
10	$\langle 3, 2 \rangle$	$ab\$3$	2	2	-	
11	$\langle 2, 3 \rangle$	$b\$2$	0	1	1	v_4
12	$\langle 3, 3 \rangle$	$b\$3$	1	1	-	
13	$\langle 4, 3 \rangle$	$b\$4$	1	1	-	
14	$\langle 2, 1 \rangle$	$bab\$2$	1	3	11	
15	$\langle 4, 2 \rangle$	$bb\$4$	1	2	11	v_5
16	$\langle 4, 1 \rangle$	$bbb\$4$	2	3	15	

[0048]

[0049]

접미사 그래프 Γ_S 의 정점 집합 V 를 생성하는 단계(220)

[0050]

앞으로 특별한 언급이 없다면, GSA의 원소가 나타내는 접미사는 종결문자를 제외한 접미사를 의미한다. 또한, GSA 상의 접미사라 함은 GSA의 각 원소가 나타내는 접미사를 의미하며, 접미사 α 의 GSA 상의 인덱스라 함은 $GSA[k]$ 가 나타내는 접미사가 α 일 때 이 때의 인덱스 k 를 의미한다. 대표접미사 $R(\alpha)$ 는 α 의 GSA 상의 인덱스 중 가장 작은 인덱스 값을 가지며, GSA 상에서 접미사 α 를 대표하는 GSA의 원소를 의미한다.

[0051]

접미사 그래프 Γ_S 의 정점 v_i 는 $R(\alpha)$ 이면서 접미사 α 가 F 에 포함되지 않는 GSA의 원소에 대해서 생성한다. 한편, GSA 상에서 인접한 두 문자열이 동일한 문자열일 경우 두 문자열간의 길이가 같고, 두 문자열간의 LCP 값이 두 문자열의 길이와 같다는 사실을 알 수 있다. 그리고, 이를 통해 GSA의 원소가 $R(\alpha)$ 인지를 확인할 수 있다. 또한, 접미사 α 가 F 에 속할 경우 $GSA[R(\alpha)]$ 값은 항상 $\langle i, 1 \rangle$ 이므로 상수시간에 F 에 대한 포함 여부를 확인할 수 있다. 또한, v_λ (표 1)는 빈 문자열을 나타내는 정점으로 실제 접미사 그래프 Γ_S 의 정점에는 포함되지 않고, link 함수의 계산을 위해서만 사용되는 특수한 정점이다.

[0052]

접미사 그래프 Γ_S 의 간선 집합 E 를 생성하는 단계(230)

[0053]

접미사 그래프 Γ_S 의 간선 집합 E 를 생성하기 위해서는 최근접접두사(nearest prefix)와 $link(v_i, \sigma)$ 함수를 계산한다.

[0054]

우선, 접미사 그래프 Γ_S 의 간선 집합 E 를 생성하는 단계(230)에서 사용될 기호에 대해서 알아보자. Len 배열은 접미사의 길이를 나타내는 배열로서, $Len[k]$ 는 $GSA[k]$ 가 나타내는 접미사(종결문자 제외)의 길이를 저장한다. $str(v_i)$ 는 $v_i(\in V)$ 에 대응되는 문자열 $s_i(\in S)$ 를 나타낸다. 또한, $ver(s_i)$ 는 s_i 에 대응되는 정점 v_i 를 나타낸다. $link(v_i, \sigma)$ 는 다음과 같이 정의된다. $str(v_i)$ 에 대한 $P(\sigma s_i)$ 의 최장 문자열이 s_j 일 경우에는

$ver(s_j)$ 를 반환하고, $f_j(\in F)$ 일 경우에는 null을 반환한다. 최근접접두사 배열 npx 는 link함수를 구하기 위해 사용되며, $npx[k]$ 는 $GSA[k]$ 가 나타내는 접미사 β 의 진접두사 중 GSA 상에 존재하는 가장 긴 접두사 α 의 $R(\alpha)$ 값으로 정의된다. 또한, α 는 SUF 에 속한 문자열 중 β 의 최장진접두사(longest proper prefix)이기도 하다.

[0055] <예제 3>

[0056] 표 1에서 $link(v_1, a)[=v_2]$ 의 경우를 살펴보자. $str(v_1)=a$ 며 $P(aa)=\{a, aa\}$ 의 최장 문자열은 aa 이다. 그리고, $aa \in S$ 이므로 $ver(aa)$ 인 v_2 를 반환한다. 또한 $link(v_2, a)$ 의 경우, $str(v_2)=aa$ 이고 $P(aaa)=\{a, aa, aaa\}$ 의 최장 문자열은 aaa 이다. 하지만 $aaa \in F$ 이므로 null을 반환한다.

[0057] 표 1에서 $GSA[8]$ 이 나타내는 접미사는 aab 이다. aab 의 진접두사는 $\{a, aa\}$ 이다. 이 중 GSA 상에 존재하는 접두사는 $\{a, aa\}$ 이고, 가장 긴 접두사인 aa 의 $R(aa)$ 는 6이다. 따라서, $npx[8] = 6$ 이 된다. $npx[14]$ 의 경우, $GSA[14]$ 이 나타내는 접미사 bab 의 진접두사는 $\{b, ba\}$ 이다. 이 중 GSA 상에 존재하는 접두사는 $\{b\}$ 이다. 따라서, 가장 긴 접두사는 b 가 되고 $R(b)$ 는 11이므로 $npx[14] = 11$ 이다.

[0058] link함수를 계산하기 위해 우선 npx 배열을 구한다. npx 배열은 대표접미사 $R(\alpha)$ 에 대해서만 구하며, Len 과 LCP 배열의 정보를 이용하여 구한다. 도 3은 스택에 기반하여 npx 를 구하는 알고리즘(이하, 'GetNPX 알고리즘'이라 칭함)이다.

[0059] 도 3에 도시한 바와 같이, GetNPX 알고리즘에서 4번 문장은 대표접미사 $R(\alpha)$ 인지 확인하는 문장으로 GSA 상에서 인접한 두 문자열간의 문자열의 길이가 같고, 두 문자열간의 LCP 값이 두 문자열의 길이와 같다면, 동일한 문자열임을 알 수 있다. 도 3에 도시한 GetNPX 알고리즘에서 7번 내지 10번 문장은 스택을 이용하여 npx 값을 구하는 문장으로 GSA 가 접미사에 대해 정렬되어 있으므로 스택에 있는 접미사 중 가장 긴 접미사 α 의 길이가 i 번째 접미사 β 의 LCP 값보다 작거나 같다면 β 의 npx 값임을 알 수 있다.

[0060] GetNPX 알고리즘의 수행 시간은 도 3의 7, 8번 문장의 수행 횟수에 비례한다. top 의 값은 초기값이 1이고, 11번 문장에서 top 의 값이 1씩 증가한다. 전체 루프가 최대 $\|F'\| - 1$ 번 수행되므로 top 값의 상한은 $\|F'\|$ 임을 알 수 있다. 따라서, 8번 문장은 최대 $\|F'\|$ 번 수행되며, 7번 문장은 최대 $2 \times \|F'\|$ 번 수행된다. 즉, GetNPX 알고리즘의 총 수행시간은 $O(\|F'\|)$ 이다. 또한, $\|F'\| = \|F\| + |F|$ 이므로 GetNPX 알고리즘의 총 수행시간은 $O(\|F\|)$ 와 같다.

[0061] GetNPX 알고리즘 수행 시 $R(\alpha)$ 이면서 α 가 F 에 포함되지 않는 GSA 의 원소를 확인함으로써 부가적으로 접미사 그래프 Γ_s 의 정점도 생성할 수 있다.

[0062] 도 4는 link함수를 계산하는 알고리즘(이하, 'calLink 알고리즘'이라 칭함)이다. calLink 알고리즘에서는 $\delta(x, y)$, $char(x, y)$ 함수를 새로이 정의한다. $\delta(x, y)$ 는 $GSA[k] = \langle x, y \rangle$ 일 때 $GSA[k]$ 가 나타내는 접미사 α 에 대한 $ver(\alpha)$ 이다. $char(x, y)$ 는 $f_x(\in F)$ 의 y 번째 문자 $f_x[y]$ 를 가리킨다.

[0063] $link(v_i, \sigma)$ 함수를 계산하는 과정은 두 가지 경우로 나누어 생각할 수 있다. 우선 임의의 σ 에 대해 $\sigma str(v_i)$ 가 GSA 상에 존재할 경우(도 4에서 문장 10-14) link함수의 정의에 따라 $link(v_i, \sigma)$ 는 $\sigma str(v_i) \in F$ 일 경우 null값을 $\sigma str(v_i) \in S$ 일 경우 $ver(\sigma str(v_i))$ 값을 가진다. 두 번째 경우(도 4에서. 문장 6-9)는 $\sigma str(v_i)$ 가 GSA 상에 존재하지 않을 경우이다. $str(v_i)$ 의 대표접미사 $R(str(v_i))$ 의 npx 값을 k ($=npx[R(str(v_i))]$)라 하고, $GSA[k]$ 가 나타내는 접미사를 α 라 하자. 이때 $link(v_i, \sigma)$ 는 $ver(\alpha)$ 의 $link(ver(\alpha), \sigma)$ 함수 값을 가져온다. 도 4에서 3-5번 문장은 $GSA[k]$ 가 나타내는 접미사 α 가 F 에 속할 경우이다. GSA 상의 접미사 α 가 여러 개 존재할 경우 7-8번 문장은 대표접미사 $R(\alpha)$ 에 대해서 처음 한번만 수행된다.

[0064] 보조정리 2.

[0065] calLink 알고리즘은 $O(\|F\|)$ 시간에 올바른 link함수를 계산한다.

[0066] <증명>

[0067] $link(v_\lambda, \sigma)$ 는 calLink 알고리즘(도 4에서 문장 10-14)을 통해 모든 $\sigma \in \Sigma$ 에 대해 link함수 값을 가진다. 왜냐하면, " F 에는 Σ 에 속한 각 문자 σ 에 대해 σ^k ($k \geq 2$)인 형태의 문자열이 존재한다."는 가정에 의해 GSA 상

에 접미사가 σ 인 문자열은 Σ 내의 모든 문자에 대해 존재하기 때문이다. 다음으로, $i \geq 1$ 인 v_i 에 대해 $\text{link}(v_i, \sigma)$ 는 다음 두 가지 경우가 존재한다.

1. $\sigma \text{str}(v_i) \in S \cup F$ 인 경우
2. $\sigma \text{str}(v_i) \notin S \cup F$ 인 경우

[0068]

[0069]

1의 경우 calLink 알고리즘인 도 4에서 10-14번 문장에 의해 $\text{link}(v_i, \sigma)$ 이 올바르게 구해짐이 link함수의 정의에 의해 명백하다.

[0070]

2의 경우를 고려해 보자. 앞으로, 최장진접두사, 최장접두사(longest prefix), 접두사라 함은 $S \cup F$ 에 속한 문자열을 대상으로 한다. $\text{str}(v_i)$ 를 β 라 했을 때, npx의 정의에 의해, $\text{GSA}[\text{npx}[\text{R}(\beta)]]$ 가 나타내는 접미사 α 는 β 의 최장진접두사이다. $\text{link}(\text{ver}(\alpha), \sigma) = \text{ver}(\sigma \alpha')$ 라 하자. link함수의 정의에 의해 $\sigma \alpha$ 의 최장접두사는 $\sigma \alpha'$ 이다. 귀류법을 이용하여 $\sigma \alpha'$ 가 $\sigma \beta$ 의 최장접두사임을 증명한다. $|\beta'| > |\alpha'|$ 인 $\sigma \beta'$ 가 $\sigma \beta$ 의 접두사라 가정하자. 만약, $|\beta'| \leq |\alpha'|$ 라면 $\sigma \beta'$ 은 $\sigma \alpha$ 의 접두사이기도 하다. 이는 $\sigma \alpha$ 의 최장접두사가 $\sigma \alpha'$ 라는 사실에 모순된다. 또한, 만약 $|\beta'| > |\alpha'|$ 라면 β 의 최장진접두사가 α 라는 사실에 모순된다. 따라서 $|\beta'| > |\alpha'|$ 인 β' 은 존재하지 않는다.

[0071]

calLink 알고리즘의 수행시간은 8번 문장의 총 수행시간에 비례한다. 전체루프는 $\|F'\|$ 번 수행되며 8번 문장은 매 루프마다 $|\Sigma|$ 번 수행된다. 따라서, calLink 알고리즘의 총 수행시간은 $O(|\Sigma| \times \|F'\|)$ 이며, $\|F'\| = \|F\| + |F|$ 이므로 $O(|\Sigma| \times \|F\|)$ 와 같다. 그러므로, calLink 알고리즘은 상수 크기 문자 집합에 대해 $O(\|F\|)$ 시간이 걸린다.

[0072]

calLink 알고리즘을 이용하여 link함수를 계산 한 후, 접미사 그래프 Γ_S 내의 모든 정점 v_i 에 대해 $\text{link}(v_i, \sigma) = v_j$ 일 때, v_j 로부터 v_i 로 향하는 간선을 연결시켜 Γ_S 를 완성할 수 있다. 하기 표 2는 $F = \{\text{aaa}, \text{bab}, \text{aab}, \text{bbb}\}$ 에 대한 link함수이며, 이때의 접미사 그래프 Γ_S 는 도 1에 도시한 것과 같다.

표 2

$\sigma \backslash v_i$	v_λ	v_1	v_2	v_3	v_4	v_5
a	v_1	v_2	null	null	v_3	v_3
b	v_4	v_4	v_4	null	v_5	null

[0073]

[0074]

따라서, 상기한 과정에 의해 생성된 접미사 그래프 Γ_S 를 이용한 LCNSS 문제는 접미사배열을 이용함으로써 $O(\|F\|)$ 시간에 해결할 수 있다. 상기한 바와 같이, 일반화접미사배열을 이용하여 Γ_S 를 $O(\|F\|)$ 시간에 생성할 수 있음을 알 수 있다. 접미사 그래프 Γ_S 에 사이클이 존재하는지 여부는 깊이우선탐색(depth first search)을 통해 $O(|V| + |E|)$ 시간에 알 수 있다. 이 과정은 Γ_S 의 정점의 수가 $O(\|F\|)$ 이고 간선의 수는 $O(|\Sigma| \times \|F\|)$ 이므로 상수 크기 문자 집합에 대해 $O(\|F\|)$ 의 수행시간이 걸린다. 또한, 접미사 그래프 Γ_S 상의 최장경로는 깊이우선탐색을 이용한 위상정렬(topological sort)을 통해 $O(\|F\|)$ 시간에 찾을 수 있다. 최장경로가 주어졌을 경우 LCNSS는 보조정리 1에 의해 $O(\|F\|)$ 시간에 구할 수 있음을 알 수 있다.

[0075]

표 3은 PG(접두사 그래프 모델), SGA(접미사배열을 이용한 접미사 그래프 모델), SGT(접미사트리를 이용한 접미사 그래프 모델) 간의 그래프 생성 시간을 나타낸 실험 결과이다. 실험환경은 Intel Pentium4 3.0GHz CPU, 2GB RAM, Windows XP OS를 사용하였으며, 실험은 접두사 그래프 모델, 접미사배열을 이용한 접미사 그래프 모델, 접미사트리를 이용한 접미사 그래프 모델 각각의 그래프 생성 시간을 비교하고, F 에 속한 문자열의 길이를 10으로 고정하며, $\|F\|$ 의 크기를 10만부터 50만까지 10만 단위로 변화시켜가면서 수행한다. 문자열 집합 Σ 의 크기는 4, 10, 26 세가지 경우에 대해서 측정한 것이다.

표 3

F \ Σ	Σ = 4			Σ = 10			Σ = 26		
	PG	SGA	SGT	PG	SGA	SGT	PG	SGA	SGA
10	0.32	0.43	1.00	0.42	0.54	1.11	0.48	0.63	1.16
20	0.60	0.87	1.94	0.81	1.21	2.25	0.93	1.29	2.29
30	0.86	1.33	2.89	1.20	1.83	3.40	1.38	2.03	3.40
40	1.11	1.76	3.74	1.62	2.37	4.52	1.80	2.79	4.61
50	1.37	2.20	4.69	1.95	3.12	5.54	2.30	3.70	5.75

[0076]

[0077]

표 3과 같이, 실험 결과 Σ 의 크기가 커질수록 실행시간이 오래 걸리는 것을 알 수 있다. Σ 의 크기가 커질수록 실행 시간이 오래 걸리는 이유는 두 가지 그래프 모델 모두 각 정점의 진출간선의 개수가 Σ 의 크기에 비례하기 때문이다. 본 발명에서 제시한 알고리즘은 접두사 그래프 모델의 생성시간보다 평균 1.5배 느리게 나타났으며, 접미사 트리를 이용한 경우보다는 평균 1.7배에서 2.1배 빠른 것으로 나타났다. 접두사 그래프 모델이 가장 빠른 이유는 다음과 같다. 접미사 그래프 모델의 경우, 접미사 그래프를 생성하기 위해서 중간 자료구조인 접미사배열이나 접미사트리를 생성한다. 하지만, 접두사 그래프모델의 경우, 중간 자료구조를 거치지 않고 직접 접두사 그래프를 생성하기 때문이다. 접미사배열을 이용한 알고리즘이 접미사트리를 이용한 알고리즘보다 빠른 이유는 중간 자료구조인 접미사배열이 접미사트리보다 생성시간이 빠르기 때문이다.

[0078]

이와 같이, 본 발명의 실시예에 따르면, 접미사배열을 이용하여 접미사 그래프 Γ_s 를 선형시간에 생성하는 알고리즘을 제공한다. 접미사배열과 접미사트리는 모두 선형시간에 생성이 가능하지만 접미사배열을 이용하여 Γ_s 그래프를 생성할 경우, 보다 빠르게 접미사 그래프 Γ_s 를 생성할 수 있으며, 더욱이 접미사배열을 이용하여 생성된 접미사 그래프를 적용함으로써 최장공통비상위문자열을 보다 빠르고 효율적으로 찾을 수 있다.

[0079]

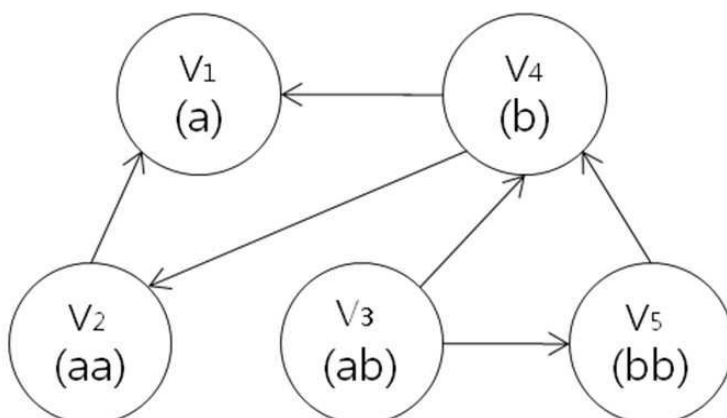
이상과 같이 본 발명은 비록 한정된 실시예와 도면에 의해 설명되었으나, 본 발명은 상기의 실시예에 한정되는 것은 아니며, 본 발명이 속하는 분야에서 통상의 지식을 가진 자라면 이러한 기재로부터 다양한 수정 및 변형이 가능하다.

[0080]

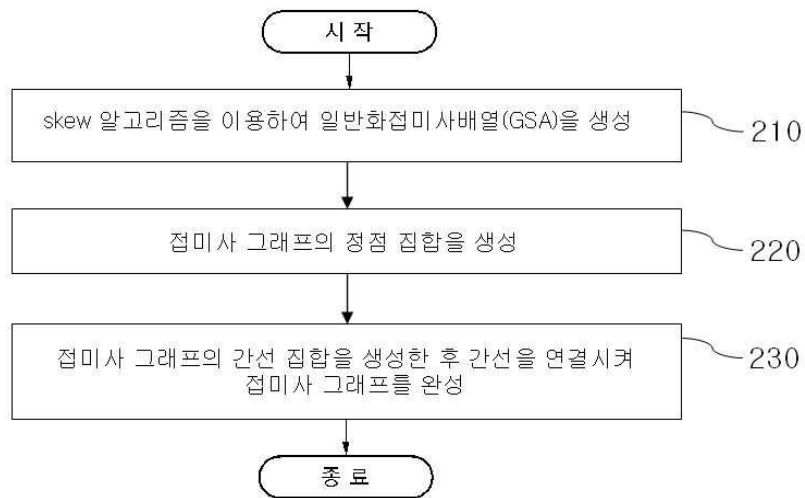
그러므로, 본 발명의 범위는 설명된 실시예에 국한되어 정해져서는 아니 되며, 후술하는 특허청구범위뿐 아니라 이 특허청구범위와 균등한 것들에 의해 정해져야 한다.

도면

도면1



도면2



도면3

```

Algorithm GetNPX
1   top := 1
2   ST[top] := 1
3   for i := 2 to ||F'|| do
4       if ( (Len[i - 1] = Len[i])
5           and (Len[i - 1] = LCP[i]) ) then
6           continue
7       fi
8       while (Len[ST[top]] > LCP[i]) do
9           top := top - 1
10          od
11          npx[i] := ST[top]
12          top := top + 1
13          ST[top] := i
14      od
  
```

도면4

Algorithm callLink

```

1  for i := 1 to ||F'|| do
2    < x, y >:= GSA[i]
3    if ( y = 1 ) then
4      continue
5    fi
6    if ( (i ≠ 1) and ((Len[i - 1] ≠ Len[i])
      or (Len[i - 1] ≠ LCP[i])) ) then
7      < x', y' >:= GSA[npX[i]]
8      ∀σ ∈ Σ[ link(δ(x, y), σ) = link(δ(x', y'), σ) ]
9    fi
10   if y - 1 = 1 then
11     link(δ(x, y), char(x, y - 1)) = null
12   else
13     link(δ(x, y), char(x, y - 1)) = δ(x, y - 1)
14   fi
15 od

```