# 문자열의 근사커버 찾기
## (Finding Approximate Covers of Strings)

심 정 섭 † 박 근 수 †† 김 성 렬 ††† 이 지 수 ††††
(Jeong Seop Sim) (Kunsoo Park) (Sung-Ryul Kim) (Jee-Soo Lee)

**요 약** 반복적인 문자열에 대한 연구는 최근 들어 여러 분야에서 활발히 진행되어 왔다. 특히 DNA 염기서열의 분석 등 분자생물학에서 그 필요성이 대두되고 있다. 주기, 커버, 시드, 스퀘어 등이 반복적인 문자열의 대표적인 예들이다. 근사문자열 매칭 분야에서도 근사주기, 근사스퀘어 등 반복적인 문자열에 관한 연구가 진행되고 있다.

본 논문에서는 근사커버의 개념을 제시한다. 길이가 각각 $m, n$인 두 문자열 $P, T$가 주어졌을 때, $P$가 $T$의 근사커버가 되는 최소의 편집거리를 $O(mn)$ 시간, 최소의 가중편집거리를 $O(mn^2)$ 시간에 찾는 알고리즘을 제시한다. 또한, 문자열 $T$만 주어졌을 때, $T$의 최소 근사커버 거리를 갖는 문자열 $P$를 찾는 문제가 NP-완전 결과임을 증명한다.

**키워드** : 근사문자열 매칭, 편집거리, 가중편집거리, 근사커버, NP-완전

***Abstract*** Repetitive strings have been studied in such diverse fields as molecular biology, data compression, etc. Some important regularities that have been studied are periods, covers, seeds and squares. A natural extension of the repetition problems is to allow errors. Among the four notions above, approximate squares and approximate periods have been studied.

In this paper, we introduce the notion of approximate covers which is an approximate version of covers. Given two strings $P(|P|=m)$ and $T(|T|=n)$, we propose an algorithm which finds the minimum distance $t$ such that $P$ is a $t$-approximate cover of $T$. The algorithm takes $O(mn)$ time for the edit distance and $O(mn^2)$ time for a weighted edit distance. Furthermore, when only $T$ is given, we show that the problem of finding a string which is an approximate cover of $T$ with the minimum distance is NP-complete.

**Key words** : approximate string matching, editdistance, weighted edit distance, approximate

## 1. 서 론

문자열 매칭(string matching) 알고리즘은 많은 분야에서 활발히 연구되고 있다. 특히 반복적인 형태의 문자열에 대한 연구는 압축 알고리즘, 분자생물학, 오토마타 이론 등 다양한 분야에서 진행되고 있다. 주기(period), 커버(cover), 시드(seed), 스퀘어(square) 등이 반복적인 문자열의 대표적인 예이다.

† 학생회원 : 서울대학교 컴퓨터공학부
jssim@theory.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
kpark@theory.snu.ac.kr
††† 비 회 원 : (주)와이즈넛 연구원
Sungryul.Kim@wisenut.com
†††† 종신회원 : 한국방송대학교 전자계산학과 교수
jslee@mail.knou.ac.kr
논문접수 : 2001년 5월 8일
심사완료 : 2001년 9월 24일

- 주기 : 문자열 $P$를 $k$번 반복적으로 연결(concatenation)시킨 문자열을 $P^k$라 하자. 길이 $n$인 문자열 $T$가 주어졌을 때, $T=P^kP'$ ($P'$은 $P$의 접두어, $k\geq 1$)으로 표현할 수 있는 문자열 $P$를 $T$의 주기라 한다. 일반적으로 그러한 $P$들 중에서 길이가 가장 짧은 문자열을 주기라고 한다.

예 1. 문자열 $abc$는 문자열 $abcabcab$의 주기이다. 즉, $abcabcab=(abc)^2ab$로 나타낼 수 있다.

[1]에서는 선형 시간 내에 주어진 문자열의 최단 주기를 찾는 알고리즘을 제시했고, [2]에서는 $O(n/\log\log n)$ 프로세서를 이용하여 $O(\log\log n)$ 시간에 모든 주기를 찾는 병렬 알고리즘을 제시했다.

- 커버(cover)와 시드(seed) : 주어진 문자열 $T$에

대해 어떤 문자열 $W$를 연결 또는 중첩(superposition)시켜서 $T$를 얻을 수 있을 때, $W$를 $T$의 커버라고 한다. 또한, $T$의 부분문자열(substring) $W$를 연결 또는 중첩시켜서 $T$의 상위문자열(superstring) $Z$를 얻을 수 있으면 $W$를 $T$의 시드라고 한다.

예 2. 문자열 $aba$는 문자열 $ababaaba$의 커버이면서 $aababaabab$의 시드이다.

[3]에서는 주어진 문자열의 커버를 구하는 선형 시간 알고리즘을 제시하였고, [4]에서는 선형 시간 온라인 알고리즘이 제시되었다. 또한, [5]에서는 주어진 문자열 내에 커버가 존재하는지의 여부를 결정하는 $O(\log\log n)$ 시간 CRCW PRAM 알고리즘을 제시하였다. [6]에서는 문자열 내에서 모든 시드를 구하는 $O(n\log n)$ 시간 알고리즘을 제시하였다.

• 스퀘어(square) : 주어진 문자열 $T$를 $T=XY^2Z$ ($X, Y, Z$는 문자열, $Y\neq\varepsilon$, $\varepsilon$은 공백문자열)로 나타낼 수 있으면 $Y^2$을 스퀘어(또는 tandem repeat)라 한다. [7]에서는 길이 $n$인 문자열의 모든 스퀘어를 $O(n\log n)$ 시간에 찾는 알고리즘을 제시하였고, [8]에서는 최적 병렬 알고리즘을 제시하였다.

한편 문자열들이 어느 정도 불일치하더라도 이를 허용하여 근사적으로 일치하는 것으로 간주하는 근사문자열 매칭(approximate string matching)에 대한 연구도 활발히 진행되고 있다. 이때, 문자열 사이의 불일치 정도를 두 문자열의 오차(error) 또는 거리(distance)라고 한다. 대표적인 거리함수로는 해밍거리(Hamming distance), 편집거리(edit distance), 가중편집거리(weighted edit distance) 등이 있다.

근사문자열 매칭에서도 반복적인 문자열 즉, 스퀘어와 주기에 대한 연구가 이루어져 왔다. 문자열 $T$($|T|=n$)가 $T=XYY'Z$($X, Y, Y', Z$는 문자열, $Y, Y'\neq\varepsilon$)로 주어졌을 때, $Y, Y'$이 미리 정해진 어떤 값 $k$ 이내의 거리를 가지면 $YY'$을 근사스퀘어라 한다. [9]에서는 $k$ 이내의 편집거리를 가지는 근사스퀘어를 $O(kn\log k\log n)$ 시간에 구하는 알고리즘을 제시하였고, [10]에서는 임의의 비용 행렬이 주어졌을 때 모든 근사스퀘어를 $O(n^2\log n)$ 시간에 구하는 알고리즘을 제시하였다.

주어진 문자열 $T$가 $T=T_1\cdots T_r$(각 $T_i\neq\varepsilon$, $T_r$은 $P$의 근사접두어)로 주어지고, 각 $T_i$와 문자열 $P$의 거리가 $k$ 이내일 경우에 $P$를 $T$의 $k$-근사주기라 한다. [11]에서는 근사주기의 개념을 제시하고 두 문자열 $P$($|P|=m$), $T$($|T|=n$)가 주어졌을 때 $P$가 $T$의 근사주기가 되는 최소의 편집거리 $k$를 구하는 $O(mn)$ 시간 알고리즘과 최소의 가중편집거리 $k$를 구하는 $O(mn^2)$ 시간 알고리즘을 제시하였다.

[11]에서는 편집거리에 대한 $O(mn)$ 시간 알고리즘을 위해 [12]과 [13]에서 제시된 방법을 이용한다. [8,9]에서는 $A, B$($|A|=m, |B|=n, B=bB', b\in\Sigma$)의 편집거리에 대한 해가 주어졌을 때, $A$와 $B'$의 편집거리에 대한 해를 $O(m+n)$ 시간에 계산하는 알고리즘들을 각각 제시하였다.

본 논문에서는 근사커버의 개념을 제시하고, 동적 프로그래밍 기법을 이용하여 길이 $n$인 문자열 $T$와 길이 $m$인 문자열 $P$가 주어졌을 때, $P$가 $T$의 근사커버가 되는 최소의 편집거리와 가중편집거리를 각각 $O(mn)$, $O(mn^2)$ 시간 내에 찾는 알고리즘을 제시한다. 또한, 문자열 $T$만 주어졌을 때, $T$에 대해 최소 근사커버 거리를 갖는 문자열 $P$를 찾는 문제가 NP-완전임을 증명한다.

본 논문은 다음과 같이 구성된다. 2장에서는 본 논문의 알고리즘을 위한 몇 가지 용어에 대한 정의와 관련 연구 결과들을 제시한다. 3장에서는 근사커버를 정의하고 알고리즘과 NP-완전 결과를 설명한 뒤, 4장에서는 결론과 향후 연구 방향을 제시한다.

## 2. 문자열의 거리와 $D$ 테이블

### 2.1 문자열

문자열이란 알파벳 집합 $\Sigma$에서 0개 이상의 문자들이 연결된 형태이다. 0개 이상의 $\Sigma$의 문자들로 이루어진 모든 문자열의 집합을 $\Sigma^*$라 한다. 공백문자는 $\Delta\notin\Sigma$로 나타내고 공백문자열은 $\varepsilon$으로 나타내기로 한다. 문자열 $A$의 길이를 $|A|$로 나타내고 $A$의 $i$번째 문자는 $A[i]$로 나타내기로 한다. 문자열 $A$의 $i$번째 문자부터 $j$번째 문자까지의 연결을 $A[i..j]$로 나타내고 이를 $A$의 부분문자열(substring)이라고 한다. 역으로, $A$는 $A[i..j]$의 상위문자열이라 한다. 또한, 문자열 $A$의 임의의 위치에서 0개 이상의 문자를 삭제하여 얻어지는 문자열 $W$를 $A$의 부분서열(subsequence)이라 하고 역으로 $A$는 $W$의 상위서열(supersequence)이라고 한다. 예를 들어 $A=abcdefg$일 때, $W=abeg$는 $A$의 부분서열이고 $A$는 $W$의 상위서열이다. 문자열의 집합 $S$에 대해 어떤 문자열 $W$가 $S$ 내의 모든 문자열의 상위서열일 때, $W$를 $S$의 공통상위서열이라고 한다. 예를 들어 $S=\{abc, abd, bcd\}$일 때, $abcd$는 $S$ 내의 모든 문자열들의 상위서열이므로 $S$의 공통상위서열이다.

### 2.2 거리와 편집연산

두 문자열 $A, B$의 거리 $f(A, B)$는 $A$를 $B$로 변환하기 위해 필요한 최소 비용으로 정의된다. 잘 알려진 몇 가지 거리함수들이 있다. 두 문자열 $A, B$의 편집거리는 $A$를 $B$로 바꾸는데 필요한 최소의 편집연산(edit operation) 수이다. 편집연산에는 한 문자의 삽입(insert), 삭제(delete), 교체(change) 등이 있다. 해밍거리는 교체 연산만이 허용될 때 $A$를 $B$로 바꾸기 위한 최소의 연산 수이다. 이는 삽입, 삭제 연산을 허용하지 않기 때문에 $|A| = |B|$일 때에만 정의된다. 편집거리는 비용행렬(penalty matrix)을 이용하여 일반화할 수 있다. 비용행렬은 그림 1과 같이 모든 문자 쌍에 대한 교체 비용, 그리고 각 문자의 삽입, 삭제 비용을 정의한다. $A$를 $B$로 변환하는 데 필요한 최소 비용을 주어진 비용행렬을 이용하여 계산한 것을 가중편집거리라고 한다. 편집거리는 가중편집거리의 특수한 경우로서 각 편집연산의 비용이 1임을 알 수 있다. 모든 문자 $a, b, c \in \Sigma$에 대하여 비용행렬이 다음과 같은 네 가지 조건을 만족할 때 메트릭(metric)이라 한다.

| | $a$ | $b$ | $c$ | $\Delta$ |
|---|---|---|---|---|
| $a$ | 0 | 2 | 1 | 2 |
| $b$ | 2 | 0 | 2 | 1 |
| $c$ | 1 | 2 | 0 | 1 |
| $\Delta$ | 2 | 1 | 1 | 0 |

그림 1 비용행렬의 예

1) $f(a, b) \geq 0$
2) $f(a, b) = f(b, a)$
3) $f(a, a) = 0$
4) $f(a, c) \leq f(a, b) + f(b, c)$ (삼각부등식)

그림 1은 메트릭인 비용행렬이며, 본 논문에서 거리함수로 쓰이는 가중편집거리는 메트릭인 비용행렬을 사용함을 가정한다.

### 2.3 $D$ 테이블

두 문자열 $X(|X| = m)$, $Y(|Y| = n)$와 메트릭 비용행렬이 주어졌을 때, 이들의 거리를 구하기 위해 동적 프로그래밍 기법을 이용한 $(|X|+1) \times (|Y|+1)$ 크기의 테이블을 이용할 수 있다. 이 테이블을 $D$ 테이블이라 하자. $D$ 테이블의 각 원소 $D[i, j]$는 $X[1..i]$와 $Y[1..j]$의 거리를 나타낸다. $D$ 테이블의 첫 번째 행과 열은 각각 다음과 같이 초기화된다.

$$\left.\begin{array}{l} D[0,0] = 0, \\ D[i,0] = D[i-1,0] + f(X[i], \Delta)\ (1 \leq i \leq m), \\ D[0,j] = D[0,j-1] + f(\Delta, Y[j])\ (1 \leq j \leq n) \end{array}\right\} \quad (1)$$

초기화되지 않은 각 $D[i, j]$는 다음과 같은 관계식으로 구할 수 있다.

$$D[i,j] = \min \left\{ \begin{array}{l} D[i,j-1] + f(Y[j], \Delta), \\ D[i-1,j] + f(\Delta, X[i]), \\ D[i-1,j-1] + f(X[i], Y[j]) \end{array} \right\} \quad (2)$$

| | | $a$ | $b$ | $c$ | $d$ | $a$ | $b$ | $c$ | $c$ |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $a$ | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| $b$ | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| $d$ | 3 | 2 | 1 | 1 | 1 | 2 | 3 | 4 | 5 |
| $b$ | 4 | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 4 |
| $c$ | 5 | 4 | 3 | 2 | 3 | 3 | 3 | 2 | 3 |
| $d$ | 6 | 5 | 4 | 3 | 2 | 3 | 4 | 3 | 3 |

그림 2 $D$ 테이블

이때 $D[m, n]$의 값이 두 문자열 $X, Y$의 거리이다. 예를 들어 $X = abdbcd$, $Y = abcdabcc$의 두 문자열이 주어졌을 때 이들의 편집거리는 식 (1), (2)를 적용하여 행순 또는 열순으로 계산하여 구하면 된다. (그림 2 참조)

## 3. 근사커버를 구하는 알고리즘

### 3.1 근사커버

두 문자열 $P(|P| = m)$, $T(|T| = n)$와 거리함수 $f$가 주어졌을 때, $T$는 $T$의 부분문자열들 $T_1, T_2, \cdots, T_r$ $(1 < r \leq n)$의 연결과 중첩으로 이루어진다. 이 때 각 $T_i (1 \leq i \leq r)$를 커버블럭이라 하자. 이 때, 각 $i$에 대해 $f(P, T_i) \leq t$이면 $P$를 $T$의 $t$-근사커버라 한다. 이 때 $t$를 $P$의 $T$에 대한 근사커버 거리라고 하자.

### 3.2 알고리즘

이 절에서는 두 문자열 $P$와 $T$, 그리고 거리함수 $f$가 주어졌을 때, $P$의 $T$에 대한 최소 근사커버 거리 $t$를 찾는 알고리즘 Approximate_Cover를 제시한다. 알고리즘 Approximate_Cover는 다음과 같이 두 단계로 구성된다.

**알고리즘 Approximate_Cover( $P$, $T$)**

**1. $P$와 $T$의 각 부분문자열( $T[i..j]$, $1 \leq i \leq j \leq n$)과의 거리 $w_{ij}$ 계산**

**2. $P$의 $T$에 대한 최소 근사커버 거리 $t$ 계산**

각 단계별 수행 방법은 다음과 같다.

**1단계:** $1 \le i \le n$인 모든 $i$에 대하여 $P$와 $T[i..n]$에 대한 $D$ 테이블을 만들어 $w_{ij}$를 구할 수 있다.

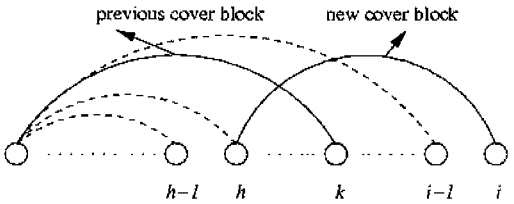


그림 3 $t_i$의 계산

**2단계:** 최소 근사커버 거리 $t$는 동적 프로그래밍 방법을 이용하여 구할 수 있다. $P$의 $T[1..i]$에 대한 최소 근사커버 거리를 $t_i$라 할 때, $t_1$부터 $t_n$까지 차례로 구한다. ( $t_0 = 0$)

$P$가 $T[1..i-1]$ ( $i \ge 2$)의 $t_{i-1}$-근사커버일 때 $t_i$는 다음의 식으로 표현할 수 있다.

$$t_i = \min_{1 \le h \le i} \{\max(w_{hi}, \min_{h-1 \le k \le i-1} t_k)\} \quad (3)$$

식 (3)을 이용하여 $T[1..i]$의 마지막 커버블럭과 $t_i$를 결정할 수 있다. 식 (3)은 $T[1..i]$의 마지막 커버블럭의 시작 위치 $j_i$가 $w_{hi}$와 각 $k(h-1 \le k \le i-1)$에 대해 최소인 $t_k$ 중 큰 값, 즉 $\max(w_{hi}, \min_{h-1 \le k \le i-1} t_k)$를 최소화하는 $h$임을 의미하며 이는 근사커버의 정의에 의해 참이다. (그림 3 참조) 그러한 $h$가 여러 개 있을 경우 가장 왼쪽 위치를 $j_i$로 결정한다. 이 때, $w_{hi}$는 단계 1에서 이미 계산된 값이다. 따라서, $\min_{h-1 \le k \le i-1} t_k$를 효율적으로 구해야 하는데 $h$를 1에서 $i$까지 증가시키면서 최소값을 찾으면 $h$가 증가할 때마다 중복된 비교 또는 부가적인 공간이 필요하게 되므로 반대로 $h$를 $i$에서 1까지 감소시키면서 구한다. 즉, $T$의 위치 $h$와 $i-1$ 사이의 각 $t$의 최소값을 $t_{\min}$이라고 할 때, $h-1$과 $i-1$ 사이의 각 $t$의 최소값은 $\min_{h-1 \le k \le i-1} t_k = \min(t_{h-1}, t_{\min})$이다. 따라서 식 (3)은 다음과 같이 쓸 수 있다.

$$t_i = \min_{1 \le h \le i} \{\max(w_{hi}, \min(t_{h-1}, t_{\min}))\} \quad (4)$$

### 3.3 수행시간 분석

거리함수로 가중편집거리를 사용할 때 이 알고리즘의 수행시간은 $O(mn^2)$이다. 즉, 1단계에서는 $O(mn)$ 크기의 $D$ 테이블을 $n$개 만들게 되므로 $O(mn^2)$ 시간이 필요하며, 2단계에서는 식 (4)에 의해 각 $i$에서 $t_i$를 결정하기 위해 $O(n)$ 시간이 걸리므로 총 $O(n^2)$ 시간이 필요하다. 따라서 알고리즘 **Approximate_Cover**의 수행시간은 $O(mn^2)$이다.

만일 거리함수로 편집거리를 사용한다면 1단계에서 [12, 13]에서 제시된 알고리즘을 이용하여 $O(mn)$ 시간에 $P$와 $T$의 모든 부분문자열에 대한 거리를 구할 수 있다. [12,13]에서는 두 문자열 $A(|A| = m)$, $B(|B| = n, B = bB', b \in \Sigma)$ 사이의 거리에 대한 해가 주어졌을 때 $A$와 $B'$ 사이의 거리에 대한 해를 $O(m+n)$ 시간에 구하는 알고리즘을 제시하였다. 즉, $P$와 $T[1..2m]$의 거리에 대한 해를 이용하여 $P$와 $T[2..2m]$의 거리에 대한 해를 $O(m)$ 시간에 구할 수 있고 이러한 방법으로 $P$와 $O(m)$ 크기의 $T$의 모든 부분문자열 사이의 거리에 대한 해를 $O(mn)$ 시간에 구할 수 있다. 2단계 역시 $O(mn)$ 시간이 걸리게 되는데, 그 이유는 각 $i(1 \le i \le n)$에서 $t_i$를 결정할 때 $2m$개 즉, $O(m)$개의 마지막 커버블럭의 시작 가능 위치에 대해서만 상수시간의 비교를 수행하면 된다. ( $P$는 최소한 $T$의 $m$-근사커버이고, 길이가 $2m$이 넘는 문자열과 $P$와의 편집거리는 당연히 $m$을 넘는다.) 따라서, $O(m)$ 시간이 걸리게 된다. 즉, 편집거리에 대해 $t_i$는 다음의 식 (5)으로 구할 수 있다.

$$t_i = \min_{i-2m+1 \le h \le i} \{\max(w_{hi}, \min(t_{h-1}, t_{\min}))\} \quad (5)$$

정리 1. 두 문자열 $P$와 $T$가 주어졌을 때, 알고리즘 **Approximate_Cover**는 $P$의 $T$에 대한 최소 근사커버 거리 $t$를 거리함수로 가중편집거리를 사용할 때 $O(mn^2)$ 시간, 편집거리를 사용할 때 $O(mn)$ 시간에 각각 찾을 수 있다.

### 3.4 NP-완전 결과

앞 절에서는 두 문자열 $P$와 $T$가 주어졌을 때 $P$의 $T$에 대한 최소 근사커버 거리를 찾는 다항식시간 알고리즘을 제시하였다. 여기에서는 $P$가 주어지지 않고 $T$만 주어졌을 때 $T$에 대한 최소 근사커버 거리를 갖는 문자열 $P$를 찾는 문제가 NP-완전임을 보인다. 근사커버 문제(**AC 문제**라고 부르기로 한다.)의 NP-완전은 이미 알려진 NP-완전 문제인 최단공통상위서열 문제 (shortest common supersequence 문제, **SCS 문제**라고 부르기로 한다.)를 이용하여 증명한다.

먼저, SCS 문제와 AC 문제의 결정형(decision version) 정의는 다음과 같다.

SCS 문제: 양의 정수 $m$, 유한알파벳 $\Sigma$로 이루어진 문자열들의 집합 $S(\subset \Sigma^*)$가 주어졌을 때, 길이가 $m$보

다 길지 않은 $S$의 공통상위서열이 존재하는가?

AC 문제: 양수 $t$, 메트릭인 비용행렬, 그리고 유한알파벳 $\Sigma'$으로 이루어진 문자열 $T(\in(\Sigma')^*)$가 주어졌을 때, $T$에 대한 근사커버 거리가 $t$ 이하인 문자열이 존재하는가?

SCS 문제의 AC 문제로의 변형(transformation)은 다음과 같다. SCS 문제는 이진 알파벳에 대해서도 NP-완전([14,15])이므로 $\Sigma=\{0,1\}$로 가정한다. 또한, $S=\{S_1,\cdots,S_n\}$ $(S_i\in\Sigma^*, 1\le i\le n)$으로 가정한다. 먼저, $t=m$, $\Sigma=\Sigma\cup\{a,b,\#,\$,*_1,*_2,\Delta\}$으로 정의한다. $T=\#S_1\$\#S_2\$\cdots\#S_n\$\#*_1^m\$\#*_2^m\$$로 정의하고 비용행렬을 그림 4와 같이 정의한다. 그림 4에서 음영된 부분은 $m$ 이상인 메트릭을 만족하는 어떤 값이 되어도 관계없다. 여기에서는 음영된 부분의 값을 $m+1$로 가정한다. 이때 그림 4의 비용행렬은 메트릭이다. 다항식 시간에 임의의 SCS 문제가 AC 문제로 변형됨을 쉽게 알 수 있다.

| | 0 | 1 | a | b | $*_1$ | $*_2$ | # | $ | Δ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 2 | 1 | 2 | 2 | 2 | | | 1 |
| 1 | 2 | 0 | 2 | 1 | 2 | 2 | | | 1 |
| a | 1 | 2 | 0 | 2 | 1 | 1 | | | 1 |
| b | 2 | 1 | 2 | 0 | 1 | 1 | | | 1 |
| $*_1$ | 2 | 2 | 1 | 1 | 0 | 2 | | | 2 |
| $*_2$ | 2 | 2 | 1 | 1 | 2 | 0 | | | 2 |
| # | | | | | | | 0 | | |
| $ | | | | | | | | 0 | |
| Δ | 1 | 1 | 1 | 1 | 2 | 2 | | | 0 |

그림 4 비용행렬

보조정리 2. $T$가 위와 같이 정의될 때, 문자열 $P$가 $T$의 $m$-근사커버이기 위해서는 $P$는 $\#\alpha\$$ $(\alpha\in\{a,b\}^m)$의 형태가 되어야 한다.

증명. 먼저 $P$가 #과 $를 가지지 않을 경우, 당연히 근사커버 거리는 $m+1$ 이상이 되므로 ($T$의 어느 커버블럭은 반드시 #이나 $를 가진다.) $P$는 하나 이상의 #과 $를 가져야 한다. $P$와 각 커버블럭의 #과 $의 개수는 모두 같아야 한다. 만일 다를 경우 비용행렬에 의해 근사커버 거리는 $m$보다 커지게 된다. 이제 $P$가 두 개의 #을 가질 경우를 가정하자. (둘 이상의 경우도 비슷하게 증명할 수 있다.) 이 때 $P$는 반드시 두 개의 $를 가져야 한다. 왜냐하면, 만약 $P$의 #과 $의 개수가 다를 경우 최소한 하나의 커버블럭은 $P$와 같은 수의 #과 $를 가질 수 없으므로 근사커버 거리가 $m$ 이하일 수 없다. 따라서, $T$의 마지막 커버블럭은 반드시 $\#*_1^m\$\#*_2^m\$$이다. 이 때, 이 커버블럭과 $P$와의 거리가 $m$ 이하이기 위해서 $P$에는 $*_1$과 $*_2$의 개수의 합이 최소한 $m$개 이상이어야 한다. 그러나 이 경우, $P$와 다른 커버블럭과의 거리가 $m$을 넘게 되므로 결국 $P$는 하나씩의 #과 $를 가져야 한다.

이제 $P$가 $\#\alpha\$(\alpha\in\{a,b\}^m)$의 형태가 됨을 증명한다. $P$가 하나씩의 #과 $를 가지므로 $T$는 각 $마다 커버블럭을 이룬다. $P$가 $\beta\#\alpha\$\gamma(\alpha,\beta,\gamma\in\{0,1,a,b,*_1,*_2,\Delta\}^*$의 형태라고 가정하고, $\alpha$에 $i(i\ge1)$개의 $*_1$이 있다고 가정하자. 마지막 두 개의 커버블럭 즉, $\#*_1^m\$$과 $\#*_2^m\$$을 고려할 때, $P$와 $\#*_2^m\$$과의 거리가 $m$ 이하이기 위해 $\alpha$에는 $i$개의 $*_2$도 있어야 하며 나머지 $m-2i$개의 문자는 $a$ 또는 $b$로 이루어져야 한다. 그러나, 이 경우 다른 어떤 커버블럭과의 거리도 $m$ 이하일 수 없다. 따라서 $\alpha$는 $*_1$ 또는 $*_2$를 가질 수 없다. 또한, $0,1,\Delta$는 마지막 두 커버블럭의 $*_1$, $*_2$와의 비용이 2이므로 $\alpha$는 $0,1,\Delta$ 역시 가질 수 없다. 따라서, $\beta$, $\gamma=\varepsilon$이며 $\alpha\in\{a,b\}^m$이다. □

정리 3. AC 문제는 NP-완전이다.

증명. AC 문제가 NP에 속함은 쉽게 알 수 있다. AC 문제가 NP-완전임을 증명하기 위해서 $S$의 최단공통상위서열 $S'(|S'|\le m)$을 구하면 $T$의 $m$-근사커버를 구할 수 있고 그 역도 성립함을 보인다.

먼저 $T$의 $m$-근사커버인 문자열 $P$를 가정하자. 보조정리 2에 의해 $P=\#\alpha\$$ $(\alpha\in\{a,b\}^m)$이다. 이 때 $\alpha$의 문자 $a$를 0으로, $b$를 1로 교체시키면 길이가 $m$인 $S$의 공통상위서열이 얻어진다. 역으로 $S$의 최단공통상위서열 $S'(|S'|\le m)$을 구하면 $S'$ 내의 모든 0을 $a$로, 1을 $b$로 교체하여 #과 $를 $S'$의 앞과 뒤에 각각 붙이면 $T$에 대한 $m$-근사커버를 구할 수 있다. ($|S'|<m$일 경우에는 $m-|S'|$개의 $a$ 또는 $b$의 문자를 $S'$에 붙이면 된다.)

따라서, AC 문제의 해를 구하면 SCS의 문제를 풀 수 있고 그 역도 성립하므로 AC 문제는 NP-완전이다. □

## 4. 결 론

본 논문에서는 근사커버의 개념을 제시하고, 두 문자열 $A(|A|=m)$, $B(|B|=n)$이 주어졌을 때 $A$의 $B$에 대한 최소 근사커버 거리를 가중편집거리와 편집거리에 대해 각각 $O(mn^2)$ 시간, $O(mn)$ 시간에 찾는 알고리즘을 제시하였다. 또한 문자열이 하나만 주어졌을 때 이

문자열에 대한 최소 근사커버를 찾는 문제는 거리함수가 메트릭일 경우에도 NP-완전임을 설명하였다.

근사커버는 근사주기, 근사스퀘어 등과 더불어 근사문자열 매칭에서의 반복적인 문자열에 대한 연구로서 패턴 매칭뿐만 아니라 DNA 염기 서열 분석 등 분자생물학 분야에서도 많이 응용될 것으로 기대된다.

## 참 고 문 헌


[1] M. Crochemore, String-matching and periods, *Bulletin of the European Association for Theoretical Computer Science* 39 (1989), 149-153.

[2] A. Apostolico, D. Breslauer and Z. Galil, Optimal parallel algorithms for periods, palindromes and squares, *Proc. 19th Int. Colloq. Automata Languages and Programming*, LNCS 623 (1992), 296-307.

[3] A. Apostolico, M. Farach and C. S. Iliopoulos, Optimal superprimitivity testing for strings, *Information Processing Letters* 39 (1991), 17-20.

[4] D. Breslauer, An on-line string superprimitivity test, *Information Processing Letters* 44 (1992), 345-347.

[5] C.S. Iliopoulos and K. Park, An optimal $O(\log\log n)$-time algorithm for parallel superprimitivity testing, *J. KISS* 21, 8 (1994), 1400-1404.

[6] C.S. Iliopoulos, D.W.G. Moore and K. Park, Covering a string, *Algorithmica* 16 (1996), 288-297.

[7] M.G. Main and R.J. Lorentz, An $O(n\log n)$ algorithm for finding all repetitions in a string, *J. Algorithms* 5 (1984), 422-432.

[8] A. Apostolico, Fast parallel detection of squares in strings, *Algorithmica* 8 (1992), 285-319.

[9] G.M. Landau and J.P. Schmidt, An algorithm for approximate tandem repeats, *Proc. 4th Symp. Combinatorial Pattern Matching, LNCS* 648 (1993), 120-133.

[10] J.P. Schmidt, All highest scoring paths in weighted grid graphs and its application to finding all approximate repeats in strings, *SIAM J. Computing* 27, 4 (1998), 972-992.

[11] J.S. Sim, C.S. Iliopoulos, K. Park, W.F. Smyth, Approximate periods of strings, *Theoretical Computer Science*, 262 (2001), 557-568.

[12] S. Kim, K. Park, A Dynamic edit distance table, *Proc. 11th Symp. Combinatorial Pattern Matching, LNCS* 1848 (2000), 60-68.

[13] G.M. Landau, E.W. Myers and J.P. Schmidt, Incremental string comparison, *SIAM J. Computing* 27, 2 (1998), 557-582.

[14] M. Middendorf, More on the complexity of common superstring and supersequence problems, *Theoretical Computer Science* 125, 2 (1994), 205-228.

[15] K.J. Räihä and E. Ukkonen, The shortest common supersequence problem over binary alphabet is NP-complete. *Theoretical Computer Science* 16 (1981), 187-198.



심 정 섭

1995년 서울대학교 컴퓨터공학과 학사. 1997년 서울대학교 컴퓨터공학과 석사. 1997년 ~ 현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 컴퓨터이론, Bioinformatics.



박 근 수

1983년 서울대학교 컴퓨터공학과 학사. 1985년 서울대학교 컴퓨터공학과 석사. 1991년 미국 Columbia University 전산학 박사. 1991년 ~ 1993년 영국 University of London, King's College 조교수. 1993년 ~ 현재 서울대학교 컴퓨터공학부 부교수. 관심분야는 컴퓨터이론, 암호학, 병렬계산.



김 성 렬

1993년 서울대학교 컴퓨터공학과 학사. 1995년 서울대학교 컴퓨터공학과 석사. 2000년 서울대학교 컴퓨터공학부 박사. 2001년 ~ 현재 주식회사 와이즈넛 근무. 관심분야는 컴퓨터이론, 암호학, 병렬계산.



이 지 수

1976년 서울대학교 전기공학과 학사. 1983년 서울대학교 컴퓨터공학과 석사. 1986년 서울대학교 컴퓨터공학과 박사수료. 1976년 ~ 1981년 대한전선, OPC 근무. 1981년 ~ 1985년 동양공업전문대학 교수. 1987년 ~ 현재 한국 방송통신대학 전산학과 교수. 관심분야는 알고리즘 설계 및 분석, 프로그래밍 언어론.