전도와 전치 연산을 사용하여 정렬하는 빠르고 간단한 알고리즘

(Fast and Simple Algorithm for Sorting by Reversals and Transpositions)

한 보 형 † 박 근 수 ††

(Bo-hyung Han)(Kunsoo Park)

요 약 최근 들어 계산분자생물학 분야에서 유전자 재배열 문제가 많은 관심을 끌고 있다. 특히 이러한 문제에는 전도(reversal)나 전치(transposition)와 같은 재배열 연산들이 사용되고 있다. 전도와 전치 두 가지 연산을 모두 사용하는 정렬은 필요한 최소 연산 회수의 2배 이내의 연산 수행만으로 가능하다고 알려져 있다. 이 논문에서는 기존의 알고리즘을 분석하고 휴리스틱을 사용함으로써 실제 연산 수행 회수를 대폭 줄일 수 있음을 보였다. 또한, 기존의 알고리즘보다 간단한 새로운 알고리즘을 제시하고, 이 알고리즘과 휴리스틱을 같이 사용하는 경우 수행 시간과 실험적 근사비(approximation ratio)에 있어서 매우 효과적임을 보였다.

Abstract Recently, genome rearrangement problems attract attention in computational molecular biology. Especially, rearrangement operations such as reversals and transpositions are widely used for those problems. The best previous algorithm for sorting by reversals and transpositions is a 2 approximation algorithm. In this paper, we show that we can reduce the number of operations by adding a heuristic to the previous algorithm. In addition, we propose a fast and simple algorithm and show that it is very efficient in the running time and the expected approximation ratio if it is combined with the heuristic.

1. 서 론

계산분자생물학에서의 유전자 서열(sequence) 비교문제는 생물의 발생 및 진화관계 규명에 있어서 매우 중요하다. 1980년대 후반에 Palmer와 Herbon은 Brassica oleracea (cabbage)와 Brassica campestris (turnip)의 미토콘드리아 게놈의 배열 상태가 매우 유사함을 발견하였다 [9] . 또한, 그들은 이 두 식물의 유전자 구성은 매우 유사하지만 유전자 정렬 순서가 다르다는 것을 알아냈다.

유전자 재배열 문제에 대한 관심이 증가함에 따라 유전자를 문자열로 모델링하여 유전자 재배열 및 정렬 문제를 해결하려는 노력이 계속되었다. 그러나, 이러한 유전자 재배열 문제를 삽입, 삭제, 및 교체와 같은 지역적인(local) 연산만을 사용하는 것은 문제 해결에 적합하지 않다는 것이 알려졌다 [4, 7]. 따라서, 최근에는 전도와 전치와 같은 전역적인(global) 연산을 수행함으로써 유전자를 재배치하여 정렬하는 알고리즘이 많이 연구되고 있다.

본 논문에서는 이전까지 알려진 전도와 전치 연산을 사용하여 유전자를 재배치하고 정렬하는 알고리즘을 분석하고 이 알고리즘에 휴리스틱을 추가하여 사용하는 경우에 보다 좋은 결과를 얻을 수 있음을 보인다. 또한, 이전의 알고리즘보다 간단한 알고리즘을 제시하고 이 알고리즘과 휴리스틱을 같이 사용할 경우 필요한 연산의 회수가 크게 감소하는 것을 이용하여 수행 시간과 근사비가 향상될 수 있음을 보인다.

2. 배경지식

2.1 용어 정의

† 학생회원 : Dept. of Computer Science, University of Maryland
bhhan@cs.umd.edu
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
kpark@theory.snu.ac.kr
논문접수 : 2000년 5월 2일
심사완료 : 2000년 8월 16일

$\pi=(\pi_1\pi_2\cdots\pi_n)$를 $\{1,2,\cdots,n\}$의 순열이라고 정의하고, 이 순열에 대해 다음과 같은 세 가지 연산을 정의한다.

전도 연산 $r(i,j)$는 $1\le i<j\le n+1$에 대해서 다음과 같이 정의된다.

$$\begin{pmatrix}1 & \ldots & i-1 & \mathbf{i} & \mathbf{i+1} & \ldots & \mathbf{j-1} & j & \ldots & n\\ 1 & \ldots & i-1 & \mathbf{j-1} & \ldots & \mathbf{i+1} & \mathbf{i} & j & \ldots & n\end{pmatrix}$$

$1\le i<j\le n+1$이고 $1\le k\le n+1$ ($k\notin[i,j]$)일 때, 전치 연산 $t(i,j,k)$은 다음과 같이 정의된다.

$$\begin{pmatrix}1 & \ldots & i-1 & \mathbf{i} & \mathbf{i+1} & \ldots & \mathbf{j-1} & j & \ldots & k-1 & k & \ldots & n\\ 1 & \ldots & i-1 & j & \ldots & k-1 & \mathbf{i} & \mathbf{i+1} & \ldots & \mathbf{j-1} & k & \ldots & n\end{pmatrix}$$

마지막으로, $1\le i<j\le n+1$이고 $1\le k\le n+1$ ($k\notin[i,j]$)일 때, 전도-전치 연산 $rt(i,j,k)$은 다음과 같이 정의된다.

$$\begin{pmatrix}1 & \ldots & i-1 & \mathbf{i} & \mathbf{i+1} & \ldots & \mathbf{j-1} & j & \ldots & k-1 & k & \ldots & n\\ 1 & \ldots & i-1 & j & \ldots & k-1 & \mathbf{j-1} & \ldots & \mathbf{i+1} & \mathbf{i} & k & \ldots & n\end{pmatrix}$$

임의의 두 개의 순열 $\pi=(\pi_1\pi_2\cdots\pi_n)$, $\sigma=(\sigma_1\sigma_2\cdots\sigma_n)$과 연산 $\rho_1,\cdots,\rho_t$에 대하여 $\pi\cdot\rho_1\cdot\rho_2\cdots\rho_t=\sigma$의 관계가 성립할 때, 두 순열사이의 전도 및 전치 거리는 연산의 개수 t의 최소값을 의미한다. 이 때, $I=(12\cdots n)$으로 정의하면 $\sigma^{-1}\pi\cdot\rho_1\cdot\rho_2\cdots\rho_t=I$이므로 π와 σ사이의 거리는 $\sigma^{-1}\pi$와 I사이의 거리와 같다. 따라서, 임의의 두 순열 사이의 전도 및 전치 거리를 구하는 문제는 하나의 순열과 정렬된 순열 I 사이의 전도 및 전치 거리 $d(\pi)$를 구하는 문제로 바꾸어 생각할 수 있다.

이 논문에서는 부호가 있는 순열을 다루게 되는데 이는 $(+1-5+4-3+2)$와 같이 순열의 모든 원소에 $+$ 혹은 $-$의 부호가 붙은 것을 말한다. 이와 같이 부호가 있는 순열에 대하여 전도와 전치 연산을 수행하여 순열을 정렬해야 한다. 부호가 있는 순열에 대한 연산은 그림 1과 같이 전도 연산이 적용되는 부분의 부호가 바뀐다는 점에서 부호가 없는 순열에 대한 연산과 다르다. 전치 연산은 부호를 바꾸지 않고, 전도-전치 연산이 사용되는 경우에는 부호가 바뀐다. 이러한 부호가 있는 순열의 전도와 전치 연산을 이용한 정렬은 이 연산들을 수행하여 임의의 부호가 있는 순열 π를 부호가 있는 정렬된 순열 $I=(+1+2+\cdots+n)$로 바꾸는 문제이다. 그림 1은 전도 연산만을 사용하여 부호가 있는 순열을 정렬한 예이다.

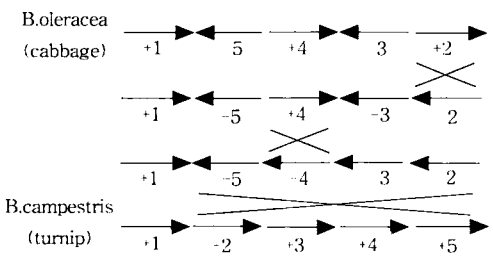


그림 1 전도 연산만을 사용한 부호가 있는 순열의 정렬

2.2 단절점 그래프

Bafna와 Pevzner[1]는 전도 연산만을 이용하여 순열을 정렬하는 문제를 해결하기 위해 단절점 그래프(breakpoint graph)를 사용하였는데 이 논문에서도 같은 자료구조를 사용하게 될 것이다. 단절점 그래프는 다음과 같은 방법으로 만들어진다.

π가 부호가 없는 순열이라고 하자. $\pi=(\pi_1\pi_2\cdots\pi_n)$의 각 원소들을 입력로 나열하여 단절점 그래프의 노드를 만들고 여기에 $\pi_0=0$, $\pi_{n+1}=n+1$의 두 개의 노드를 양쪽 끝에 추가한다. 이 때, $|i-j|=1$이면 $i\sim j$라고 표시하는데, 연속으로 나열된 노드 π_i와 π_{i+1}에 대해 $\pi_i\sim\pi_{i+1}$이면 두 노드는 인접한다고 하고, 그렇지 않으면 그 사이를 단절점이라고 말한다. 단절점, 즉 $\pi_i\neq\pi_{i+1}$인 두 노드 사이는 검은 간선으로 연결하고, $\pi_i\sim\pi_j$이면서 $i\neq j$인 두 노드 π_i와 π_j는 회색 간선으로 연결한다. 이 논문에서는 검은 간선을 실선으로, 회색 간선을 점선으로 표시한다.

부호가 있는 순열에 대해서는 하나의 노드를 두 개로 나누어 부호를 표현한다. 즉, $+i$ 노드는 $(2i-1,2i)$로, $-i$ 노드는 $(2i,2i-1)$로 각각 나누고, 위에서 설명한 규칙대로 간선을 만들어 주면 된다. 그림 2는 $\pi=(+1-5+4-3+2)$의 단절점 그래프이다. 이런 방법으로 단절점 그래프를 그릴 경우, I는 간선이 없는 그래프가 된다. 이러한 부호가 있는 순열에 대한 단절점 그래프는 다음과 같은 성질을 갖는다.

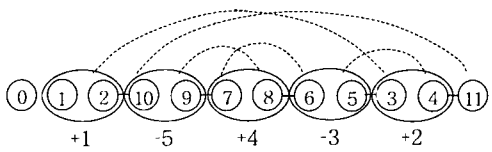


그림 2 순열 $\pi=(+1-5+4-3+2)$의 단절점 그래프 $G(\pi)$

보조정리 1 [4] 부호가 있는 순열 π에 대한 단절점 그래프 $G(\pi)$는,

1. 각 노드의 회색 간선과 검은 간선 차수는 같고 항상 0 또는 1이다.
2. 이 단절점 그래프내의 사이클은 검은 간선과 회색 간선이 교대로 반복되는 사이클이다.
3. 각 사이클은 적어도 2개의 회색(또는 검은색) 간선을 가지고 있다.

3. 관련연구

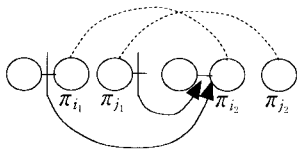
부호가 있는 순열을 전도 연산만을 사용하여 정렬하는 문제에 대한 연구는 비교적 활발히 진행되었다. Hannenhalli와 Pevzner[5]는 가장 먼저 이 문제를 다항시간 내에 해결하는 $O(n^4)$ 알고리즘을 발표하였다. 또한, Berman과 Hannenhalli[3]는 위의 알고리즘과 비교하여 시간복잡도가 개선된 $O(n^2\alpha(n))$ 알고리즘을 제시하였고, Kaplan, Shamir와 Tarjan[6]은 Berman과 Hannenhalli의 알고리즘보다 더 간단하고 빠른 $O(n^2)$ 알고리즘을 발표하였다. 여기서 $\alpha(n)$은 Ackerman 함수의 역함수이다

Bafna와 Pevzner[2]는 전치 연산만을 사용하여 주어진 순열을 정렬하는 1.5 근사 알고리즘을 발표하였다. 한편, Gu, Peng과 Sudborough[4]는 부호가 있는 두 순열 사이의 거리를 구하는데 있어서 전도와 전치 연산을 모두 사용하는 2 근사 알고리즘을 발표하였다. 그들은 두 순열 π와 I와의 거리의 하한값이 $(b(\pi)-c(\pi))/2$ 이상임을 보였다. 여기서 $b(\pi)$는 단절점의 개수를 $c(\pi)$는 단절점 그래프에서 길이가 홀수인 사이클의 개수를 의미한다. 사이클의 길이는 사이클에 속한 검은 간선의 개수를 의미한다.

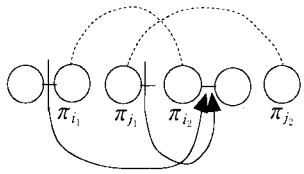
Gu-Peng-Sudborough[4]의 2 근사 알고리즘은 다음과 같다. 지금부터 언급하는 단절점 그래프는 모두 부호가 있는 순열에 대한 단절점 그래프를 의미한다. 단절점 그래프 내에 두 회색간선 (π_{i_1},π_{i_2})와 (π_{j_1},π_{j_2})가 있을 때, $i_1<j_1<i_2<j_2$ 또는 $j_1<i_1<j_2<i_2$이면 두 간선은 교차한다고 말한다.

보조정리 2 [4] 사이클 C가 교차하는 회색 간선을 가지고 있다면 다른 사이클의 길이에는 영향을 미치지 않으면서 사이클 C의 길이를 하나 이상 줄일 수 있는 전도 및 전치 연산이 항상 존재한다. (그림 3)

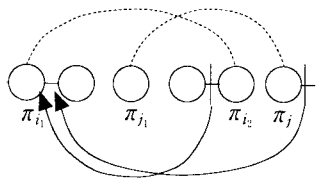
그림 3의 case 1과 case 4는 전도-전치 연산을, case 2와 case 3은 전치 연산만을 사용한 경우로서 네 경우 모두 π_{i_1}과 π_{i_2}가 연산 후에 인접하게 되어 단절점이 하나 제거된다.



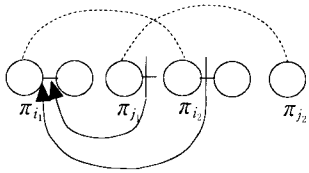
(a) case 1



(b) case 2



(c) case 3



(d) case 4

그림 3 같은 사이클 내의 교차하는 회색 간선을 이용한 사이클의 길이를 줄이는 방법

두 검은 간선 $e_1=(\pi_{i_1},\pi_{i_2})$와 $e_2=(\pi_{j_1},\pi_{j_2})$에 대하여 $\max\{i_1,i_2\}<\min\{j_1,j_2\}$이면 $e_1<e_2$라고 한다. 또한, 사이클 C의 검은 간선 e_1,e_2와 사이클 C'의 검은 간선 e'_1,e'_2이 있을 때, 그림 4와 같이 $e_1<e'_1<e_2<e'_2$ 또는 $e'_1<e_1<e'_2<e_2$의 관계가 성립하면 두 사이클이 중첩된다고 한다.

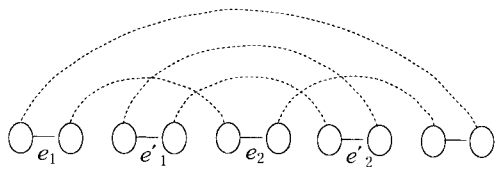
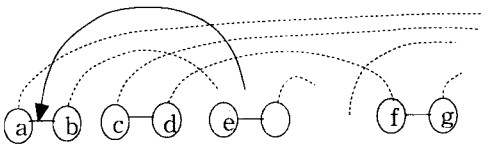


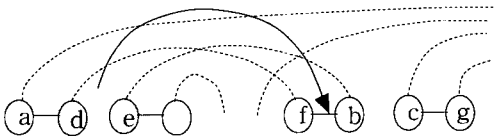
그림 4 중첩된 사이클의 예

보조정리 3 [4] 정렬되지 않은 순열에 대한 단절점 그래프 $G(\pi)$ 내에 교차하는 회색간선을 가진 사이클이 하나도 존재하지 않을 때, 이 그래프 내에는 중첩되는 두 사이클 C와 C'이 항상 존재한다.

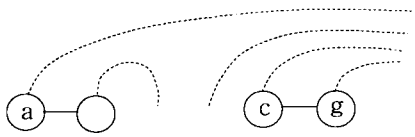
보조정리 4 [4] 두 사이클 C와 C'이 교차하는 회색간선이 존재하지 않는 두 중첩되는 사이클일 때 두 사이클의 길이를 각각 1이상씩 감소시킬 수 있는 연속적인 두 번의 연산이 항상 존재한다. (그림 5) 이 때, $|C| = |C'| = 2$이면 이 두 번의 연산으로 두 사이클이 모두 제거된다.



(a) 첫 번째 연산



(b) 두 번째 연산



(c) 연속된 두 번의 연산 수행결과

그림 5 중첩된 두 사이클의 단절점 제거

그림 5의 첫 번째와 두 번째 연산 모두 전치 연산만을 사용한 것이다. 첫 번째 연산은 두 개의 중첩된 사이클을 하나의 사이클로 연결하지만 단절점의 개수를 줄이지 않고, 두 번째 연산에서 단절점의 개수를 2개 줄임으로써 두 번의 연산으로 두 개의 단절점을 제거한다.

위의 세 개의 보조정리에 따라 Gu-Peng-Sudborough[4]이 구성한 부호가 있는 순열 π를 정렬하는 알고리즘 SORT는 다음과 같다.

```
Algorithm SORT(π)
begin
단절점 그래프 G(π)구성, C_1, ..., C_r은 G(π)내의 사이클
while ( ∃C_i ) do {
    while ( C_i에 교차하는 회색간선 존재) do
        { C_i의 길이를 줄이는 연산 수행}
    if (중첩된 사이클 C와 C' 가 존재) then
        { C와 C' 의 길이를 1씩 줄일 수
          있는 연속된 두 개의 연산 수행}
}
end
```

그림 6 부호가 있는 순열을 정렬하는 2 근사 알고리즘

$r(\pi)$를 순열 π에 존재하는 사이클의 개수라고 하자. 보조정리 2-4에 의하면 주어진 순열을 정렬하기 위해서는 $b(\pi) - r(\pi)$번 이하의 연산이 필요함을 알 수 있다 [4]. 이 때, $r(\pi) \geq c(\pi)$이므로

$$b(\pi) - c(\pi) \geq b(\pi) - r(\pi)$$

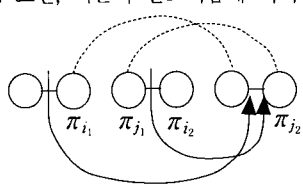
이고, 필요한 연산 회수의 하한값이 $(b(\pi) - c(\pi))/2$이므로 다음 정리가 성립한다.

정리 1 [4] 알고리즘 SORT는 $O(n^2)$의 시간복잡도를 갖는 2 근사 알고리즘이다.

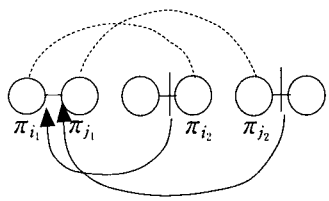
4. 개선된 알고리즘

4.1 휴리스틱 사용

이 논문에서는 단절점 그래프에서 한 번의 연산으로 단절점을 2개 이상씩 줄일 수 있는 경우가 빈번히 발생하는 것에 착안하여 이러한 연산을 먼저 수행하는 휴리스틱을 사용하였다. 그림 7은 그림 3에서 보인 경우 가운데 특수한 두 가지 경우를 보인 것이다. 이 두 가지 경우 모두 단절점을 2개 이상 줄일 수 있기 때문에 이러한 연산을 먼저 수행시키면 정렬에 필요한 거리를 줄일 수 있다. 또한, 기존의 알고리즘에 이러한 연산을 먼



(a) case 1



(b) case 2

그림 7 단절점을 2개 이상 제거할 수 있는 연산

저 수행시키는 부분을 추가하여도 알고리즘의 시간복잡도에는 영향을 미치지 않는다.

4.2 간소화된 알고리즘

기존의 알고리즘은 같은 사이클 내에 교차하는 회색 간선이 있는 경우에만 그림 3과 같은 연산들을 취하도록 하였다. 그러나, 간소화된 알고리즘에서는 같은 사이클 내에 있는 회색 간선들뿐만 아니라 다른 사이클에 있는 회색 간선들끼리도 서로 교차하기만 하면 단절점의 개수를 항상 줄일 수 있다는 사실을 이용한다.

보조정리 5 주어진 순열 π에 대한 단절점 그래프 $G(\pi)$가 있을 때, 이 순열이 완전히 정렬되지 않았다면 $G(\pi)$에는 항상 교차하는 회색 간선이 존재한다.

증명 보조정리 3에서 설명한 바와 같이 단절점 그래프에는 교차하는 회색 간선을 가지고 있는 사이클이 적어도 하나 존재하거나, 그렇지 않으면 사이클들이 서로 중첩되어 있다.

사이클 C와 C'이 중첩되어 있을 때, 각 사이클의 검은 간선들이 번갈아 가며 그래프 상에 나타나기 때문에 중첩된 두 사이클의 회색 간선들 중에는 서로 교차하는 회색 간선이 적어도 한 쌍 존재한다. □

따름정리 5.1 단절점 그래프 $G(\pi)$에서 단절점을 하나 이상 줄일 수 있는 연산은 항상 존재한다.

증명 그림 3은 교차하는 회색 간선이 있는 경우 단절점을 어떻게 줄일 수 있는지 보여주고 있다. 이 그림은 교차하는 회색 간선이 같은 사이클 내에 있는 경우뿐만 아니라 일반적인 경우에도 적용된다. 따라서, 회색 간선이 같은 사이클 내에 있지 않아도 서로 교차하기만 하면 그림 3과 같은 연산으로 단절점의 개수를 줄일 수 있다. 또한, 보조정리 5가 성립하므로 단절점 그래프에서 단절점을 하나 이상 줄일 수 있는 연산은 항상 존재한다. □

이 알고리즘은 우선 휴리스틱을 적용할 수 있는지 먼저 알아보고, 가능한 한꺼번에 2개 이상의 검은 간선을 제거할 수 있는 연산을 먼저 수행한다. 그러다가 휴리스틱을 더 이상 적용할 수 없는 경우, 임의의 교차하는 회색 간선을 선택하여 검은 간선을 제거한 후에 다시 휴리스틱이 적용되는지 확인하는 과정을 반복하게 된다.

기존의 알고리즘은 한 사이클 내에 교차하는 회색 간선이 없는 경우 서로 중첩되는 사이클 C와 C'을 찾아내어 그림 4와 같은 연속적인 두 번의 연산을 수행하였다. 이 두 번의 연산 중 첫 번째 연산은 두 번째 연산에서 두 개의 검은 간선을 한꺼번에 제거할 수 있는 경우를 만들기 위한 연산에 불과하다. 실제로 그림 5의 (b)를 보면 그림 7의 (a)와 같은 경우임을 알 수 있다. 그러나, 이 알고리즘에서는 항상 교차하는 회색 간선이 그래프 내에 존재하기 때문에 이러한 경우를 고려할 필요가 없이 각 연산마다 1개 이상의 단절점을 제거할 수 있도록 하여 알고리즘이 매우 간단하다.

Algorithm SIMPLE_SORT(π)
begin
단절점 그래프 $G(\pi)$구성, $C_1, \cdots, C_r$은 $G(\pi)$내의 사이클
while ($\exists C_i$) **do** {
    while { $G(\pi)$에 단절점을 두 개 이상 제거하는 경우 존재}**do**
        {단절점을 2개 이상 줄이는 연산 수행}
    if ($G(\pi)$에 교차하는 회색간선 존재) **then**
        {단절점 개수를 줄이는 연산 수행}
}
end

그림 8 부호가 있는 순열을 정렬하는 간단한 2 근사 알고리즘

휴리스틱이 추가된 간소화된 알고리즘 SIMPLE_SORT는 위의 그림 8과 같다. 이 알고리즘은 위에서 언급한 바와 같이 단절점의 개수를 2개 이상 줄일 수 있는 경우가 빈번히 일어나기 때문에 매우 효율적이다.

정리 2 알고리즘 SIMPLE_SORT는 부호가 있는 순열 π를 전도 및 전치를 이용하여 정렬하는 2 근사 알고리즘이며 $O(n^2)$의 시간복잡도를 갖는다.

증명 보조정리 5, 따름정리 5.1과 [4]에 의해서 $b(\pi) - r(\pi)$번 내의 전도 및 전치 연산을 통해 주어진 순열을 정렬할 수 있다. 또한, 휴리스틱이 적용되는 경우를 찾는데 $O(n)$ 시간이 걸리고 실제 연산을 하는데도 역시 $O(n)$ 시간이 걸리므로, 이 알고리즘은 기존의 알고리즘 SORT와 같은 $O(n^2)$의 시간복잡도를 갖는다. □

5. 실험 결과

표 1은 Gu-Peng-Sudborough[4]가 제시한 2 근사 알고리즘 SORT와 여기에 휴리스틱을 추가한 알고리즘, 이 휴리스틱을 추가한 간소화된 알고리즘 SIMPLE_SORT를 구현하여 비교해본 결과이다. 표 1에서 구한 근사비는 정렬에 필요한 연산 회수의 하한값과 실제 알고리즘을 수행시켰을 때 사용된 연산 회수의 평균비이다. 이 실험에서는 임의로 생성된 데이타를 세 가지 알고리즘에 적용하여 각각의 실험적 근사비, 근사

비의 표준편차와 수행시간을 구하였다. 이 실험의 결과는 입력크기가 800까지는 100개, 1600은 30개, 3200일 때는 10개의 데이타를 이용하여 평균을 구한 것이다. 입력의 크기가 커질수록 편차가 작아지기 때문에 데이타의 개수가 적어도 실험결과에 큰 영향을 미치지 않는다.

표 1 각 알고리즘의 근사비와 수행 시간 비교

입력 크기	기존 알고리즘 (SORT)			휴리스틱 사용 (SORT + heuristic)			간소화된 알고리즘 (SIMPLE_SORT)		
	근사비	표준편차	시간(s)	근사비	표준편차	시간(s)	근사비	표준편차	시간(s)
100	1.440	0.098	0.12	1.240	0.049	0.09	1.280	0.067	0.06
200	1.531	0.031	0.99	1.204	0.061	0.48	1.122	0.064	0.35
400	1.556	0.047	8.08	1.172	0.029	3.44	1.184	0.022	2.83
800	1.515	0.025	61.66	1.051	0.016	20.97	1.101	0.011	18.99
1600	1.561	0.010	534.15	1.025	0.006	165.71	1.045	0.006	158.15
3200	1.549	0.005	4328.31	1.021	0.002	1344.94	1.026	0.003	1295.72

기존의 알고리즘을 구현하여 실험해본 결과 이론적 근사비보다 훨씬 적은 약 1.5 정도의 근사비를 얻을 수 있었다. 이는 휴리스틱을 적용하기 위해서 특별한 고려를 하지 않더라도 우연히 이 휴리스틱이 적용되는 경우가 많이 발생했기 때문이다. 그러므로, 거리를 2 이상 줄일 수 있는 경우가 빈번히 발생한다는 것을 알 수 있다.

실험 결과 주어진 순열을 정렬할 때 휴리스틱을 추가하면 전도 및 전치 거리가 매우 많이 줄어들었음을 알 수 있다. 또한, 입력의 크기가 커질수록 근사비가 점점 감소하였고, 수행시간도 많이 줄일 수 있었다. 따라서, 기존의 알고리즘 SORT에 휴리스틱을 추가한 알고리즘은 매우 효율적이다.

또한, 휴리스틱이 추가된 간소화된 알고리즘을 사용한 경우 근사비가 기존의 알고리즘에 휴리스틱을 추가하여 사용한 경우와 거의 비슷하였다. 이 알고리즘은 단절점 그래프의 사이클 정보가 필요 없기 때문에 수행시간이 다른 알고리즘에 비해 빠르다는 장점이 있다. 또한, 이 알고리즘도 입력의 크기가 커질수록 근사비가 감소하였다. 따라서, 이 알고리즘은 실험적으로 매우 빠르고, 매우 적은 연산 수행만으로 주어진 순열을 정렬하는 알고리즘이다.

휴리스틱을 사용한 경우 입력의 크기가 커질수록 근사비와 근사비의 표준편차가 점점 작아진다. 이는 입력의 크기가 크면 클수록 휴리스틱을 추가한 알고리즘이 보다 효율적일 뿐만 아니라, 실험적으로 거의 일정한 근사비를 보장할 수 있음을 나타낸다.

6. 결론 및 향후 과제

지금까지 전도 및 전치 연산을 사용하여 주어진 순열을 정렬하는 알고리즘을 분석하고 보다 개선된 알고리즘들을 제시하였다. 휴리스틱을 사용함으로써 기존의 알고리즘이 매우 빠른 속도로 수행될 수 있음을 실험을 통하여 보였고, 기존의 알고리즘에는 고려하지 않았던 사이클간 연산을 수행함으로써 알고리즘이 간단해지고 실험적 수행시간도 빨라졌다.

지금까지는 전도 및 전치 연산을 사용한 정렬 알고리즘은 2 근사 알고리즘이 가장 좋은 것으로 알려져 있으나, 근사비를 더 줄일 수 있는 알고리즘을 개발하는 것이 앞으로의 과제이다.

참 고 문 헌

[1] V. Bafna, P. Pevzner, Genome rearrangements and sorting by reversals, SIAM J. Comput. 25 (2), pp. 272-289, 1996

[2] V. Bafna, P. Pevzner, Sorting by transpositions, SIAM J. Discrete Math, pp. 224-240, 1998

[3] P. Berman and S. Hannenhalli, Fast sorting by reversals, In Proc. Combinatorial Pattern Matching(CPM'96), pp. 168-185, 1996

[4] Q. Gu, S. Peng, H. Sudborough, A 2-approximation algorithm for genome rearrangements by reversals and transpositions, Theoretical Computer Science, pp. 327-339, 1999

[5] S. Hannenhalli, P. Pevzner, Transforming cabbage into turnip (polynomial algorithm for sorting signed permutation by reversals), Proc. 27th ACM symp. on Theory of Computing(STOC'95), pp. 178-189, 1995

[6] H. Kaplan, R. Shamir, R. Tarjan, Faster and simpler algorithm for sorting signed permutation by reversals, In Proc. ACM-SIAM Symp. on Discrete Algorithms (SODA'97), pp. 344-351, 1997, also: Proc. RECOMB'97

[7] S. Karlin, E.S. Mocarski, G.A. Schachtel, Molecular evolution of herpesviruses: genomic and protein sequence comparisions, J. Virol. 68, pp. 1886-1902, 1994

[8] J. Kececioglu, D. Sankoff, Exact and approximation algorithms for the inversion distance between two permutations, Algorithmica 13, pp. 180-210, 1995

[9] J.D. Palmer, L.A. Herbon, Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence, J. Mol. Evol. 27, pp. 87-97, 1988

한 보 형

1993년 3월 ~ 1997년 2월 서울대학교 컴퓨터공학과 학사. 1997년 3월 ~ 2000년 8월 서울대학교 컴퓨터공학부 석사. 2000년 8월 ~ 현재 University of Maryland at College Park 박사과정. 관심분야는 스트링 알고리즘, 근사알고리즘, 바이오인포매틱스

박 근 수

1983년 서울대학교 컴퓨터공학과 학사. 1985년 서울대학교 컴퓨터공학과 석사. 1991년 미국 Columbia 대학교 전산학 박사. 1991년 11월 ~ 1993년 8월 영국 런던대학교 King's College 조교수. 1995년 7월 ~1995년 8월 호주 Curtin 대학교 방문연구원. 1993년 8월 ~ 현재 서울대학교 컴퓨터공학부 부교수. 관심분야는 컴퓨터 이론, 병렬 계산, 암호학