확장된 단어 서픽스 트리에서의 완전매칭 알고리즘

박준영° 정원형 김삼묘
경북대학교 컴퓨터공학과 살림계산 공학연구실
{ws104, state}@comeng.ce.knu.ac.kr
kims@bh.knu.ac.kr

Exact Matching Algorithm on Expanded Word Suffix Tree

Joon-Young Park° Won-Hyong Chung Sam-Myo Kim
Kyungpook Uni. Experimenatal Computational Research Lab

요 약

DNA 염기 서열을 분석하는데 효율적으로 쓸 수 있는 자료구조서 서픽스 트리(Suffix Tree)가 제시되었다. 그러나 매우 큰 유전자 서열에 대한 서픽스 트리는 대용량의 메모리 공간을 필요로 한다. 따라서 메모리 공간의 절약을 위해서 단어 서픽스 트리를 이용하는 방법이 제안되었다. 단어 서픽스 트리는 이러한 장점에도 불구하고 단어에 의미를 두고 만든 트리 구조이기 때문에 완전 매칭 문제를 해결하기 위한 정보가 부족해서 제한적 완전 매칭 알고리즘이 제시되었다. 제한적 완전 매칭 알고리즘에서는 찾으려는 패턴이 어떤 단어의 부-문자열에 위치하거나, 두 단어 이상에 걸쳐 나오면 찾지 못하는 문제가 발생한다. 본 논문에서는 단어 서픽스 트리의 완전 매칭 문제를 해결하기 위해 각 단어들의 서픽스에 대한 정보로 구성된 Generalized 서픽스 트리를 사용하여 확장된 단어 서픽스 트리를 제시하고, 완전 매칭 알고리즘을 제안한다.

1. 서 론

DNA염기 서열을 효과적으로 분석 할 수 있고 응용 범위가 넓은 자료구조인 서픽스 트리(Suffix Tree)는 계산 생물학(Computational Biology)에 널리 이용되고 있다[1,2,3]. 그러나 서픽스 트리의 우수한 특성에도 불구하고 대용량의 DNA 데이터를 서픽스 트리로 구현하고 이용하는데는 문제가 있다. 서픽스 트리가 문자열의 크기 n에 대하여 O(n)의 메모리 공간을 필요로 하지만, 여기에 내포된 상수는 최소한 10 이상이므로[4], 문자열의 크기가 매우 큰 유전자 서열에 대한 서픽스 트리를 만들려면 대용량의 메모리가 필요하다. 이런 문제를 해결하기 위해 DNA 서열에 대하여 단어 서픽스 트리(Word Suffix Tree)를 이용하는 방법이 제시되었다[5]. 서픽스 트리는 O(n)의 메모리 공간을 필요로 하지만 단어 서픽스 트리는 전체 문자열의 크기 n에 대하여 분리자에 의해 나누어진 단어들의 개수 m에 대한 공간, O(m)을 필요로 한다. 이러한 메모리 공간의 절약이라는 장점을 가지지만 단어 서픽스 트리는 완전 매칭 문제에서 몇 가지 문제점이 나타난다. 단어 서픽스 트리는 나누어진 단어에 의미를 두고 만들어진 자료구조이기 때문에 어떤 단어가 다른 단어에 포함되어 있거나 다른 두 단어 이상에 걸쳐서 나오는 단어는 완전히 찾지 못하거나 전부 찾지 못하는 문제가 생긴다.

본 논문에서는 유전자 서열을 단어 서픽스 트리로 만들고 난 후 이를 이용한 제한적 완전 매칭 알고리즘과 여기에서 파생되는 문제를 알아보고 이를 해결하기 위한 확장된 단어 서픽스 트리와 완전 매칭 알고리즘을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 단어 서픽스 트리와 제한적 완전 매칭 알고리즘[6]을 소개하고 그 문제점을 제시한다. 3장에서는 Generallized 서픽스 트리[3]를 사용한 확장된 단어 서픽스 트리와 이를 이용한 완전 매칭 알고리즘을 제시한다. 마지막 4장에서는 결론 및 향후 과제를 제시한다.

2. 단어 서픽스 트리와 제한적 완전 매칭 알고리즘

2.1 단어 서픽스 트리

주어진 문자열의 크기가 매우 커서 서픽스 트리를 주기억장치에 둘 수 없을 때는 서픽스 트리를 생성할 때 뿐 아니라, 이를 이용할 때에도 디스크 입출력시간 지연 때문에 심각한 문제에 직면한다. 이를 해결하기 위한 한가지 방편으로 단어 서픽스 트리(word Suffix Tree)가 제안되었다. 단어 서픽스 트리는 크게 다음 4단계로 만들어 진다: (a) 주어진 문자열을 특정 분리자(delimiter)를 이용 일련의 단어$W_1W_2...W_m$로 분리한 후, (b) 단어 W_i 마다 새로운 식별자(identifier) I_i(일반적으로 정수)를 부여하여 원래의 문자열을 단축된 문자열 $I_1I_2...I_m$로 변환하고, (c) 이 단축된 문자열에 대한 서픽스 트리를 만든 다음 ,(d) 최종적으로 이 서픽스 트리상의 각 식별자 I_i를 해당 단어 W_i로 복원하여 단어 서픽스 트리를 만든다. 단어 서픽스 트리생성과정에서 식별자 선택과 복원과정을 효율적으로 하기 위해서 워드 트라이를 이용한다.

2.2 제한적 완전 매칭 알고리즘과 문제점

단어 서픽스 트리는 모든 문자열을 추가하여 트리를 구성하는 것이 아니라, 분리자로 나누어진 모든 단어들을 이용하여

트리를 구성한다. 그러므로 완전 매칭 문제에 있어서 모든 부-문자열(sub-string)에 대한 매칭은 할 수 없고 모든 부-단어(sub-word)들에 대한 완전 매칭만 가능하다. 그래서 이 문제를 해결하기 위한 제한적 알고리즘[6]이 다음에 나와 있다. 여기서 제한이라 함은 패턴과 텍스트의 분리자가 동일해야 하며, 패턴의 크기는 한 개 이상의 단어로 구성되어 있어야 한다.

2.2.1 주어진 패턴 P의 전처리 단계

1.각각의 문자열로 구성된 패턴 P(1..n)를 단어로 구분:
P= $w_1 w_2 \ldots w_m$
변환된 패턴 P(1..m)를 $\alpha\ \beta\ \gamma$로 대치한다.
(단, $|\alpha| = 1$, $|\beta| \geq 0$, $|\gamma| = 1$ or 0)
2.패턴의 첫 번째 단어를 α로 대치 (i=1)
3.두 번째 단어부터 마지막 앞까지의 단어를 β로 대치 (i=2..m-1)
4..마지막 단어는 γ로 대치 (i=m)
(단, i는 단어열의 인덱스를 가리킨다.)

2.2.2 변환된 패턴을 단어 서픽스 트리에서 찾는 단계

패턴 크기가 1인 경우 (α로 대치한다.)

α가 텍스트로 구성된 단어 트라이에 존재하면 패턴 α는 단어 서픽스 트리에 반드시 존재한다. 그 외의 경우는 제한적 성질에 의해서 서픽스 트리에 존재하는지 알 수 없다.

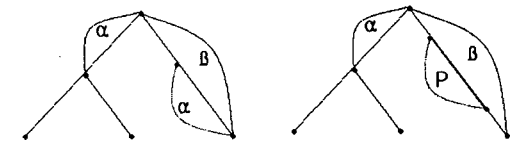
패턴 크기가 2인 경우

1. α와 β가 모두 텍스트로 구성된 단어 트라이에 존재하면, 패턴 $\alpha\beta$는 단어 서픽스 트리에 반드시 존재한다.
2. α, β 둘 중 하나의 단어만 텍스트로 구성된 단어 트라이에 존재하는 경우는 다음 두가지 경우를 고려하여야 한다.
(1)텍스트로 구성된 단어 트라이에 α만 존재하고 β가 존재하지 않는 경우는 다음과 같은 방법으로 완전매칭을 한다. 우선 텍스트로 구성된 단어 서픽스 트리에서 α를 찾는다. 그 후, α를 전치열로 하는 자식들에서 β를 전치열로 하는 단어가 존재하면, 패턴 $\alpha\beta$는 단어 서픽스 트리에 반드시 존재한다(단, β는 분리자를 가지지 않는 단어이다).
(2)텍스트로 구성된 단어 트라이에서 α는 존재하지 않고 β만 존재하는 경우는 다음과 같은 방법으로 완전매칭을 한다. 우선 텍스트로 구성된 단어 서픽스 트리에서 β를 찾는다. 그 후, β를 후치열로 가지면서 α를 후치열로 하는 β-1 존재하면, 패턴 $\alpha\beta$는 단어 서픽스 트리에 반드시 존재한다.
3.그 외의 경우는 패턴이 단어 서픽스 트리에 존재하지 않는다.

패턴 크기가 3 이상인 경우 (α로 대치한다.)

1.텍스트로 구성된 단어 트라이에 α,β,γ가 모두 존재한다면, 패턴 $\alpha\beta\gamma$는 단어 서픽스 트리에 반드시 존재한다.
2.텍스트로 구성된 단어 트라이에 α와 β는 존재하나 γ는 존재하지 않는 경우는 다음과 같은 방법으로 완전매칭을 한다.(단 γ는 분리자를 가지지 않는 단어이다.) 우선 단어 서픽스 트리에서 $\alpha\beta$를 찾는다. 그 후, $\alpha\beta$를 전치열로 하는 자식들(X,...Y)에서 γ를 전치열로 하는 단어가 존재하며, 패턴 $\alpha\beta\gamma$는 단어 서픽스 트리에 반드시 존재한다.
3.텍스트로 구성된 단어 트라이에 α는 존재하지 않고 β와 γ는 트라이에 존재하는 경우는 다음과 같은 방법으로 완전매칭을 한다. 우선 단어 서픽스 트리에서 $\beta\gamma$를 찾는다. 그 후, $\beta\gamma$를 후치열로 가지면서 α를 후치열로 하는 β-1이 존재하면, 패턴 $\alpha\beta\gamma$는 단어 서픽스 트리에 반드시 존재한다
4.그 외의 경우는 패턴이 단어 서픽스 트리에 존재하지 않는다.

패턴의 크기가 1인 경우의 문제점은 α가 단어 트라이에 존재하면 단어 서픽스 트리에서 반드시 존재 하기는 하지만 텍스트에서 모든 α를 찾지 못하는 문제가 발생한다. α가 단어이기는 하나 만약 다른 단어들의 부-문자열에 존재를 한다면 α와 동일한 부-문자열을 찾지 못한다. 그림 2-1(1)에서 β에 속한 α는 찾지 못한다. 또 다른 문제점은 단어 트라이에 나타나지 않는 패턴은 텍스트에 있음에도 불구하고 찾지 못하는 것이다. 그림 2-1(2)에서 보면 패턴P가 있음에도 불구하고 β속에 숨겨져서 찾지 못한다.



(1) β에 α가 포함되어 있을 경우 (2) β에 P가 포함되어 있을 경우

그림 2-1. 패턴의 크기가 1인 경우의 문제점

패턴의 크기가 2이상인 경우의 문제점은 패턴이 2단어 이상에 걸쳐서 나올 때 찾을 수도 있고 못 찾을 수도 있다. 예를 들어 그림2-2는 텍스트에서 atgt를 패턴으로 하여 찾으면 처음에 나오는 atgt(①)는 단어 서픽스 트리에서 찾을 수 있다. 그러나 at가 다른 단어(ggat)의 후치열로 존재하는 경우 만약 그 단어(ggat) 다음에 gt가 있다면 여기에도 atgt(②)가 존재 하지만 단어 서픽스 트리에서는 찾지 못한다.

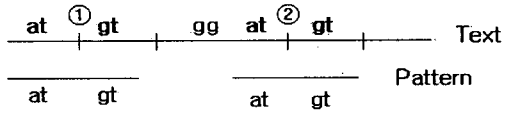


그림 2-2. 패턴이 두 단어에 걸쳐 나오는 경우

3. Generalized 서픽스 트리를 사용한 확장된 단어 서픽스 트리와 완전 매칭 알고리즘

단어 서픽스 트리는 단어에만 의미를 두고 있기 때문에 위의 2장에서 제시한 문제점을 해결할 수가 없다. 이 문제를 해결하기 위해서는 각 단어들의 후치열들에 대한 정보를 단어 서픽스 트리에 추가해야만 한다. 따라서 단어 트라이에 대한 정보는 그대로 유지하면 후치열들에 대한 정보를 추가하기 위해서는 단어 트라이 대신에 다른 자료 구조가 요구되는 데 이를 만족시키는 것이 generalized 서픽스 트리다.

3.1 Generalized 서픽스 트리의 정의

어떤 스트링들의 집합{S_1,S_2, ..., S_2}을 하나의 서픽스 트리로 만들어 놓은 것을 Generalized 서픽스 트리라 한다. 또한 모든 스트링 길이의 합에 비례하는 시간 안에 Generalized 서픽스 트리를 만들어 낼 수 있다[3].

3.2 Generalized 서픽스 트리를 사용한 확장된 단어 서픽스 트리

여기에서는 단어 서픽스 트리에서 완전 매칭 문제를 해결하기 위한 확장된 단어 서픽스 트리를 제안한다. 이 트리의 구조는 Andersson이 제시한 단어 서픽스 트리와 거의 유사하지만 번호기반 서픽스 트리의 확장에서 Generalized 서픽스 트리를 사용하는 부분이 추가 된 것이다. 확장된 단어 서픽스 트리는 크게 다음 5단계로 만들어진다: (a)주어진 문자열을 특정 분리자(delimiter)를 이용 일련의 단어 $W_1W_2...W_m$로 분리한 후, (b) 단어 W_i 마다 새로운 식별자(identifier) I_i(일반적으로 정수)를 부여하여 원래의 문자열을 단축된 문자열 $I_1I_2...I_m$로 변환하고, (c)이 단축된 문자열에 대한 서픽스 트리를 만든 다음, (d)서로 다른 단어 $W_{m'}$를 사용하여 Generalized 서픽스 트리를 만들고, (e)최종적으로 이 서픽스 트리 상의 각 식별자 I_i를 해당 단어 W_i로 복원하여 확장된 단어 서픽스 트리를 만든다. 그림 3-1은 DNA 스트링(CAGTAGATAGAATCTTCAGTCTTTAGAT CAGTCAG$)으로부터 생성된 확장된 단어 서픽스 트리를 보여준다.

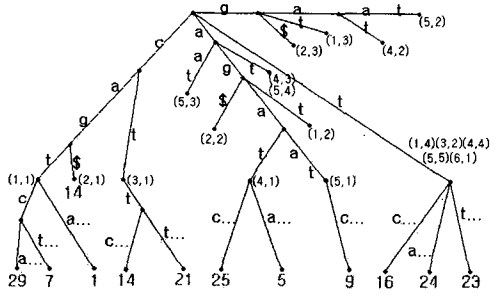


그림 3-1. Generalized 서픽스 트리를 사용하여 확장한 단어 서픽스 트리

3.3 확장된 단어 서픽스 트리에서의 완전 매칭 알고리즘

3.3.1 주어진 패턴 P의 전처리 단계

1.각각의 문자열로 구성된 패턴 P(1..n)를 단어로 구분:
P= $w_1 w_2 \ldots w_m$
변환된 패턴 P(1..m)를 α β로 대치한다.
(단, $|\alpha| = 1$, $|\beta| \geq 0$)
2.패턴의 첫 번째 단어를 α로 대치 (i=1)
3.두 번째 단어부터 마지막까지의 단어를 β로 대치 (i=2..m)
(단, i는 단어열의 인덱스를 가리킨다.)

3.3.2 변환된 패턴을 Generalized 서픽스 트리를 사용한 단어 서픽스 트리에서 찾는 단계

패턴의 크기가 1인 경우(α로 대치한다.)

α가 텍스트로 구성된 단어 트라이에 존재할 수도 있고(α가 단어로 존재)없을 수도 있다(α 가 어떤 단어의 부 문자열(substring)이다). 그러나 α가 Generalized 서픽스 트리에 없다면 α는 텍스트에 존재하지 않는 패턴이다.
기본적인 탐색 방법은 일단 α를 찾고 난 후 노드의 정보를 보고 α가 다른 단어의 부-문자열에 있는지 찾는다. 여기서 노드의 정보에 (i,k) 라는 것은 i 번호를 가지는 단어에서 k 번째 위치에서의 후치열(suffix)을 가리키는 것으로 i 번호를 가지는 단어의 모든 문자열을 나타내는 노드는 (i,1)을 찾으면 된다. 여기서는 패턴이 분리자를 가지는 것과 가지지 않는 것으로 분리해서 기술한다.
1.패턴(α)이 분리자를 가지는 경우 : α를 Generalized 서픽스 트리를 사용한 단어 서픽스 트리에서 찾는다. α가 없으면 텍스트에서 α는 존재하지 않는 것이다. α가 찾아지면 α가 다른 단어들의 후치열에 존재하는지를 찾기 위해 α를 찾은 다음 그 노드의 정보에서 α가 다른 단어의 후치열에 존재한다는 정보(예를 들어 그림 3-2(1)에서 (j,3)를 보고(j,1)노드를 찾을 수 있다.)를 보고 찾아가면 모든 α를 텍스트에서 찾을 수 있다.
2.패턴(α)이 분리자를 가지지 않는 경우 : α는 단어가 아닌 단어들의 부-문자열로 나올 수 있다. α를 Generalized 서픽스 트리를 사용한 단어 서픽스 트리에서 찾는다. α가 없으면 텍스트에서 α는 존재하지 않는 것이다. α가 찾아지면 α 및의 모든 잎 노드의 정보에서 α가 다른 단어의 부-문자열에 존재한다는 정보(예를 들어 그림 3-2(2)에서 (j,2)를 보고(j,1)노드를 찾고 (k,2)를 보고 (k,1)노드를 찾을 수 있다.)를 보고 찾아가면 모든 α를 텍스트에서 찾을 수 있다.

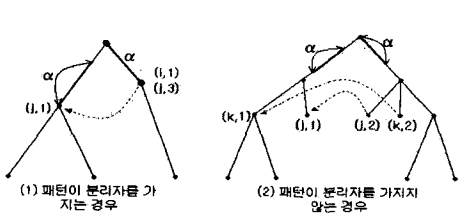


그림 3-2. 패턴 크기가 1인 경우의 완전 매칭

패턴의 크기가 2이상인 경우(α β 로 대치한다.)

패턴이 분리자에 의해 두 부분이상으로 나누어 졌기 때문에 α는 분리자를 마지막 문자로 항상 가지게 된다. 따라서 α가 단어 트라이에 존재하는 단 어 일수도 있고 동시에 다른 단어의 후치열인 경우와 α가 단어가 아닌 단어들의 후치열로만 존재하는 두 가지 경우가 있을 수 있다. 따라서 이 두 경우를 분리해서 기술한다.
1.α가 단어이면서 동시에 다른 단어의 후치열인 경우 : α를 generalized 서픽스 트리를 사용한 단어 서픽스 트리에서 찾는다. α가 없으면 텍스트에서 패턴은 존재하지 않는 것이다. α가 찾아지면 그 노드 및에서 β를 찾아 패턴이 존재하는지를 검색한다. 또한 α가 다른 단어들의 후치열에 존재하는지를 찾기 위해 그 노드의 정보에서 α가 다른 단어의 후치열에 존재한다는 정보(예를 들어 그림3-3(1)에서 (j,3)를 보고(j,1)노드를 찾을 수 있다.)를 보고 그 단어(k)의 자식들 중에 β가 있는지를 검색하면 된다.
2.α가 단어가 아닌 단어들의 후치열인 경우 : α를 generalized 서픽스 트리를 사용한 단어 서픽스 트리에서 찾는다. α가 없으면 텍스트에서 패턴은 존재하지 않는 것이다. α가 찾아지면 α가 다른 단어들의 후치열에 존재하는지를 찾기 위해 α를 찾은 다음 그 노드의 정 보에서 α가 다른 단어의 후치열에 존재한다는 정보(예를 들어 그림 3-3(2)에서 (j,2)를 보고 (j,1)노드를 찾고, (k,3)을 보고 (k,1)노드를 찾을 수 있다.)를 보고 찾아가면 모든 α를 찾을 수 있다. 따라서 검색된 노드((j,1),(k,1)) 다음의 자식들에서 β가 잇는지를 검색하면 된다.

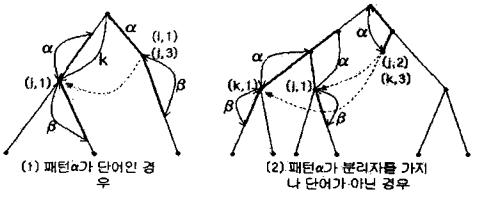


그림 3-3. 패턴의 크기가 2 이상인 경우의 완전 매칭

4. 결론

본 논문에서는 Generalized 서픽스 트리를 사용하여 단어 서픽스 트리를 확장해 완전 매칭 문제를 해결했다. 나누어진 단어(m)들의 크기가 클수록 전통적 서픽스 트리에 비해 단어 서픽스 트리는 메모리 효율이 좋아진다. 확장된 단어 서픽스 트리는 단어 서픽스 트리보다 거의 Generalized 서픽스 트리의 노드 수많 큼 노드수가 증가를 하지만 그 수가 서픽스 트리의 노드 수보다는 명확히 줄어든다.

지금까지는 분리자를 어떤 것을 사용했는지는 고려를 하지 않았다. 따라서 유전자 서열 정보인 A, C, G, T 각각을 분리자로 했을 때의 메모리 공간의 효율성 비교와 여러 가지 유전 정보(예를 들어 시작 코돈(Codon)인 'ATG)를 분리자로 두었을때의 메모리 효율성에 관한 연구가 필요하다.

5. 참고문헌

[1]M. V. Olson, "A time to sequence," *Science, vol. 270,* pp.394-396, 1995.
[2]G. von Heijne, *Sequence Analysis in Molecular Biology: Treasure Trove or Trivial Pursuit*, Academic Press, New York, 1987.
[3]D.Gusfield, *Algorithms on Strings, Trees, and Sequences,* Cambridge University Press, pp.8-10,93-124,116-117 1997.
[4]Juha Karkkainen and Esko Ukkonen, "Sparse Suffix Tree", *COCOON* 1996, pp. 219-233.
[5]Arne Andersson, "Suffix Trees on Words," Journal of Algorithmica, 1999.
[6]Jin young Kim "단어 서픽스 트리 구현 및 응용" 경북대학교 컴퓨터 공학과 석사 학위 논문.