최장공통비상위문자열을 찾는 새로운 알고리즘
(A New Algorithm for the Longest Common Non-superstring)

최 시 원 [†] (Siwon Choi)　이 도 경 [††] (Dokyoung Lee)
김 동 규 [†††] (Dong Kyue Kim)　나 중 채 [††††] (Joong Chae Na)
심 정 섭 [†††††] (Jeong Seop Sim)

요 약 문자열 불포함 문제에 대한 연구는 최근 들어 여러 분야에서 활발히 진행되어 왔다. 문자열 집합 F가 주어질 때, F 내의 어떤 문자열도 포함하지 않는 문자열을 F에 대한 공통비상위문자열이라 하고 공통비상위문자열 중에서 가장 긴 유한길이의 문자열을 최장공통비상위문자열이라 한다. 본 논문에서는 공통비상위문자열과 관련된 연구 결과들을 제시한다. 먼저 기존의 공통비상위문자열에 대한 접미사 그래프 모델과 달리 접두사를 이용하여 직관적인 그래프 모델링이 가능함을 증명한다. 다음으로, 상수 크기의 알파벳에 대해 정의된 문자열 집합 F의 모든 문자열들의 길이의 합을 N라 할 때 $O(N)$시간에 접두사 그래프를 생성하고 이를 이용하여 최장공통비상위문자열을 찾는 알고리즘을 제시한다.

키워드 : 문자열 불포함, 최장공통비상위문자열

Abstract Recently, the string non-inclusion related problems have been studied vigorously. Given a set of strings F over a constant size alphabet, consider a string x such that x does not include any string in F as a substring. We call x a Common Non-SuperString(*CNSS* for short) of F. Among the *CNSS*'s of F, the longest one with finite length is called the Longest Common Non-SuperString(*LCNSS* for short) of F.

In this paper, we first propose a new graph model using prefixes of F. Next, we suggest an $O(N)$-time algorithm for finding the *LCNSS* of F, where N is the sum of the lengths of all the strings in F.

Key words : string non-inclusion, longest common non-superstring

· 이 논문은 2008년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2008-331-D00478)
· 이 논문은 2007년도 정부(교육과학기술부)의 재원으로 국제과학기술협력재단의 지원을 받아 수행된 연구임(No. K20717000007O7B010000710)
· 이 논문은 2008 한국컴퓨터종합학술대회에서 '접두사 그래프 모델을 이용한 최장공통비상위문자열 찾기'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 인하대학교 컴퓨터정보공학부
siwonred@gmail.com
†† 비 회 원 : 서울대학교 컴퓨터공학부
domeng@naver.com
††† 종신회원 : 한양대학교 전자통신컴퓨터공학부 교수
dqkim@hanyang.ac.kr
†††† 정 회 원 : 세종대학교 컴퓨터공학과 교수
jcna@sejong.ac.kr
††††† 종신회원 : 인하대학교 컴퓨터정보공학부 교수
jssim@inha.ac.kr

논문접수 : 2008년 8월 27일
심사완료 : 2008년 10월 20일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제15권 제1호(2009.1)

1. 서 론

문자열 포함 문제는 많은 분야에서 연구되고 있다. 대표적으로 주어진 문자열들의 공통 부분서열 중 가장 긴 서열(Longest Common Subsequence)을 찾는 문제와 공통 상위서열 중 가장 짧은 서열(Shortest Common Supersequence)을 찾는 문제 등이 연구되어 왔다[1-4]. 한편, 주어진 문자열을 포함하지 않는 문자열 불포함 문제가 [4]에서 소개되어 압축 알고리즘, 분자생물학 등 다양한 분야에서 필요성이 대두되어 진행되고 있다.

문자열 집합 F가 입력으로 주어졌을 때, F의 모든 문자열들을 포함하지 않는 문자열을 공통비상위문자열(Commom Non-SuperString, 이하 CNSS라 부름)이라 한다. CNSS중 가장 긴 문자열을 F의 최장공통비상위문자열(Longest Common Non-SuperString, 이하 LCNSS라 부름)이라 한다. 한편, CNSS의 집합이 유한 집합이 아닌 경우 LCNSS는 유한 길이의 문자열로 구해지지 않는다. 이때 LCNSS는 존재하지 않는다고 한다.

예 1. 알파벳 $\Sigma=\{a,b\}$에서 $F=\{aaa,aba,bba,bbb\}$일 때, F의 CNSS 집합은 $\{\lambda,a,aa,aab,aabb,ab,abb,b,ba,baa,baab,baabb,bab,babb,bb\}$이며, 이들 중 $baabb$가 LCNSS이다.

본 논문에서는 [5]와 [6]의 접미사를 이용한 그래프 모델과 달리 접두사를 이용한 그래프 모델로 CNSS를 모델링 할 수 있는 것을 증명하고, 상수 크기의 알파벳에서 $O(N)$의 시간에 접두사 그래프를 생성할 수 있는 알고리즘을 제시한다. 또한 이렇게 생성된 접두사 그래프를 바탕으로 LCNSS를 찾는 알고리즘을 제시한다.

본 논문은 다음과 같이 구성된다. 2장에서는 본 논문

의 알고리즘을 위한 몇 가지 용어에 대한 정의와 관련 연구들을 제시한다. 이를 바탕으로 3장에서는 접두사 그래프를 정의하고 이를 통해 CNSS를 모델링 할 수 있음을 보인다. 4장에서는 주어진 문자열 집합에 대한 접두사 그래프를 생성하고 F의 LCNSS를 찾는 문제를 해결하는 알고리즘을 제시한다. 5장에서는 결론과 향후 연구 방향을 제시한다.

2. 관련 연구

문자열이란 유한 알파벳 집합 Σ에서 0개 이상의 문자들이 연결된 형태이다. 0개 이상의 Σ의 문자들로 이루어진 모든 문자열의 집합을 Σ^*라 한다. 공백문자열은 λ로 나타낸다. 두 문자열 A,B 의 연결(concatenation)을 AB 라고 표기한다. 문자열 A의 길이를 $|A|$ 로 표기하고 A의 i번째 문자는 $A[i]$로 나타내기로 한다. 공백문자열의 길이는 $|\lambda|=0$ 이다. 문자열 A의 i번째 문자부터 j번째 문자까지의 연결을 $A[i..j]$로 나타내고 이를 A의 부분문자열(substring)이라고 한다. 예를 들어 $A=abc$일 때, $A[3]=c$ 이고, $A[1..2]=ab$ 이다. 특별히 $i>j$ 인 경우 $A[i..j]=\lambda$로 정의한다. 문자열 A가 문자열 B의 부분문자열일 때 $A \trianglelefteq B$로 표기하고 역으로 B를 A의 상위문자열(superstring)이라 한다. 예를 들면 $cde \trianglelefteq abcdef$이다. 문자열 A가 문자열 B의 부분문자열이 아닐 때 $A \ntrianglelefteq B$로 표기한다. 문자열 A에서 $i \le |A|$일 때 $A[1..i]$를 A의 접두사(prefix)라 하고, $i<|A|$일 때 $A[1..i]$를 A의 진접두사(proper prefix)라 한다. 유사하게 $i \ge 1$ 일 때 $A[i..|A|]$를 A의 접미사(suffix)라 하고, $i>1$ 일 때 $A[i..|A|]$를 A의 진접미사(proper suffix)라 한다. 편의상 공백문자열 λ는 모든 문자열의 접두사(접미사)이며 진접두사(진접미사)로 정의한다.

정의 1. 공통비상위문자열 집합

유한 알파벳 Σ에 대한 문자열 집합 $F=\{f_1,f_2,\cdots,f_n\}$가 주어졌을 때, F의 모든 CNSS 집합을 $CNSS_F$라 한다.

즉, $CNSS_F=\{A|A\in\Sigma^*$ 이고 $f_i \ntrianglelefteq A$. 단, $1 \le i \le n\}$이다.

정의 2. 최장공통비상위문자열 문제 (LCNSS 문제)

최장공통비상위문자열 문제는 주어진 문자열 집합 F에 대한 $CNSS_F$에서 유한 길이의 최장문자열 $LCNSS_F$의 존재여부를 결정하는 문제이다. 만약 $LCNSS_F$가 존재할 경우 $LCNSS_F$를 출력한다.

각 f_i는 F의 CNSS가 포함하지 않아야 할 문자열이므로 금지문자열이라 부르기로 하자. 또한 F는 공백문자열을 포함하지 않고, 다음과 같은 비포함규칙(inclusion free)을 만족하는 것으로 가정한다.

$$\forall A,B\in F \Rightarrow A \ntrianglelefteq B$$

만약 위의 조건이 만족되지 않는다면 다항시간 안에 F에서 A의 상위문자열 B를 삭제하여 위의 조건이 만족되고 동일한 $CNSS_F$가 생성되는 F를 만들 수 있다. F 의 원소 중 가장 긴 문자열의 길이를 $\lceil F \rceil$ 라 하고, 각 금지문자열 길이의 합을 N이라고 표기한다.

3. 그래프 모델링

본 논문에서는 F 내의 문자열들의 접두사들을 이용하여 $CNSS_F$를 모델링한다.

정의 3. 진접두사 집합

F의 진접두사 집합 P는 각 금지문자열의 서로 다른 진접두사들의 집합이다. 특히 $p_0=\lambda$라고 정의한다.

즉, $P=\{p|p=f_i[1..k]$, 단 $1 \le i \le n$ 이고 $k<|f_i|\}$

예 3. 예 1에 대해 $P=\{\lambda,a,aa,ab,b,bb\}$이다.

비포함규칙이 만족된다고 가정했으므로 $P\cap F=\varnothing$이고 $P\subset CNSS_F$임을 알 수 있다.

이제 유향그래프 $G_F(V,E)$를 이용하여 $CNSS_F$를 모델링한다. G_F의 정점 집합 V는 $F\cup P$의 문자열들에 대응되는 정점들로 구성된다.

즉, $F\cup P=\{f_1,\cdots,f_n, p_0,\cdots,p_m\}$에 대해

- $f_i(1 \le i \le n)$에 대응되는 정점은 \$이고,
- $p_j(0 \le j \le m)$에 대응되는 정점은 v_j이다. 이때 v_j는 유일하다. 즉, $p_s \ne p_t$ 이면 $v_s \ne v_t$이다.

$F\cup P$의 원소(문자열) s에 대응되는 정점을 $ver(s)$라 표기하자. 그리고 p_j에 대해 $p_j=str(v_j)$로 표기한다. 특별히 혼동되지 않는 한 앞으로 정점 v가 나타내는 문자열과 문자열 A의 연결 $str(v)A$를 편의상 단순히 $v\oplus A$로 표기하기로 한다.

V에서 정점 \$를 삭제한 정점 집합을 $V'(=V-\{\$\})$이라 하자. p_j와 대응되는 정점 $v_j\in V'$를 $ver'(p_j)$로 나타내고 ver'과 str함수의 정의에 의해 $ver'(str(v_j))=v_j$임을 알 수 있다. 정점 집합 V의 크기는 $|F\cup P|$에 비례하므로 $|V|\in O(N)$ 임을 알 수 있다.

간선 E의 생성을 위해 다음의 함수들을 이용한다.

정의 4. LS함수 ($LS: \Sigma^* \rightarrow P\cup F$)

문자열 $A\in\Sigma^*$가 주어졌을 때 $LS(A)$는 A의 접미사 중 $F\cup P$에 속하는 가장 긴 문자열을 나타낸다.

즉, $LS(A)=A[k..|A|]$ 이다.

(단 , $k=\min_{1 \le i}\{i|A[i..|A|]\in F\cup P\}$ 이다.)

$LS(A)$에 대응되는 정점, $ver(LS(A))$는 편의상 $LSV(A)$로 축약하여 사용하기로 한다.

정의 5. δ함수 ($\delta: V\times\Sigma\rightarrow V$)

모든 $v\in V, x\in\Sigma$에 대해 $\delta(v,x)$는 다음과 같다.

$$\delta(v,x) = \begin{cases} v = \$ \text{ 이면, } \$ \\ v \neq \$ \text{ 이면, } LSV(v \oplus x) \end{cases} \text{ 이다.}$$

$\delta(v,x)$는 $v \oplus x$의 접미사 중 $P \cup F$에 속하는 가장 긴 문자열에 해당하는 정점을 나타낸다.

정의 6. δ^*함수 ($\delta^* : V \times \Sigma^* \rightarrow V$)

모든 $v \in V$와 $A \in \Sigma^*$에 대해 $\delta^*(v,A)$는 다음과 같다.

$$\delta^*(v,A) = \begin{cases} A = \lambda \text{ 이면, } v \\ A \neq \lambda \text{ 이면, } \delta(\delta^*(v, A[1..|A|-1]), A[|A|]) \end{cases}$$

G_F의 간선 집합 E는 하나의 간선이 연결하는 두 정점 v_i, v_j와 문자 x의 쌍의 집합이며 다음과 같다.

$$< v_i, x, v_j > \in E \Leftrightarrow \delta(v_i, x) = v_j$$

각 정점마다 $|\Sigma|$의 개수의 간선이 존재하므로 $|E| \in O(|\Sigma| \times N)$이다. (그림 1 참조)

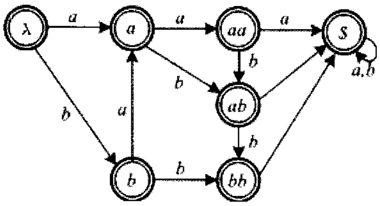

그림 1 예 1에 대한 G_F ($F = \{aaa, aba, bba, bbb\}$)

위의 정의에 의해 얻어지는 보조정리는 다음과 같다.

보조정리 3.1. $A \in \Sigma^*$, $x \in \Sigma$ 일 때 $LS(Ax) = LS(LS(A)x)$이다.

증명. $A = \lambda$인 경우 자명하다. $A \neq \lambda$인 경우 $Ax = B$라 하면, $|B| > 1$이다. 이 때, 세 개의 자연수 s,t,u가 다음을 만족한다고 하자. $B[s..|B|] = LS(Ax)$이고 $B[t..|B|] = LS(A)x$이다. 또한, $B[u..|B|] = LS(LS(A)x)$ 이다. 먼저 $t \leq s$임을 증명한다. $B[s..|B|-1]$는 A의 접미사이고, $B[s..|B|] \in P \cup F$이므로 $B[s..|B|-1] \in P$이다. 또한 $B[t..|B|-1] = LS(A)$이다. $LS(A)$는 A의 접미사 중 $P \cup F$에 속하는 최장문자열이므로 $t \leq s$이다. 이제 $u \leq s$임을 보인다. $t \leq s$이므로 $B[s..|B|]$는 $B[t..|B|]$의 접미사이고, $B[s..|B|] \in P \cup F$이다. $B[u..|B|]$는 $LS(B[t..|B|])$이므로 $B[t..|B|]$의 접미사 중 $P \cup F$에 속하는 최장문자열이다. 즉, $u \leq s$이다. 다음으로 $s \leq u$임을 보인다. $B[u..|B|]$는 B의 접미사이고 $B[u..|B|] \in P \cup F$이다. 또한 B와 s의 가정에 의해 $B[s..|B|]$는 B의 접미사 중 $P \cup F$에 속하는 최장문자열이다. 즉, $s \leq u$ 이다. 따라서 $u = s$이므로 $LS(Ax) = LS(LS(A)x)$이다. □

따름정리 3.2. $A \in CNSS_F$, $x \in \Sigma$ 일 때 $LSV(Ax) = LSV(LSV(A) \oplus x)$이다.

증명. $LSV(Ax) = ver(LS(Ax))$이다. 보조정리 3.1에 의해 $LS(Ax) = LS(LS(A)x)$이고 $A \in CNSS_F \Rightarrow LS(A) \notin F$이므로 $LS(LS(A)x) = LS(ver(LS(A)) \oplus x)$이다. 따라서 $LSV(Ax) = ver(LS(ver(LS(A)) \oplus x)) = LSV(LSV(A) \oplus x)$이다. □

보조정리 3.3. $v \in V'$, $v \oplus A \in CNSS_F$ 일 때 $\delta^*(v,A) = LSV(v \oplus A)$ 이다.

증명. $\delta^*(v, A[1..t]) = LSV(v \oplus A[1..t])$ $(t = 0, \cdots, |A|)$임을 증명한다. 귀납법으로 $t = 0$일 때 $\delta^*(v, A[1..0]) = v = LSV(v \oplus \lambda)$이고 $t = 1$일 때 $\delta^*(v, A[1..1]) = \delta(v, A[1]) = LSV(v \oplus A[1])$가 성립한다. 이제 $t = k (< |A|)$일 때 $\delta^*(v, A[1..k]) = LSV(v \oplus A[1..k])$가 성립함을 가정하자. 그러면 $v \oplus A \in CNSS_F \Rightarrow v \oplus A[1..k] \in CNSS_F$이고, $\delta^*(v, A[1..k+1])$은 다음과 같다.

$$\begin{aligned} \delta^*(v, A[1..k+1]) &= \delta(\delta^*(v, A[1..k]), A[k+1]) \\ &= \delta(LSV(v \oplus A[1..k]), A[k+1]) \quad (LSV(v \oplus A[1..k]) \neq \$) \\ &= LSV(LSV(v \oplus A[1..k]) \oplus A[k+1]) \\ &= LSV(v \oplus A[1..k]A[k+1]) \quad (\because \text{따름정리 } 3.2) \\ &= LSV(v \oplus A[1..k+1]) \text{ 이다.} \end{aligned}$$

따라서 $\delta^*(v,A) = LSV(v \oplus A)$ 이다. □

따름정리 3.4. $v \in V'$, $v \oplus A \in CNSS_F$ 일 때 $\delta^*(v,A) = \delta^*(v_0, v \oplus A)$이다.

증명. 보조정리 3.3에 의해 좌변은 $\delta^*(v,A) = LSV(v \oplus A)$이고, 우변은 $\delta^*(v_0, v \oplus A) = LSV(v \oplus A)$이다. □

정리 3.5. $A \in \Sigma^*$에서 $A \in CNSS_F \Leftrightarrow \delta^*(v_0, A) \in V'$.

증명. (a) $A \in CNSS_F \Rightarrow \delta^*(v_0, A) \in V'$: 모순법으로 이를 거짓으로 가정하면, 임의의 $A \in CNSS_F$ 대해 $\delta^*(v_0, A) = \$$이다. 이때 $\delta^*(v_0, A[1..t]) = v\ (\neq \$)$를 만족하는 가장 큰 자연수 t를 택하자. v의 다음 δ함수는 $\delta(v, A[t+1]) = \$$이고, 이때 δ함수가 $\$$이기 위해서는 $LSV(v \oplus A[t+1]) = \$$여야 한다. 따라서 $LS(v \oplus A[t+1])$은 어떤 금지문자열임을 의미한다. 하지만 $v \oplus A[t+1]$는 A의 부분문자열이고 A는 금지문자열을 포함하지 않기 때문에 이는 모순이다. (b) $\delta^*(v_0, A) \in V' \Rightarrow A \in CNSS_F$: 모순법으로 이를 거짓으로 가정하면, $\delta^*(v_0, A) \in V'$인 문자열 A는 어떤 금지문자열을 $f_i \in F$를 포함한다. 즉, $f_i \trianglelefteq A$이다. 이 때 $f_i \ntrianglelefteq A[1..t]$를 만족하는 가장 큰 자연수 t를 택하자. $\delta^*(v_0, A[1..t]) = v$라 두면 (a)에 의해 $v \in V - \{\$\}$이다. 이제, v에서의 다음 경로가 항상 $\$$임을 보일 것이다. $\$$에서 출발하는 경로 $\delta^*(\$, \Sigma^*)$는 항상 $\$$이므로 $\delta(v, A[t+1])$가 $\$$임을 보이면 충분하다. 보조정리 3.3에 의해 $v = LSV(A[1..t])$이고 $LS(A[1..t])$는 t의 정의에 의해 $f_i[1..|f_i|-1]$ $(\in P)$이다. 즉, 문자열 $str(v)$는 $f_i[1..|f_i|-1]$이다. 따라서 $\delta(v, A[t+1]) = \delta(v, f_i[|f_i|]) = LSV(f_i) = \$$ 임을 알 수 있

다. 이는 $\delta^*(v_0,A) \in V'$라는 가정에 모순이다. □

4. LCNSS문제 해결 알고리즘

본 알고리즘은 다음의 3단계로 구성된다. 첫째 F의 접두사를 이용하여 G_F를 생성하고, 다음으로 G_F에서 \$를 삭제한 그래프를 생성한다. 마지막으로 변형된 그래프에서 최장경로를 찾아 $LCNSS_F$문제를 해결한다.

4.1 G_F 생성을 위한 함수

G_F의 정점은 $F \cup P$에 해당하므로 $O(N)$의 시간에 쉽게 생성할 수 있으나 간선은 δ함수의 계산에 의존한다. 본 논문에서는 δ함수를 효율적으로 구하기 위해 다음 함수들을 이용한다.

정의 7. SV함수 (SV: $V' \to V'$)

$SV(v)$는 $str(v)$의 진접미사 중 P에 속하는 최장문자열이 나타내는 정점을 나타내는 함수다. 즉, $v \in V'$에 대해 $SV(v) = LSV(str(v)[2..|str(v)|])$ 이다.

단, $|str(v)| \le 1$ 일 때 $SV(v) = v_0$로 정의한다.

정의 8. PV함수 (PV: $V' \to V'$)

$PV(v)$는 $str(v)$의 진접두사 중 P에 속하는 최장문자열이 나타내는 정점을 나타내는 함수다. 즉, $v \in V'$에 대해

$$PV(v) = ver'(str(v)[1..|str(v)|-1]) \text{ 이다.}$$

단, $PV(v_0) = v_0$로 정의한다.

정의 9. LC함수 (LC: $V' - \{v_0\} \to \Sigma$)

$LC(v)$는 $str(v)$의 마지막 문자를 나타내는 함수다. 즉, $v \in V' - \{v_0\}$에 대해 $LC(v) = str(v)[\,|str(v)|\,]$ 이다.

위의 함수들을 이용하여 다음의 결과들을 얻는다.

보조정리 4.1. $v \in V'$이고 $|str(v)| > 1$ 일 때, $SV(v) = \delta(SV(PV(v)), LC(v))$이다.

증명. $A = str(v)$, $B = str(PV(v))$, $x = LC(v)$로 두면 $A = Bx$이고 $SV(v) = LSV(A[2..|A|]) = LSV(B[2..|B|]x) = LSV(LSV(B[2..|B|]) \oplus x) = LSV(SV(PV(v)) \oplus x)$이다. □

보조정리 4.2. $v \in V' - \{v_0\}$ 이고 $x \in \Sigma$ 에 대해 $v \oplus x \notin P \cup F$일 때, $LSV(v \oplus x) = LSV(SV(v) \oplus x)$ 이다.

증명. $A = str(v)$, $B = v \oplus x$라 하자. 조건에 의해 $A \in P$이고 $B \notin P \cup F$이므로 $LSV(B) \ne \$$이고 $str(LSV(B)) \ne B$이다. 따라서 $LSV(B) = LSV(B[2..|B|]) = LSV(A[2..|A|]x)$와 같다. 따름정리 3.2에 의해 $LSV(A[2..|A|]x) = LSV(LSV(A[2..|A|]) \oplus x)$이고 $LSV(LSV(A[2..|A|]) \oplus x) = LSV(LSV(str(v)[2..|str(v)|]) \oplus x) = LSV(SV(v) \oplus x)$이다. □

따름정리 4.3. $v_j \in V'$와 $x \in \Sigma$에 대해 다음과 같다.

$$\delta(v_j,x) = \begin{cases} v_j \oplus x \in F \cup P \text{이면, } ver(v_j \oplus x) \\ v_j \oplus x \notin F \cup P \text{일 때, } \begin{cases} j = 0 \text{이면 } v_0 \\ j \ne 0 \text{이면 } \delta(SV(v_j),x) \end{cases} \end{cases}$$

증명. $v_j \oplus x \in F \cup P$ 경우 자명하고 $v_j \oplus x \notin F \cup P$ 일 때 $j=0$인 경우 $LS(x) = \lambda$이므로 증명되고, $j \ne 0$인 경우 보조정리 4.2에 의해 증명된다. □

4.2 G_F 생성 알고리즘

G_F를 생성하는 알고리즘은 다음과 같다. 각 f_i $(1 \le i \le n)$의 길이 t인 접두사에 대응되는 정점들을 원소로 하는 집합 $S[t]$라 하자. 즉, $S[t] = \{v | v = ver(f_i[1..t])$, 단 $t \le |f_i|$ 이고 $1 \le i \le n\}$이다. 더불어 $V[t] = S[0] \cup S[1] \cdots \cup S[t]$라 하자. 본 알고리즘은 $t = 0, 1, \cdots, \lceil F \rceil$ 순서로 $S[t]$를 생성해 나가며 최종적으로 생성된 $\cup_{all} S[t]$가 G_F의 정점집합 V이다. 본 알고리즘은 다음의 자료구조들을 추가로 이용한다.

- $FLI[\lceil F \rceil]$: F를 길이별로 분류하기 위한 배열로, $FLI[len]$은 길이 len인 금지문자열 f_i의 인덱스 i를 원소로 하는 집합이다.
- $\delta[\|F\|][|\Sigma|]$, $SV[\|F\|]$: δ, SV 값을 저장할 배열.
- $lastPV[|F|]$: $lastPV[i]$는 f_i의 접두사 중 마지막으로 처리된 정점을 의미하며, t에 따라 각 단계별 $PV(ver(f_i[1..t]))$를 나타낸다. 따라서 루프 시작 시 $lastPV[i] \in S[t-1]$이다.

본 알고리즘의 의사코드(pseudocode)는 다음과 같다.

```
알고리즘 MAKE_G( F )
입력: 금지문자열 집합 F
출력: G(V,E)
δ[][]를 NOT_DEF으로 초기화 한다.
SV[]와 lastPV[]를 모두 v0으로 초기화 한다.
S[0]와 V을 모두 {v0}로 초기화 한다.
for all fi∈F do FLI[ |fi| ].INSERT(i)
for t ← 1 to ⌈F⌉ do
    for k ← t to ⌈F⌉ do
        for all i∈FLI[k] do
            PV ← lastPV[i]
            LC ← fi[t]
            if t = k then δ[PV][LC] ← $
            else if δ[PV][LC] is NOT_DEF then
                v ← NewVertex()
                S[t].INSERT(v)
                δ[PV][LC] ← v
                if t≤1 then SV[v] ← v0
                else SV[v] ←δ[ SV[ PV ] ][LC]
            lastPV[i] ← δ[PV][LC]
    for all v∈S[t−1] do
        for all w∈Σ do
            if δ[v][w] is NOT_DEF then
                if v=v0 then δ[v][w] ← v0
                else δ[v][w] ← δ[ SV[v] ][w]
            E.INSERT( <v, w, δ[v][w]> )
    V ← V∪S[t]
for all w∈Σ do E.INSERT( <$, w, $> )
return G(V,E)
```

MAKE_G의 시간 복잡도(complexity)는 다음과 같다. 4행에서 f_i의 길이가 모두 계산되므로 $O(N)$시간이 소요된다. 5~24행의 반복문에서 t가 증가함에 따라 각 f_i의 t번째 문자들이 한 번씩 선택되므로 N회의 연산이 이루어지고, 18~23행의 내부 루프에서 각 정점마다 $|\Sigma|$만큼 δ함수가 계산되므로 총 $O(|\Sigma| \times N)$시간이 소요된다. 따라서 MAKE_G는 총 $O(|\Sigma| \times N)$의 시간이 걸린다.

4.3 LCNSS문제 해결 알고리즘

본 알고리즘은 F를 입력받아 우선 G_F를 생성하고 정점 \$를 삭제한 그래프 G_F'를 생성한다. 그리고 G_F'에서 최장경로를 찾아 $LCNSS_F$를 구한다.

정의 10. 그래프 $G_F'(V', E')$

G_F의 부분그래프로서 G_F'는 $V'(= V - \{\$\})$을 정점 집합으로 갖고 E에서 \$와 연결된 간선들을 삭제하여 생성된 그래프이다. 즉, E'는 다음과 같이 정의된다.

$$E' = E - \{< v_i, x, v_j > \mid < v_i, x, v_j > \in E \text{ 이고 } v_j = \$\}$$

정리 4.5. G_F'가 사이클이 없는 유향그래프(directed acyclic graph)이면 G_F'의 v_0에서 시작하는 최장경로에 대응되는 문자열이 $LCNSS_F$이고, G_F'에 사이클이 있다면 $LCNSS_F$는 존재하지 않는다.

증명. 정리 3.5를 통해 G_F'에서 v_0에서 시작하는 경로에 대응되는 문자열이 $CNSS_F$와 일대일대응 이라는 것을 알 수 있다. G_F'에 사이클이 존재하는 어떤 경로를 M이라 하자. 따름정리 3.4에 의해 M의 임의의 정점은 반드시 v_0으로부터 도달 가능하므로 v_0에서 시작하는 무한히 긴 경로를 찾을 수 있다. 따라서 대응되는 무한히 긴 $CNSS_F$를 만들 수 있으므로 사이클이 존재할 때 $LCNSS_F$는 존재하지 않는다. 사이클이 존재하지 않으면, v_0에서 시작하는 유한 최장경로를 찾을 수 있고 이에 대응되는 $CNSS_F$가 $LCNSS_F$임을 알 수 있다. □

정리 4.5에 따라 G_F'에서 사이클이 없다면 v_0에서 출발하는 최장경로를 찾아 $LCNSS_F$를 구한다. 최장경로 문제는 일반적인 그래프에서 NP-hard이나[7], 사이클이 없는 그래프에서는 다항시간에 해결할 수 있다. 사이클 검사는 잘 알려진 방법으로 깊이우선탐색(DFS)을 통해 검사할 수 있으며 $O(|V|+|E|)$시간이 걸린다. 사이클이 존재하지 않는다면 DFS또는 위상정렬(topological sort)를 이용해 $O(|V|+|E|)$시간에 최장경로를 검색하고 최장경로에 대응되는 문자열을 출력하여 $LCNSS_F$를 구한다. 따라서 $LCNSS_F$를 찾는 알고리즘은 총 $O(|\Sigma| \times N)$의 시간의 수행시간을 갖는다.

5. 결 론

본 논문에서는 LCNSS문제에 대한 기존의 연구와 달리 접두사를 이용한 그래프 표현이 가능함을 보였고, 상수 크기의 알파벳에 대해 $O(N)$의 수행시간에 LCNSS의 존재여부와 존재한다면 이를 찾는 알고리즘을 제시하였다.

문자열 포함 및 불포함 문제들에 대한 연구는 DNA 염기서열과 관련된 문제에 많은 응용이 기대 된다[8-10]. 특히 DNA에서 특정한 DNA 세그먼트(segment)가 존재하면 생명체에 유전 질환이 발생함이 알려졌으며 현재 이런 유전 질환을 발현시키는 DNA 세그먼트들이 많이 발견되었다. 본 논문을 이 세그먼트들을 금지문자열로 하는 DNA 분석 문제에 응용할 수 있을 것으로 기대한다.

참 고 문 헌

[1] J. Gallant, D. Maier, and J. Storer, On finding minimal length superstrings, *Journal of computer and System Sciences*, 20, 50-58, 1980.

[2] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis, Linear approximation of shortest superstrings, *In Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, 328-336, 1991.

[3] D. S. Hirschberg, Algorithms for the Longest Common Subsequence Problem, *Journal of the ACM*, 24, 4, 664-675, 1977.

[4] V. G. Timkovsky, Complexity of common subsequence and supersequence problems and related problems, *Cybernetics and Systems Analysis* 25, 5, 565-580, 1990.

[5] A. R. Rubinov, and V. G. Timkovsky, String noninclusion optimization problems, *SIAM Journal on Discrete Mathematics* 11, 3, 456-467, 1998.

[6] T. Jiang, and V. G. Timkovsky, Shortest consistent superstrings computable in polynomial time, *Theoretical Computer Science* 143, 1, 113-122, 1995.

[7] M. R. Garey, and D. S. Johnson, *Computers and Intractability*, Freeman, 1979.

[8] P. A. Pevzner, and R. J. Lipshutz, Towards DNA Sequencing Chips, *Proceedings of the 19th International Symposium on Mathematical Foundations of Computer Science* 1994, August 22-26, 143-158, 1994.

[9] T. Jiang, and M. Li, DNA Sequencing and String Learning, *Mathematical Systems Theory* 29, 4, 387-405, 1996.

[10] M. Li. Towards a DNA sequencing theory. *31st IEEE Symposium on Foundations of Computer Science*, 125-134, 1990.