# Suffix Tree를 이용한 DNA Sequences의 Multiple Alignment에 관한 연구

한상일*, 이성근, 안대명, 황규석
부산대학교 화학공학과
(1three1@hanmail.net*)

# A Study on Multiple Alignment of DNA Sequences with Suffix Tree

Sang Il Han*, Sung Keun Lee, Dae Myung An, Kyu Suk Hwang
Department of Chemical Engineering, Pusan National University
(1three1@hanmail.net*)

## 서론

생명공학이 발달함에 따라 생물체의 기반이 되는 DNA, RNA 및 단백질을 분석하는 기술도 발전을 이루었고, 그에 따른 유전정보들이 빠른 속도로 축적되었다. 실험에 의해 밝혀진 서열 정보들은 텍스트 형태로 컴퓨터에 저장되고, 따라서 컴퓨터를 이용한 서열정보의 비교, 분석이 아주 광범위하게 수행되고 있다. 또한 생명과학 분야에서는 이러한 서열정보를 이용해서 생물체의 유전적 진화 관계나 유전자의 기능 등을 밝히려고 시도하고 있다. 전체 유전자(Whole genome)의 서열정보는 대용량 텍스트이며, 순차적으로 컴퓨터에 저장된다. 이러한 저장방법은 대용량 서열정보를 처리하는데 있어서 효율적이지 못하다. 또한 최근 서열 데이터가 빠르게 증가함에 따라, 데이터 엑세스 시간이 서열정보를 처리하는데 제한 요소가 되어가고 있다. 이런 이유 때문에 효율적인 자료구조와 알고리즘이 필요하게 되었다.

문자열을 비교하는 방법에는 두 개를 비교하는 pair-wise alignment 와 여러 개의 서열을 동시에 비교하는 multiple alignment 방법이 있다. 본 연구에서는 여러 개의 서열을 동시에 비교하고 위에서 제시한 문제점들을 해결하기 위해 문자열에 관련된 문제를 효율적으로 해결하는 자료 구조인 서픽스 트리(Suffix Tree)와 클러스터링 방법인 STC(Suffix Tree Clustering)을 도입하였다. 서픽스 트리는 String의 내부구조(Internal Structure)를 표현하는 자료 구조이며, STC는 서픽스 트리를 이용한 클러스터링 방법이다. 서열들 사이의 공통된 염기 서열(Common Subsequence)을 효율적으로 찾아내기 위해 Suffix Tree를 형성한다. 검색된 공통 염기 서열들을 바탕으로 클러스터들이 형성된다.

또한 염기 서열들은 특정 염기가 계속해서 반복되는 부분(-AAAAAAAAAA-)과 특정 염기 단위(-ATCATCATCATC-)가 반복되는 부분을 종종 포함한다. 이러한 부분을 낮은 복잡도 지역(Low-Complexity Region)이라 하고, 의미 있는 유전정보를 가지는 지역이 아니다. 클러스터링 할 때, 낮은 복잡도 지역은 국소 유사성을 발생시켜서, Bad Basic Cluster들을 생성시킬 수 있다. 따라서 낮은 복잡도 지역을 스크리닝(Screening)하여서 Bad Basic Cluster를 제거하는 절차도 수행하였다.

## 본론

STC(Suffix Tree Clustering)는 ST를 이용하여 선형시간(Linear Time)으로 클러스터링 하는 알고리즘이다. 염기서열 데이터에서 공통된 염기(Subsequence)를 토대로 해서 클러스터링 하게 된다. STC는 Suffix Tree Construction, Common Subsequence Search, Basic Cluster Overlapping과 같은 절차를 가지고, 마지막으로 Bad Basic Cluster를 걸러내기 위해 Low-Complexity Region을 가지는 클러스터를 제거한다.

### 1. Suffix Tree Construction

Suffix Tree를 이용하여 Sequences의 동일한 부분을 정확하고, 신속하게 찾아낼 수 있다. Suffix Tree를 선형 시간으로 구축하는 알고리즘은 Weiner에 의해서 처음 제안되었고, McCreight, Ukkonen은 똑같은 수행시간에 공간을 더 작게 하는 방법을 제시하였다. 특히 Ukkonen의 알고리즘은 구현과 이해가 쉽고, 온라인에서 사용 가능한 선형 알고리즘으로써 대부분 Suffix Tree 구축에 사용된다.[1]

Suffix Tree는 다음과 같은 특징을 가진다.

- 루트(Root)와 방향성이 있는 Tree이다.

- 길이가 m인 sequence의 경우, 1부터 m까지의 가지를 가진다.
- 각각의 내부 노드(node)는 루트 이외의 두 개 이상의 자식 노드를 가진다.
- 가지 위에, Subsequence가 Label로 표시된다.
- 같은 노드에서 나오는 가지들은 다른 Label을 가진다.
- Nodes의 Labels은 Sequence의 Suffix들이다.

**2. Common Subsequence Search**

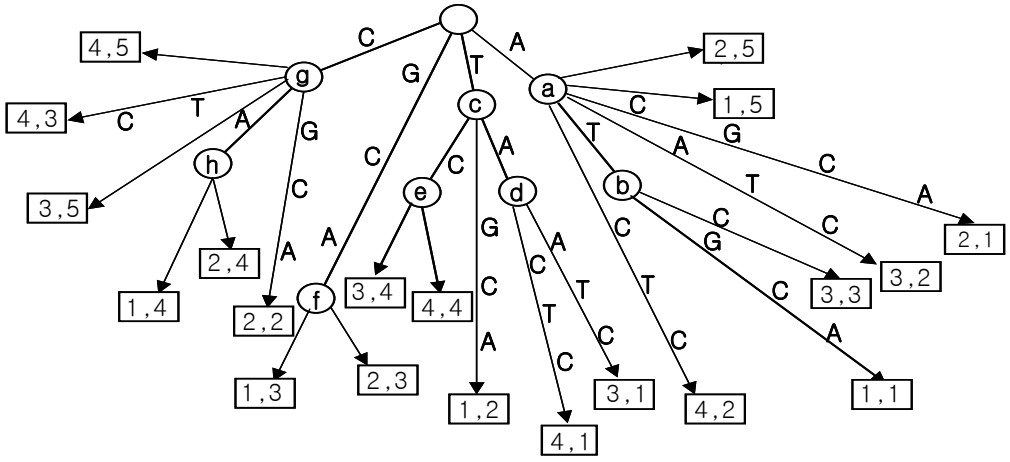


Fig.1 Suffix Tree for Sequences(1.ATGCA 2.ACGCA 3.TAATC 4.TACTC)

Fig.1는 4개의 Sequences 1. ATGCA, 2. ACGCA, 3. TAATC, 4. TACTC을 Suffix Tree로 구축한 예이다. Suffix Tree에서 원은 Suffix Tree의 Node이며, 맨 끝 노드는 각각의 사각형 박스를 가진다. 이 사각형 박스에 의해서 Subsequences가 Sequecnes의 어떤 위치에서 시작되는지 알 수 있게된다. 즉 사각형 안의 첫 번째 숫자에 의해서 각각의 Sequence를 식별할 수 있고, 두 번째 숫자에 의해서 Subsequence의 시작 위치를 찾을 수 있게된다. 각각의 Nodes와 가지(Node와 Node사이)위의 Label은 Sequences와 그 Sequences에 포함하고 있는 Common Subsequences(공통 염기)를 나타내고 있다. 따라서, Suffix Tree의 Nodes를 검색하여서 Common Subsequences와 Sequence Group을 구할 수 있으며, 이 Nodes가 Basic Clusters가 된다. Fig.1의 Suffix Tree로부터 Node, Common-Subsequences, Index of Subsequences, Sequences를 구하여 Table로 만들었다. Index of Subsequences는 Subsequence가 속한 Sequence와 그 Sequence에서의 시작 위치를 나타낸다.

염기서열정보에는 특정 염기가 계속해서 반복되는 부분(-AAAAAAAAAAAA-)과 특정 염기단위(-ATCATCATCATC-)가 반복되는 부분을 종종 포함하고 있다. 이러한 부분을 낮은 복잡도 지역(Low-Complexity Region)이라 한다. 낮은 복잡도 지역에 속한 염기단위가Basic Cluster가 되는 것을 차단하기 위해서 Basic Cluster에 점수를 할당하고, 할당된 점수에 의해서 Bad Basic Cluster를 제거할 수 있다. 또한 빈도수가 적은 Basic Cluster들도 이러한 방법에 의해서 삭제된다. Basic Cluster는 크게 두 부분으로 이루어져 있다. (1) Subsequence (2) a set of Sequence

이 두 구성요소에 의해서 Basic Cluster에 점수가 할당된다.

$$S(B) = |SB| \cdot P|CS|$$

여기서 S(B)는 Basic Cluster의 점수이고, |SB|는 Basic Cluster에 있는 Sequence의 수, |CS|는 Subsequence을 구성하고 있는 문자수(염기)이다. P는 0에서 1의 값을 가진다. Subequence을 포함하는 Sequence의 수가 작거나, Subsequence가 반복되는 염기단위인 경우에 " 0 "의 값을 가지게 된다. 두 가지 경우 이외에는 " 1 "의 값을 가진다. 여기에서 AT repeats는 Low-complexity region이므로 P는 0의 값을 가지므로 두 개의 Basic Cluster가 삭제된다.

| Node | Subsequence | Index of Subequence | Number of Sequence |
|---|---|---|---|
| a | A | 1={(1,1),(2,1)}, 2={(3,2),(4,2)}<br>3={(3,3)}, 4={(1,5),(2,5)} | 1, 2, 3, 4 |
| b | AT | 1={(1,1)}, 2={(3,3)} | 1, 3 |
| c | T | 1={(3,1),(4,1)}, 2={(1,2)}<br>3={(3,4),(4,4)} | 1, 3, 4 |
| d | TA | 1={(3,1),(4,1)} | 3, 4 |
| e | TC | 1={(3,4),(4,4)} | 3, 4 |
| f | GCA | 1={(1,3),(2,3)} | 1, 2 |
| g | C | 1={(2,2)}, 2={(4,3)}<br>3={(1,4),(2,4)}, 4={(3,5),(4,5)} | 1, 2 ,3, 4 |
| h | CA | 1={(1,4),(2,4)} | 1, 2 |

Table 1. Basic Clusters(Nodes) for Suffix Tree in Figure 1

**3. Basic Cluster Overlapping**

Sequences은 하나 이상의 문자(염기)로 구성된 Subsequence를 포함 할 수 있으며, 그 결과로써 Basic Clusters가 겹치는(Overlap) 부분이 생성된다. 이를 피하기 위해서 Basic Clusters를 서로 결합한다. Basic Clusters를 결합하기 위해서, Basic Cluster사이에 두 가지 유사도(Similarity Measure)를 정의한다.

**(1) 염기순서유사도(Subsequence Order and Similarity Measure)**

각각 |CSm|와 |CSn|크기를 가진 Subsequence - CSm, CSn - 이 주어 졌을 때, |CSm∩CSn|은 두 Subsequence에 공통된 문자(염기)수를 나타낸다

$$|CSm \cap CSn| / |CSm| \geq 0.5 \text{ and } |CSm \cap CSn| / |CSn| \geq 0.5$$

위의 조건을 만족하는 경우 유사도 "1", 그렇지 않은 경우는 "0"이 된다. 유사도가 "1"인 경우 Basic Cluster가 결합하게 된다. 이때 염기순서도 함께 고려되어진다.

**(2) 문자열유사도(Sequence Similarity Measure)**

각각 |SBm|와 |SBn|크기를 가진 Basic Cluster에 있는 Sequence - SBm, SBn - 이 주어 졌을 때, |SBm∩SBn|은 두 Basic Cluster에 있는 동일한 Sequence의 수를 나타낸다.

$$|SBm \cap SBn| / |SBm| \geq 0.5 \text{ and } |SBm \cap SBn| / |SBn| \geq 0.5$$

위의 조건이 만족하는 경우 유사도 "1", 그렇지 않은 경우는 "0"이 된다. 유사도가 "1"인 경우 Basic Cluster가 결합하게 된다. Fig.2에 Basic Cluster Overlapping을 나타내었다. 먼저 염기순서유사도에 의해서 일렬로 된 Subsequence의 위치와 크기를 결정 할 수 있으며, 11개의 Basic Clusters가 4개의 BC(Basic Cluster) - BC1 = {(1, 2, 1, 1)}, BC2 = {(1, 2, 3, 3)}, BC3 = {(3, 4, 1, 2)}, BC4 = {(3, 4, 4, 2)} - 로 겹쳐진다. 4개의 BC는 문자열유사도에 의해서 2개의 Cluster - C1={(1, 2, 1, 1), (1, 2, 3, 3)}, C2={(3, 4, 1, 2), (3, 4, 4, 2 )} - 로 된다. 소괄호의 첫 번째 숫자와 두 번째 숫자는 Sequence Number, 세 번째 숫자는 Subsequence의 시작 위치, 네 번째 숫자는 Subsequence의 크기이다. Fig.2에서 알 수 있듯이 각각의 Cluster는 Basic Clusters의 연결임을 알 수 있다.

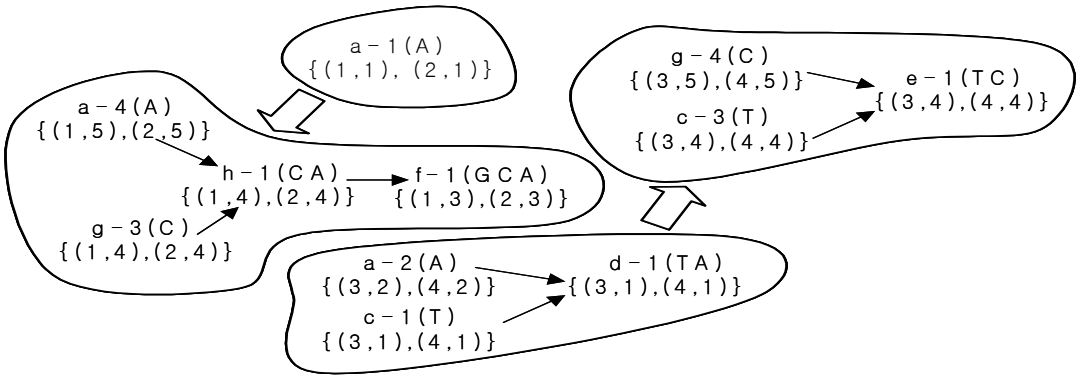


Fig.2 Basic Clusters Overlapping

Low-complexity region을 제거하기 위해서 AT repeats를 포함하는 클러스터를 제거하고 query sequences를 배열하면 다음과 같다.

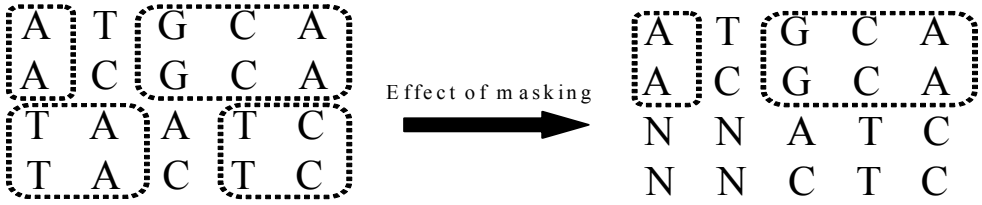


Fig.3 Cluster for Four Sequences

## 결론

STC(Suffix Tree Clustering)은 Suffix Tree를 기반으로 한 $O(n)$ Time 클러스터링 알고리즘으로써, 염기서열들을 효율적이고 신속하게 클러스터링할 수 있다. (1) 염기서열들을 Suffix Tree로 구축하고, (2) 구축된 Suffix Tree를 통하여 Common Subsequence를 검색한 후, Bad Basic Clusters를 제거한다. (3) 최종적으로, Basic Clusters를 결합함으로써 염기서열들을 클러스터링 할 수 있다. 이러한 절차에 의해서, 4개의 염기서열이 2개로 클러스터링되어짐을 알 수 있다. C1={(1, 2, 1, 1), (1, 2, 3, 3)}, C2={(3, 4, 1, 2), (3, 4, 4, 2 )}
그리고 Low-complexity region을 포함하는 클러스터를 제거하면 클러스터 C1={(1, 2, 1, 1), (1, 2, 3, 3)}만 남게되어 Fig.3 과 같이 된다.

1. Dan Gusfield "Algorithms on Stings, Trees, and Sequences", Cambridge, 1997
2. Arthur L. Delcher, Simon Kasif, Robert D. Fleischmann "Alignment of whole genomes", Nucleic Acids Research, 1999, Vol.27, No.11
3. Oren Zamir and Oren Etzioni, "Web Document Clustering : A Feasibility Demonstration", ACM,1998
4. Cyntbia Gibas and Per Jambeck "Developing Bioinformatics Computer Skills", O'Reilly
5. David W.Mount, "Bioinformatics : Sequence and Genome Analysis", Cold Spring Harbor Laboratory Press
6. A.Floratos, I. Rigoutsos, L. Parida, Y. Gao "DELPHI : A pattern-based method for detection sequence similarity", IBM J. RES & DEV. VOL 45, 200
7. E. Ukkonen, "On-Line Construction of Suffix Trees", Algorithmica(1995) 14: 249-260
8. Lipman, D. J. and Pearson, W.R.(1985). Rapid Sensitive Protein Similarity Searches. Science, 227, 1435-41
9. "Trends Guide to Bioinformatics", 1998 Elsevier Science.