스트링 B-트리를 이용한 게놈 서열 분석 시스템

최 정 현[†] · 조 환 규[††]

요 약

생명 과학의 발전과 많은 게놈(genome) 프로젝트의 결과로 여러 종의 게놈 서열이 밝혀지고 있다. 생물체의 서열을 분석하는 방법은 전역정렬(global alignment), 지역정렬(local alignment) 등 여러 가지 방법이 있는데, 그 중 하나가 *k*-mer 분석이다. *k*-mer는 유전자의 염기 서열내의 길이가 *k*인 연속된 염기 서열로서 *k*-mer 분석은 염기서열이 가진 *k*-mer들의 빈도 분포나 대칭성 등을 탐색하는 것이다. 그런데 게놈의 염기 서열은 대용량 텍스트이고 *k*가 클 때 기존의 온메모리 알고리즘으로는 처리가 불가능하므로 효율적인 자료구조와 알고리즘이 필요하다. 스트링 B-트리는 패턴 일치(pattern matching)에 적합하고 외부 메모리를 지원하는 좋은 자료구조이다. 본 논문에서는 스트링 B-트리(string B-tree)를 *k*-mer 분석에 효율적인 구조로 개선하여, *C. elegans* 외의 30개의 게놈 서열에 대해 분석한다. *k*-mer들의 빈도 분포와 대칭성을 보여주기 위해 CGR(Chaotic Game Representation)을 이용한 가시화 시스템을 제시한다. 게놈 서열과 매우 유사한 서열 상의 어떤 부분을 시그니처(signature)라 하고, 높은 유사도를 가지는 최소 길이의 시그니처를 찾는 알고리즘을 제시한다.

An Analysis System for Whole Genomic Sequence Using String B-Tree

Jeong-Hyeon Choi[†] · Hwan-Gue Cho[††]

ABSTRACT

As results of many genome projects, genomic sequences of many organisms are revealed. Various methods such as global alignment, local alignment are used to analyze the sequences of the organisms, and *k*-mer analysis is one of the methods for analyzing the genomic sequences. The *k*-mer analysis explores the frequencies of all *k*-mers or the symmetry of them where the *k*-mer is the sequenced base with the length of *k*. However, existing on-memory algorithms are not applicable to the *k*-mer analysis because a whole genomic sequence is usually a large text. Therefore, efficient data structures and algorithms are needed. String B-tree is a good data structure that supports external memory and fits into pattern matching. In this paper, we improve the string B-tree in order to efficiently apply the data structure to *k*-mer analysis, and the results of *k*-mer analysis for C. elegans and other 30 genomic sequences are shown. We present a visualization system which enables users to investigate the distribution and symmetry of the frequencies of all *k*-mers using CGR (Chaotic Game Representation). We also describe the method to find the signature which is the part of the sequence that is similar to the whole genomic sequence.

키워드 : 문자열 일치(string matching), pattern matching, 스트링 B-트리(string B-tree), 서열 분석(sequence analysis), *k*-mer 분석(*k*-mer anlysis)

1. 서 론

생명 공학의 빠른 발전으로 여러 생물체의 전체 염기 서열(whole genomic sequence)이 밝혀지고 있다. 염기 서열의 기능이나 진화적 관계를 밝히기 위해서는 염기 서열을 비교 분석해야 한다. 많은 염기 서열 분석 방법 중 하나가 *k*-mer 분석 방법이다. *k*-mer는 올리고핵산(oligonucleotide) 또는 단어(words)와 같은 말로 사용되며 유전자의 염기 서열내의 길이가 *k*인 연속된 염기를 말한다. *k*-mer 분석 방법은 염기서열이 가진 *k*-mer들의 빈도 분포나 대칭성 등을 탐색하는 것이다.

염기 서열에 대한 *k*-mer 분석은 1980년대부터 연구되어 왔다. 1991년에 Burge *et. al.*이 di-, tri-, tetra-nuleotide의 상대적 빈도를 이용하여 DNA 서열에서 짧은 올리고핵산의 과발현(over-representation)과 미발현(under-representation)에 대해 설명했다[1]. Karlin *et. al.*은 이 방법을 이용하여 진핵 생물의 게놈 서열에 대한 비교를 하였다[2]. Blaisdell *et. al.*은 di-nucleotide relative abundance value를 측정하여 박테리오파지 게놈들의 유사성과 상이성에 대해[3], Karlin *et. al.*은 진핵 생물별 구성의 차이와 게놈의 부분별 구성의 차이에 대해[4], 진핵 생물의 미토콘드리아와 게놈의 복합된 원핵 조상에 대해 분석하였다[5].

k-mer 분석을 위해서는 모든 *k*-mer의 빈도를 구해야 하는데 이것은 전형적인 문자열 일치(pattern matching) 문제이다. 즉 전체 염기 서열이 텍스트이고 하나의 *k*-mer가

※ 본 연구는 한국과학재단 목적기초연구(2000-1-30300-012-2)지원으로 수행되었음.
† 준 회 원 : 부산대학교 대학원 전자계산학과
†† 정 회 원 : 부산대학교 전기전자정보컴퓨터공학부 교수
논문접수 : 2001년 7월 10일, 심사완료 : 2001년 10월 9일

패턴이 되어 텍스트에서 패턴이 나타나는 횟수와 위치를 구하는 문제와 같다. 문자 일치 알고리즘인 Kruth-Morris-Pratt[6], Boyer-Moore[7], Karp-Rabin[8]의 알고리즘 등을 이용하면 모든 k-mer의 빈도들을 구할 수 있다. 그러나 길이 m인 서열에서 하나의 k-mer에 대해 $O(m)$이 걸리고 존재할 수 있는 모든 k-mer의 개수는 4^k이므로 전체적으로 $O(4^k m)$이 걸린다.

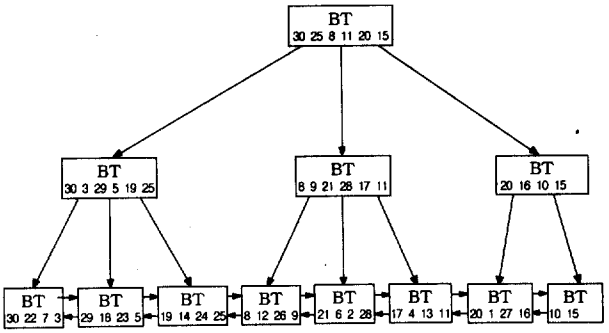
염기 서열에서 모든 k-mer들의 빈도를 가장 빠르게 구하는 방법은 Karp-Rabin[8] 알고리즘을 응용하는 방법이다. 길이가 m인 서열을 S라 하면 $S=S[1]S[2]\cdots S[m]$, $\forall 1 \le j \le m$, $S[j] \in \{A,C,G,T\}$이고, i번째 k-mer는 $S[i][i+1]\cdots S[i+k-1]$이다. 하나의 k-mer는 Karp-Rabin의 알고리즘에 의해 숫자로 변환된다.

$$N_i = H(S[i]) \times 4^{k-1} + H(S[i+1]) \times 4^{k-2} + \cdots + H(S[i+k-1])$$

여기서 $H(A)=0$, $H(C)=1$, $H(G)=2$, $H(T)=3$이다. 모든 가능한 k-mer의 수가 4^k이므로 N_i의 값은 0에서 4^k-1까지의 값을 가진다. 그러므로 모든 k-mer의 빈도를 저장하기 위해서는 4^k의 공간이 필요하다. 이 알고리즘의 수행 시간은 $O(m)$이고 특정 패턴의 빈도에 대한 탐색 시간은 $O(k)$이다. 그러나 이 방법은 공간의 사용량이 $O(4^k)$인 단점이 있다. 실제로 12-mer는 256MB의 메모리가 필요하고, 13-mer 이상일 때는 메모리가 부족하여 수행할 수 없다.

이 방법보다 더 좋은 방법은 패턴 일치를 위한 서픽스 트리와 같은 인덱스 구조를 생성한 후 각 k-mer의 빈도를 구하는 것이다. 서픽스 트리를 선형 시간에 생성하는 알고리즘은 Weiner가 처음 제안하였다[9]. McCreight는 Weiner의 알고리즘을 개선하여 공간을 적게 사용하는 알고리즘을 개발하였다[10]. Ukkonen는 이해와 구현이 쉽고 온라인에서 사용할 수 있는 알고리즘을 제안하였다[11]. 서픽스 트리의 생성 시간은 $O(m)$이고 문자 검색 시간은 $O(k)$이다. 그러나 서픽스 트리의 공간 사용량은 $O(m)$이지만 사람의 염색체 염기 서열이 40MB 이상의 대용량이므로 메모리 부족으로 수행할 수 없거나 가상 메모리의 사용으로 속도가 매우 저하된다. 많은 사람들이 이 단점을 극복하기 위해 노력하여 현재 가장 효율적인 구현은 Compact PAT-trees를 이용한 방법으로 $5m$ 바이트의 공간을 사용한다[12].

대용량의 텍스트에 대해서는 외부 메모리의 사용이 필수적으로 역파일(inverted file), B-트리(B-tree), 프리픽스 B-트리(prefix B-tree)와 같은 변형들이 있다. Ferragina는 B-트리에 기반한 스트링 B-트리(string B-tree)를 개발했다[13]. 스트링 B-트리는 외부 메모리 자료구조에 가장 좋은 B-트리와 서픽스 배열의 특성을 조합한 패턴 일치를 위한 자료구조이다. 그것은 역파일(수정성과 atomic key), 서픽스 배열(수정성과 연속 공간), 서픽스 트리(불균형한 트리 위상), 프리픽스 B-트리(bounded-length key) 등의 제한을 모두 극복하였고, 최악의 경우에 B-트리와 같은 성능을 가지는 첫 번째 외부 메모리 자료 구조이다. 페이지의 크기를 B라 하고 문자열의 길이를 m이라 하면 스트링 B-트리 생성은 $O(m \log(Bm))$ 디스크 접근이 필요하고, f번 나타나는 길이 n인 문자열 P에 대한 검색은 $O((n+f)/B+\log(Bm))$ 디스크 접근이 필요하다.



(그림 1) 길이 30인 문자열 TTATCTAGGTTGTCTTTCCTTACGGGTTCA에 대한 스트링 B-트리. 단말 노드에는 서픽스들의 번호가 사전순으로 정렬되어 있고, 내부 노드에는 자식 노드의 가장 왼쪽 서픽스 번호와 가장 오른쪽 번호들이 저장되어 있다. 모든 노드는 검색을 빠르게 하기 위해 BT(Blind Trie)를 가지고 있다.

본 논문에서는 k-mer 분석을 위해 패턴 일치(pattern matching)를 지원하고 외부 메모리 구조에 적합한 스트링 B-트리(string B-tree)를 개선하여 사용한다.

2. 스트링 B-트리(String B-Tree)

대용량 문자열의 서픽스 트리는 메모리를 많이 사용하므로 외부 메모리를 사용해야 한다. 외부 메모리를 사용할 경우 서픽스 트리의 높이가 너무 높아 디스크 입출력 시간이 많이 걸린다. 이런 단점을 해결하는 자료구조가 스트링 B-트리이다. 스트링 B-트리는 기존의 B-트리를 문자열 처리에 맞게 변형시킨 것이다.

길이가 m인 문자열 S의 서픽스들을 S_i, 이 서픽스들이 사전순으로 정렬된 집합을 $K=\{K_1, K_2, \cdots, K_m\}$, 노드 π에 대해 노드에 저장된 문자열의 인덱스 집합을 S_π, S_π의 가장 왼쪽 문자열 인덱스를 $L(\pi)$, S_π의 가장 오른쪽 문자열 인덱스를 $R(\pi)$라고 하자. S에 대한 스트링 B-트리 SBT는 단말 노드에 n개의 서픽스 S_i들의 인덱스가 사전 순서로 저장되고, 내부 노드 π는 $n(\pi)$개의 자식 노드 $\sigma_1, \sigma_2, \cdots, \sigma_n(\pi)$를 가지고, 각 자식 노드의 $L(\sigma)$와 $R(\sigma)$의 집합 $S_\pi = \{L(\sigma_1), R(\sigma_1), L(\sigma_2), R(\sigma_2), \cdots, L(\sigma_n(\pi)), R(\sigma_n(\pi))\}$를 가진다. (그림 1)은 길이가 30인 문자열 TTATCTAGGTTGTCTTTCCTTACGGGTTCA에 대한 스트링 B-트리이다. (그림 1)에서 볼 수 있듯이 모든 노드는 Blind Trie(BT)를 가진다. Blind Trie는 어떤 문자열이 노드 π의 문자열 인덱스 집합 S_π에서 사전순으로 따질때 어느곳에 위치하는

지를 효과적으로 검색할 수 있도록 한다.

또한 하나의 문자열의 서픽스들에 대한 스트링 B-트리를 빠르게 생성하기 위해서는 *succ pointer*가 필요하다. 서픽스 S_i의 *succ pointer*는 S_{i+1}을 포함하는 단말 노드에 대한 포인터이고, $succ^{-1}$ *pointer*는 S_{i-1}을 포함하는 단말 노드에 대한 포인터이다.

스트링 B-트리의 단말 노드 π는 다음과 같은 구조를 가진다.

- $n(\pi)$개의 문자열 인덱스의 집합 $S_\pi = \{i \mid S_i$ is the suffix$\}$.
- 문자열 인덱스 집합 S_π에 대한 BT
- 단말 노드들을 이중 연결 리스트로 만들기 위한 *next*(π)와 *prev*(π) 포인터.
- $lcp(R(prev(\pi)), L(\pi))$와 $lcp(R(\pi), L(next(\pi)))$.
- *parent*(π) 포인터
- S_π 문자열에 대한 *succ*와 $succ^{-1}$ 포인터 집합.

스트링 B-트리의 내부 노드 π는 다음의 정보를 가진다.

- $n(\pi)$개의 자식 노드 포인터 집합 $C_\pi = \{ \sigma_1, \sigma_2, \cdots, \sigma_{n(\pi)} \}$
- $2n(\pi)$개의 문자열 인덱스 집합 $S_\pi = \{ L(\sigma_1), R(\sigma_1), L(\sigma_2), R(\sigma_2), \cdots, L(\sigma_{n(\pi)}), R(\sigma_{n(\pi)}) \}$.
- 문자열 인덱스 집합 S_π에 대한 BT
- *parent*(π) 포인터

여기서 $lcp(S_1, S_2)$란 문자열 S_1과 S_2 사이의 가장 긴 공통의 프리픽스(longest common prefix)를 말한다.

스트링 B-트리에서 특정 문자열의 빈도 계산은 단말노드에 문자열(서픽스)들이 정렬되어 있으므로 단말 노드에서 사전순서로 볼 때 그 문자열의 시작 위치와 끝 위치를 찾는다. 예를 들어, 염기 서열에서 ACGTACG를 찾는다면 단말 노드에서 ACGTACG#의 위치와 ACGTACT#의 위치를 찾는다. 여기서 #은 염기서열의 끝문자로 C 언어에서 NULL과 같다. 찾은 각 위치는 노드와 문자열 인덱스 배열 S_π에서의 인덱스로 주어진다. 단말 노드들은 이중 연결 리스트로 연결되어 있으므로 앞에서 찾은 시작 위치의 노드에서 끝 위치의 노드가 나올 때까지 리스트를 따라가며 빈도를 증가한다. 사전순서로 볼 때 단말노드들에서 어떤 문자열의 위치는 루트 노드의 문자열 인덱스 집합 S_π에서 BT를 이용하여 그 문자열의 위치를 찾고 해당 자식 노드로 이동하는 연산을 단말 노드에 도달할 때까지 순환적으로 수행하면 된다.

3. *k*-mer 분석 시스템

3.1 스트링 B-트리의 확장

효율적인 검색을 위해 스트링 B-트리의 단말 노드에 인접한 서픽스들의 *lcp* 정보를 저장한다. 즉, $n(\pi)$개의 문자열 인덱스 집합 S_π에 대해 $n(\pi)$개의 $LCP_\pi = \{lcp(S_i, S_{i+1}) \mid 0 \le i < m\}$이 저장된다. 이것을 이용하면 Ferragina가 패턴 일치시에 구했던 오른쪽 경계를 구하지 않고 단말 노드를 따라가면서 lcp가 k보다 적으면 끝이라는 것을 알 수가 있으므로 속도가 향상된다.

유전자 염기서열이 정적이라는 사실을 이용하면 Ferragina가 제안한 방법보다 더 빠르게 스트링 B-트리를 생성할 수 있다. Manber와 Myers가 개발한 방법으로 염기 서열의 서픽스들을 정렬하여 정렬 리스트와 *lcp* 리스트를 구한다[14]. 그런 다음 정렬 리스트에 해당하는 서픽스들을 순서대로 단말 노드에 추가하고, 생성된 단말 노드의 부모 노드인 내부 노드를 만든다. 같은 높이의 노드가 하나(루트 노드)가 될 때까지 계속 내부 노드를 만든다. 이 방법의 다른 장점은 공간을 컴팩트하게 사용한다.

3.2 *k*-mer의 빈도 계산

스트링 B-트리에서 *k*-mer의 빈도를 구하는 것은 Ferragina가 제안한 패턴 일치 알고리즘을 사용하면 된다[13]. 그러나 스트링 B-트리에서 모든 *k*-mer의 빈도를 구할 때 위의 방법은 반복적인 루트 노드에서 단말 노드로의 검색과 빈도가 0인 *k*-mer에 대해서도 수행을 하므로 효율적이지 못하고 시간이 많이 걸린다.

스트링 B-트리의 모든 단말 노드에는 서픽스들이 사전순으로 정렬되어 있고 인접한 서픽스들의 *lcp* 정보가 있기 때문에 단말 노드를 순회하면서 *k*-mer의 빈도를 계산하는 것이 더 빠르다. *k*를 구하고자 하는 *k*-mer라 하고, N_{leaf}를 하나의 단말노드에 저장되는 서픽스의 개수라 하고, *F*를 해당 *k*-mer의 빈도가 저장되는 리스트라 할때 (알고리즘 1)은 빈도가 0이 아닌 모든 *k*-mer들의 빈도를 구한다.

```
SB-kmer
    Input : k      // k-mer
    Output : F     // 빈도 리스트
count = 0 ;
pat = 첫 번째 k-mer ;
node = 첫 번째 단말노드 ;
while node ≠ NULL
  node를 메모리에 로드한다 ;
  for i from 0 to N_leaf 1
   s = node의 i번째 서픽스 ;
   lcp = node의 i번째 lcp ;
   if lcp > k then
     count = count + 1 ;
   else
     if length(s) ≥ k then
       F에 count 추가 ;
       pat = s[0] ··· s[k-1] ;
     end if
     count = 1 ;
   end if
  end for
  node = next(node) ;
 end while
```

(알고리즘 1) 단말 순회를 이용한 모든 *k*-mer 빈도 구하기

3.3 유사도 측정 방법

두 개의 염기 서열 S_1과 S_2에 대해서 k-mer 분석을 하면 두 개의 빈도 테이블 F_1과 F_2가 계산된다. 이것을 이용하여 두 염기 서열간의 유사도를 측정할 수 있다. 두 염기 서열의 유사도를 측정하는 방법은 Compell[15]의 방법과 Deschavanne[16]의 방법이 있다. Deschavanne의 방법은 다음과 같이 최소제곱법(method of least squares)으로 구한다.

$$Sim(S_1, S_2) = \sqrt{\sum_i (F_{1,i} - F_{2,i})}$$

여기서 $F_{1,j}(F_{2,j})$는 서열 $S_1(S_2)$의 j번째 k-mer의 빈도를 의미한다.

우리는 Deschavanne의 방법을 수정하여 사용한다. 각 빈도를 최대 빈도로 정규화한 후 최소제곱법을 적용한다. 그 이유는 염기 서열의 길이에 따라 k-mer의 빈도가 증가하기 때문에 길이에 의존하지 않는 유사도 측정을 하기 위해서이다. 최종적인 식은 다음과 같이 된다.

$$Sim(S_1, S_2) = \sqrt{\sum_i \left(\frac{F_{1,i}}{F_{1,\max}} - \frac{F_{2,i}}{F_{2,\max}}\right)^2}$$

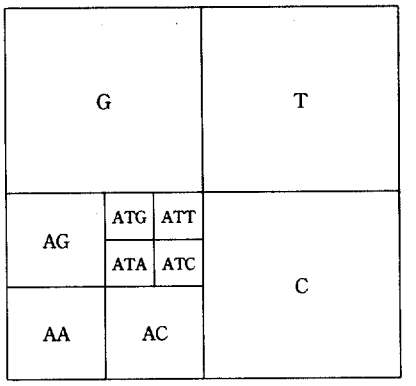
(수식 1) 유사도 측정 방법

여기서 $F_{1,\max}(F_{2,\max})$는 $F_1(F_2)$에서 빈도가 최대인 k-mer이다

4. 가시화 시스템

k-mer analysis의 결과를 효과적으로 가시화하여 사람이 분석하기 용이하도록 할 필요가 있다. 1990년에 Jeffery는 게놈의 서열을 위한 스케일에 독립적인 표현으로서 CGR(Chaos Game Representation)을 제안하였다[17]. 이 제안은 임의의 문자 서열을 위해 확장되었고[18], 단백질과 같은 다른 생물학적 서열들을 포함하게 되었다[19, 20]. 그러나 Jeffery의 제안은 CGR이 nucleotide 서열을 표현하는 데 사용될 가능성을 완전히 탐색하지 못했다[21]. 3년이 지난 후에 CGR 빈도를 oligonucleotide의 빈도로 해석하는 것이 논증되었고, CGR이 단순한 nucleotide, di-nucleotide, tri-nucleotide의 분석보다 더 유용하다고 해석되었다[22]. 또한 CGR이 일반화되고 스케일 독립적인 HMM(Hidden Markov Model)이라는 것이 알려지게 되었다[23].

서열에 대한 CGR은 쿼드트리(quad-tree)와 유사한 가시화 방법으로 k-mer 빈도의 분포를 잘 보여 준다. 모든 k-mer들은 사각형에 순환적인(recursive) 구조로 배치되고 빈도는 색깔로 나타낸다. 하나의 사각형을 4등분하면 monomer인 A, C, G, T의 영역으로 나누어진다. 각 사각형을 다시 4등분하면 di-mer인 AA, AC, AG, AT, , TT의 영역으로 나누어진다. 계속적으로 각 사각형으로 나누면 tri- mer의 영역으로 나누어진다. k까지 계속적으로 하면 각 k-mer의 영역으로 나누어진다. (그림 2)는 쿼드트리의 영역에서 tri-mer ATA, ATC, ATG, ATT의 위치를 보여준다.



(그림 2) CGR에서 3-mer인 ATA, ATC, ATG, ATT의 위치

각 k-mer의 빈도는 색깔로 표현된다. 먼저 빈도를 최대 빈도로 정규화하고 그 값으로 색깔을 결정한다. 색깔은 검은색, 빨강색, 노랑색, 녹색, 청록색, 파란색, 흰색의 7 단계로 나눈다. 정규화된 빈도는 7단계에서 자기가 속한 구간의 시작과 끝의 색으로 보간 한다.

CGR 그림에서 오른쪽 위에서 왼쪽 아래로의 대각선에는 A나 T로만 구성된 k-mer가 배치되고, 왼쪽 위에서 오른쪽 아래로의 대각선에는 G나 C로만 구성된 k-mer가 배치되므로 AT-rich나 GC-rich를 쉽게 살펴볼 수 있다. (그림 3) (a)와 (b)는 *A.thaliana*의 염색체 II와 IV에 대한 CGR_6 그림이고, (그림 4) (a)와 (b)는 *H.sapiens*의 염색체 21과 22에 대한 CGR_6 그림이다. 두 그림에서 (a)와 (b)의 색깔 분포가 거의 같다. 결론적으로 진핵 생물에 속하는 종들의 염색체에 대한 CGR은 매우 유사하다. 각각의 그림은 순환적인(recursive) 구조를 가지는 특징이 있다. (그림 4)에서 검은 사각형 영역은 CG 미발현을 나타낸다.

(그림 5)는 원핵 생물에 속하는 몇 개의 종에 대한 CGR_6이다. (a)는 *E.coli*, (b)는 *D.radiodurans*, (c)는 *Synechocystis*, (d)는 *R.prowazekii*이다. 각각의 그림은 고유한 빈도 분포를 가진다. 다른 원핵 생물에 대한 CGR_6 그림은 (그림 5)의 네 가지 형태 중 하나의 형태를 가진다.

5. 게놈 시그니처(Genomic Signature)

게놈의 한 부분이 전체 게놈과 매우 유사하면 그 부분이 전체를 대표할 수 있다. 이것을 게놈의 시그니처(signature)라 부른다. $GS(p, l)$이 게놈 S에 대해 p에서 시작하고 길이 l인 게놈의 시그니처라 하자. 시그니처가 게놈을 대표할 수 있는 정도는 시그니처와 게놈의 유사성으로 측정할 수 있고, (수식 1)에 의해 계산될 수 있다.

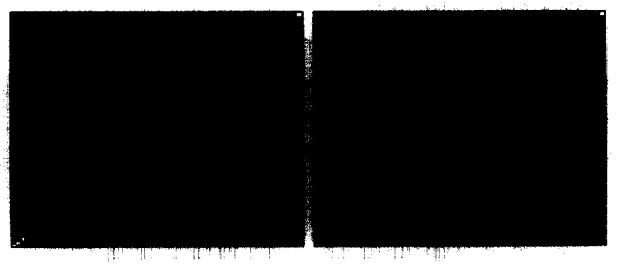
일반적으로 시그니처의 길이가 길수록 시그니처와 게놈의 유사성이 높아지고, 게놈을 더 대표할 수 있다. Karlin *et.al.*은 50K bp[2], Deschavanne *et.al.*은 100K bp[16]를 시그니처의 길이로 정하였다.

우리는 시그니처의 길이에 따른 시그니처와 게놈의 유사도를 측정하였다. (그림 6)는 인간 염색체 21번의 시작 위치

($p=1$)에서 시그니쳐의 길이에 따른 CGR_6 그림을 보여준다. 그림에서 보듯이 30K bp 미만에서는 전체와 차이가 있고, 30K bp 이상에서는 전체와 비슷한 패턴이 나타나기 시작하여 50K bp에서는 전체와 거의 똑같은 패턴이 나타난다.

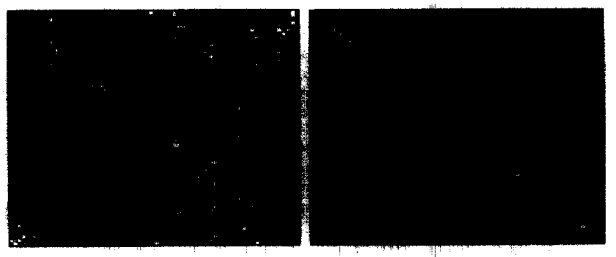
(a) 염색체 II (b) 염색체 IV

(그림 3) *A. thaliana*에 대한 CGR_6. (a)는 염색체 2번, (b)는 염색체 4번에 대한 hexa-nulceotide의 빈도 분포를 보여준다. 두 개의 그림이 매우 유사한다는 것은 주목할 만하다.

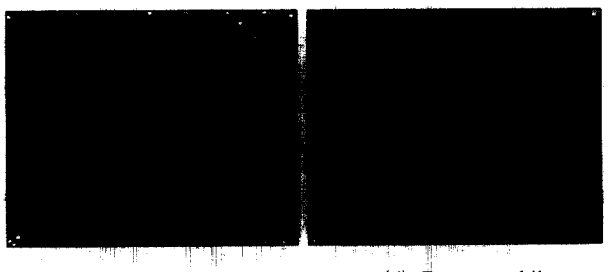


(a) 염색체 21 (b) 염색체 22

(그림 4) *H. sapiens*에 대한 CGR_6. (a)는 염색체 21번, (b)는 염색체 22번에 대한 hexa-nulceotide의 빈도 분포를 보여준다. 두 개의 그림이 전체적으로 볼 때 유사하다. 검은 사각형은 인간 게놈 서열에서 CG가 많이 나타나지 않음을 보여준다.



(a) *E.coli* (b) *D.radiodurans*

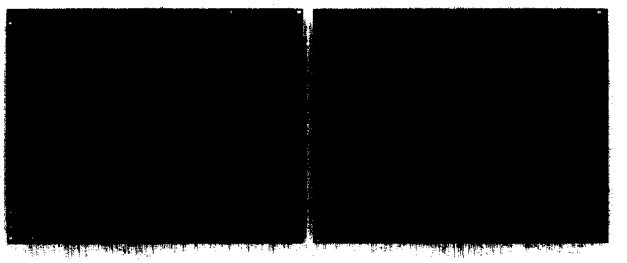


(c) *Synechocystis* (d) *R.prowazekii*

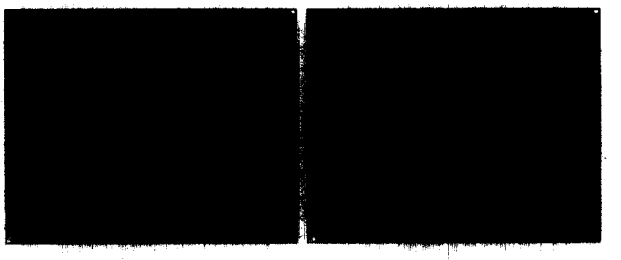
(그림 5) 원핵 생물에 속하는 몇 개의 종에 대한 CGR_6. 많은 원핵 생물의 게놈에 대한 CGR_6 그림은 여기에서 보인 네 가지 중 하나의 형태를 가진다.

두 염기 서열의 유사도를 비교하는 것은 $O(4^k)$의 수행 시간이 걸리므로, 모든 $p(1 \le p \le m-l)$에 대해 $GS(p,l)$을 구하는 것은 $O(m4^k)$ 정도의 많은 시간이 걸린다. 이것을 개선하는 방법은 게놈의 시작 ($p=1$)에서 길이 l인 시그니쳐의 k-mer 빈도와 유사도를 구한다 $GS(1,l)$. 다음 위치 ($p>1$)의 시그니쳐는 마지막 부분에 새 k-mer가 추가되고 이전의 첫 부분의 k-mer가 삭제되므로 이에 따라 빈도 배열을 갱신한다. 새 시그니쳐의 전체와의 유사도는 다음의 식을 이용하여 구할 수 있다.

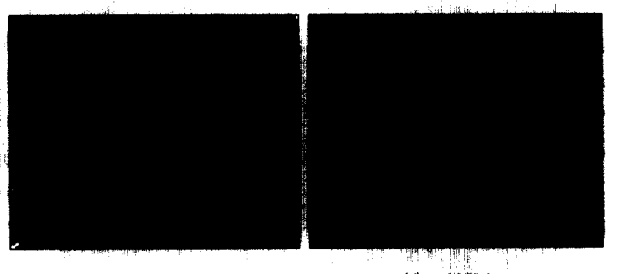
$$GS(p,l)^2 = \sum_{i=0}^{4^k-1} \left(\frac{f_{s,i}}{f_{s,\max}} - \frac{f_{g,j}}{g_{s,\max}}\right)^2$$
$$= \frac{A_p}{f_{s,\max}^2} - 2\frac{B_p}{f_{s,\max} f_{g,\max}} + \frac{C_p}{f_{g,\max}^2}$$



(a) 1K bp (b) 5K bp



(c) 10K bp (d) 30K bp



(e) 50K bp (f) 100K bp

(그림 6) 시작 위치 p가 1이고 길이 l이 10K bp, 30K bp, 50K bp, 100K bp일때 인간 염색체 21번의 시그니쳐에 대한 CGR_6. 시그니쳐의 길이가 30K bp 미만에서는 게놈 전체의 CGR_6과 차이가 많고, 30K bp 이상에서는 전체와 비슷한 패턴이 나타나기 시작하여 50K bp에서는 전체와 거의 똑같은 패턴이 나타난다.

$$A_p = \sum_{i=0}^{4^k-1} f_{s,i}^2$$

$$B_p = \sum_{i=0}^{4^k-1} f_{s,j} f_{g,j}$$

$$C_p = \sum_{i=0}^{4^k-1} f_{g,i}^2$$

여기서 $f_{s,i}(f_{g,i})$는 시그니쳐(게놈)의 i 번째 k-mer의 빈도를 의미하고, $f_{s,\max}(f_{g,\max})$는 시그니쳐(게놈)의 빈도 중 최대값을 의미한다. 추가되는 k-mer의 인덱스를 w라 하고, 삭제되는 k-mer의 인덱스를 u라 하자. 만약 $u=w$라면 GS는 변화가 없다. 다른 경우에는 C_p와 $f_{g,\max}$는 상수이므로 나머지 항만 계산하면 된다. A_p와 B_p는 다음의 식으로 구할 수 있다.

$$A_{p+1} - A_p = -2f_{p,u} + 2f_{p,w} + 2$$
$$B_{p+1} - B_p = -f_{g,u} + f_{g,w}$$

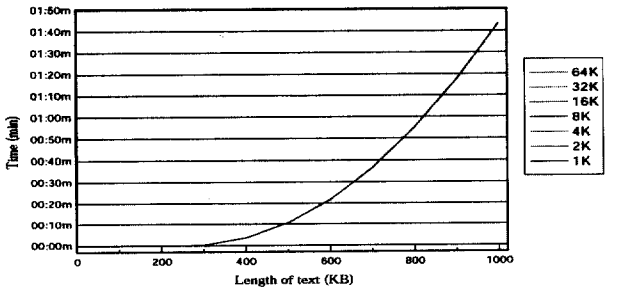
이제 GS를 구하기 위해서는 $f_{s,\max}$만 구하면 된다. 이것은 $f_{s,i}$에 대한 힙을 구성하면 $O(k)$만에 최대값을 알 수 있으므로 하나의 시그니쳐의 유사도는 $O(k)$만에 구할 수 있고, 모든 시그니쳐의 유사도는 $O(4^k+mk)$의 시간에 구할 수 있다.

6. 실험 결과

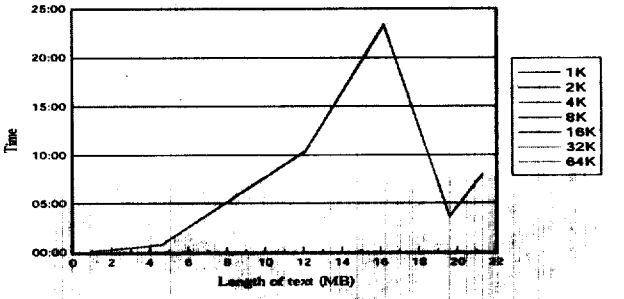
본 논문에서 제시한 시스템은 CPU가 300MHz (R12000), 메모리가 512MB인 SGI *Octane*에서 C++와 *Open Inventor* 라이브러리를 사용하여 구현하였다. 실험에 사용한 데이터는 *NCBI*에 있는 생물체의 전체 염기 서열(whole genome)이다. 시간에 관련된 실험 결과는 *timex* 프로그램을 이용하여 측정한 시간이다.

6.1 스트링 B-트리의 생성 시간

(그림 7)은 랜덤 데이터와 6종의 게놈 서열에 대해 본 논문에서 제안한 방법으로 스트링 B-트리를 생성하는데 걸리는 시간을 보여준다. 염기 서열의 길이가 약 16M bp인 *C. elegans*에의 염색체 I에 대한 스트링 B-트리의 생성 시간은 25분 걸리고, 약 21M bp인 *C. elegans*에의 염색체 V에 대한 스트링 B-트리의 생성 시간은 12분 걸린다. 생성 시간이 서열의 길이에 비례하지 않는 이유는 서픽스들의 정렬 리스트에서 인접한 서픽스들의 *lcp*들이 차이가 나기 때문이다. *lcp*들이 작을수록 서픽스 정렬이 빨라지므로 더 빠르게 스트링 B-트리를 생성할 수 있다. 페이지의 크기는 생성 시간이나 파일 크기에 별로 영향을 미치지 않는다.



(a)

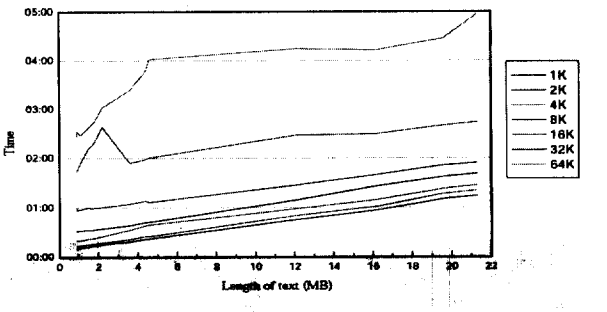


(b)

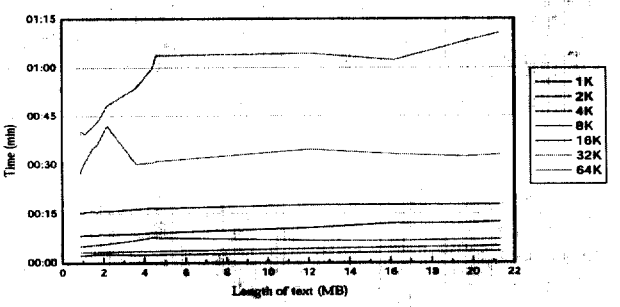
(그림 7) 랜덤 데이터(a)와 6종의 게놈 서열(b)에 대한 스트링 B-트리 생성 시간. 생성 시간이 서열의 길이에 비례하지 않는 이유는 서픽스들의 정렬 리스트에서 인접한 서픽스들의 lcp들이 차이가 나기 때문이다. lcp들이 작을 수록 서픽스 정렬이 빨라지므로 더 빠르게 스트링 B-트리를 생성할 수 있다. 페이지의 크기는 생성 시간이나 파일 크기에 별로 영향을 미치지 않는다.

6.2 k-mer의 빈도 계산

(그림 8)은 Ferragina가 제안한 방법으로 6-mer(a)와 8-mer(b)의 빈도를 계산하는 데 걸리는 시간에 대한 그래프이다. 약 4M bp인 *E.coli*의 게놈 서열에 대해 k가 6에서 8로 증가할 때 수행 시간이 4분에서 65분으로 증가하고 페이지의 크기에 영향을 많이 받았다.



(a) 6-mer

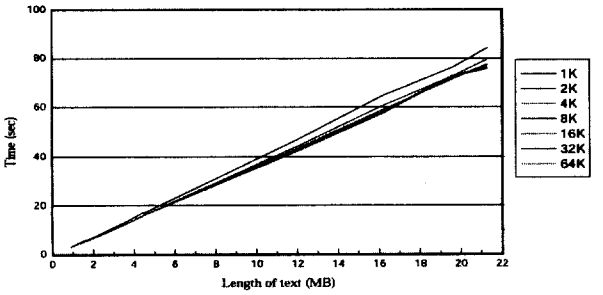


(b) 8-mer

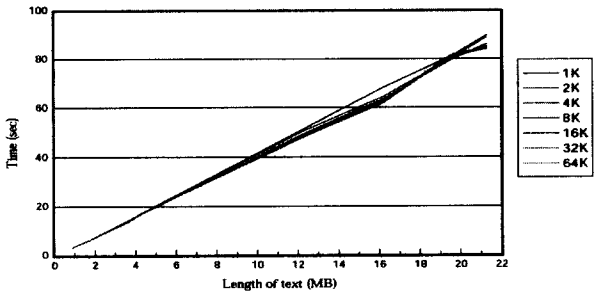
(그림 8) Ferragina의 방법으로 k-mer의 빈도 계산에 걸린 시간에 대한 그래프. 약 4M bp인 *E.coli*의 게놈 서열에 대해 k가 6에서 8로 증가할 때 수행 시간이 4분에서 65분으로 증가하고 페이지의 크기에 영향을 많이 받았다.

(그림 9)는 본 시스템에서 제안한 방법으로 계산할 때 걸리는 시간이다. 약 21M bp인 *C. elegans*의 염색체 V에 대

해 k가 6일때 Ferragina가 제안한 방법은 5분이 걸렸지만 단말노드 순회법으로 계산한 것은 90초정도 걸린다. Ferragina가 제안한 방법은 k가 6에서 8로 증가하면 5분에서 70분으로 증가하지만 본 논문에서 제안한 단말노드 순회법에서는 k가 6에서 20으로 증가하더라도 시간이 거의 증가하지 않는다는 것은 주목할 만하다.

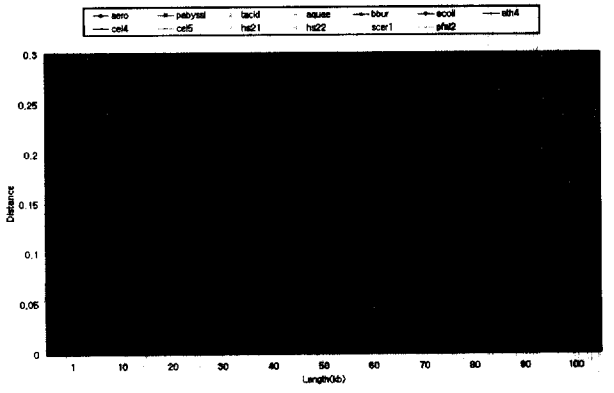


(a) 6-mer



(b) 20-mer

(그림 9) 단말노드 순회 방법으로 k-mer의 빈도 계산에 걸린 시간에 대한 그래프. 약 21M bp인 C. elegans의 염색체 V에 대해 k가 6일때 Ferragina가 제안한 방법보다 단말노드 순회법이 수행 시간이 적게 걸린다. Ferragina가 제안한 방법은 k가 증가하면 수행 시간이 증가하지만 본 논문에서 제안한 단말노드 순회법에서는 k가 증가하더라도 시간이 거의 증가하지 않는다.



(그림 10) 길이의 변화에 따른 시그니쳐와 게놈의 평균 유사도의 변화. 대부분의 생물체가 30K bp 미만에서는 전체와 차이가 있고, 30K bp 이상에서는 전체와 비슷한 패턴이 나타나기 시작하여 50K bp에서는 전체와 거의 똑 같은 패턴이 나타난다.

6.3 길이에 따른 시그니쳐와 게놈의 유사도 계산

(그림 10)은 다양한 종에 대해 길이가 변화할 때 시그니쳐와 게놈의 유사도 변화를 보여준다. 여기에 사용된 유사도 측정 방법은 (수식 1)이다. 그림에서 알 수 있듯이 길이에 따른 유사도의 변화는 다음의 식으로 나타낼 수 있다.

$$Sim(l) = a(l+c)^{-b}$$

여기서 a, b, c는 곡선의 특징을 나타내는 값이다.

대부분의 생물체가 30K bp 미만에서는 전체와 차이가 있고, 30K bp 이상에서는 전체와 비슷한 패턴이 나타나기 시작하여 50K bp에서는 전체와 거의 똑 같은 패턴이 나타난다.

7. 결론 및 향후 연구과제

본 논문에서는 생물체의 염기 서열 분석과 진화적 관계를 밝히는 데 사용되는 분석 방법 가운데 하나인 k-mer를 이용하는 분석 시스템을 설명하였다. k-mer의 빈도를 구하기 위해 염기 서열의 길이와 메모리량에 따라 Karp-Rabin 알고리즘, 서픽스 트리(suffix tree), 스트링 B-트리(string B-tree)를 사용할 수 있다. Karp-Rabin의 알고리즘을 응용한 방법은 단순하고 속도가 빠른 알고리즘이지만 사용되는 메모리가 4^k이므로 큰 k-mer에 대해 적용할 수 없다. 다른 방법인 서픽스 트리를 이용하는 것은 사람과 같은 대용량 게놈 서열에 대해서는 서픽스 트리를 온메모리 상에 만들 수 없다. 결론적으로 대용량 게놈 서열에 대해 k-mer 분석을 하기 위해 외부 메모리에 효율적인 스트링 B-트리를 사용해야 한다.

본 논문의 의의는 Ferragina가 제안한 스트링 B-트리를 대용량 게놈 서열의 k-mer 분석에 효율적인 자료구조와 알고리즘을 개발한 데 있다. Ferragina[13]가 개발한 스트링 B-트리 생성 방법은 서픽스들을 순서대로 노드에 추가해 나가는 삽입 정렬 방식을 먼저 서픽스들에 대해 정렬을 한 다음 노드를 만들어 가는 상향식 알고리즘으로 변형하여 생성 시간을 단축시켰다. 이 방법을 사용하면 스트링 B-트리 생성 시간은 염기 서열의 길이에 의존하며 $O(m/B)$ 디스크 접근이 필요하다. 이것보다 더 중요한 요소가 정렬된 서픽스 리스트에서 인접한 서픽스들 간의 lcp이다. lcp가 크면 클 수록 생성 시간은 더 걸린다. 또한 k-mer 분석을 위해 단말 노드에 lcp들을 저장하고 이것을 이용한 k-mer들의 빈도 계산 방법을 개발하여 계산 시간을 개선하였다. Ferragina가 제안한 방법으로 k-mer들의 빈도를 계산할 때 수행 시간에 영향을 미치는 것은 k와 m이고, 시간 복잡도는 $O(m/B)$의 디스크 접근과 $O(km)$의 문자 비교이다[13]. 스트링 B-트리의 자료구조를 확장하여 단말노드에 lcp 정보를 저장함으로써 $O(m)$의 문자 비교만 필요하므로 k에 의존하지 않고 k-mer의 빈도를 구할 수 있다.

앞으로 연구해야 할 과제는 본 논문에서 제안한 방법으

로 스트링 B-트리를 생성할 때 온메모리에서 서픽스 정렬을 할 수 없는 초대용량 염기 서열에 대해 서픽스 정렬하는 것이다. Ferragina가 제안한 스트링 B-트리 알고리즘을 이용하면 스트링 B-트리를 생성할 수 있지만 속도가 문제가 된다. 이보다는 디스크 입출력에 효율적인 서픽스 정렬 방법을 연구 개발하는 것이 더 좋다.

참 고 문 헌

[1] C. Burge, A. M. Campbell, and S. Karlin. "Over- and under-representation of short oligonucleotides in DNA sequences," Proc. Natl. Acad. Sci., Vol.89, pp.1358-1362, 1992.

[2] S. Karlin and I. Ladunga, "Comparisons of eukaryotic genomic sequences," Proc. Natl. Acad. Sci., Vol.91, pp.12832-12836, 1994.

[3] B. E. Blaisdell, A. M. Campbell, and S. Karlin, "Similarities and dissimilarities of phage genomes," Proc. Natl. Acad. Sci., Vol.93, pp.5854-5859, 1996.

[4] S. Karlin and J. Mrázek, "Compositional differences within and between eukaryotic genomes," Proc. Natl. Acad. Sci., Vol.94, pp.10227-10232, 1997.

[5] S. Karlin, L. Brocchieri, J. Mrazek, A. M. Campbell, and A. M. Spormann, "A chimeric prokaryotic ancestry of mitochondria and primitive eukaryotes," Proc. Natl. Acad. Sci., USA, Vol.96, No.16, pp.9190-9195, 1999.

[6] D. E. Knuth, J. H. Morris, and V. B. Pratt, "Fast pattern matching," Algorithmica, Vol.6, pp.323-350, 1977.

[7] R. S. Boyer and J. S. Moore, "A fast string matching algorithm," Comm. ACM, Vol.20, pp.762-772, 1977.

[8] R. Karp and M. Rabin, "Efficient randomized pattern matching algorithm," IBM J. Res. Development, Vol.21, pp. 249-260, 1987.

[9] P. Weiner, "Linear pattern matching algorithm," In Proceeding 14th IEEE Symposium on Switching and Automata Theory, pp.1-11, 1973.

[10] E. M. McCreight, "A space-economical suffix tree construction algorithm," Journal of ACM, Vol.23, No.12, pp.262-272, 1976.

[11] E. Ukkonen, "On-line construction of suffix trees, Algorithmica, Vol.14, No.3, pp.249-260, 1995.

[12] D. R. Clark and J. I. Munro, "Efficient suffix trees on secondary storage," In Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp.383-391, 1996.

[13] P. Ferragina and R. Grossi, "The string B-tree : A new data structure for string search in external memory and its application," Journal of ACM, Vo.46, No.2, pp.236-280, 1999.

[14] U. Manber and G. Myers, "Suffix arrays : A new method for on-line string searches," SIAM Journal on Computing, Vol.22, No.5, pp.935-948, 1993.

[15] A. Compell, J. Mrázek, and S. Karlin, "Genome signature comparisons among prokaryote, plasmid, and mitochondrial DNA," Proc. Natl. Acad. Sci., Vol.96, pp.9184-9189, 1999.

[16] P. J. Deschavanne, A. Giron, J. Vilain, G. Fagot, and B. Fertil, "Genomic signature : Characterization and classification of species assessed by chaos game representation of sequences," Mol. Biol. Evol., Vol.16, pp.1391-1399, 1999.

[17] H. J. Jeffery, "Chaos game representation of gene structure," Nucleic Acids Res., Vol.18, pp.2163-2170, 1990.

[18] P. Tino, "Spatial representation of symbolic sequences through iterative function system," IEEE Trans. Syst. Man Cybernet., Vol.29, pp.386-393, 1999.

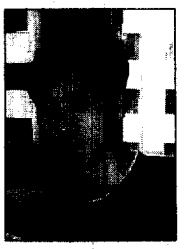
[19] S. Basu, A. Pam, and J. Das, "Chaos game representation of protein," J. Mol. Graphics Mod., Vol.15, pp.279-289, 1997.

[20] K. P. Pleißner, L. Wernisch, H. Osvald, and E. Fleck, "Representation of amino acid sequences as two-dimensional point patterns," Electrophoresis, Vol.18, pp.2709-2713, 1997.

[21] J. S. Almeidal, J. A. Carriço, A. Maretzek, P. A. Noble, and M. Fletcher, "Analysis of genomic sequences by Chaos Game Representation," Bioinformatics, Vol.17, No.5, pp.429-437, 2001.

[22] N. Goldman, "Nucleotide, dinucleotide, and trinucleodtide frequencies explain patterns observed in chaos game respresentations of DNA sequences," Nuclear Acids Res., Vol. 21, pp.2487-2491, 1993.

[23] P. Baldi and S. Brunak, "Bioinformatics : the machine learning approach," MIT Press, 1998.



최 정 현

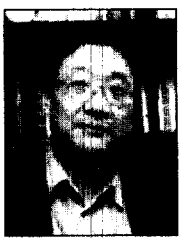
e-mail : jhchoi@pearl.cs.pusan.ac.kr

1995년 부산대학교 물리학과 졸업(학사)

2000년 부산대학교 대학원 전자계산학과 졸업(이학석사)

2000년~현재 부산대학교 대학원 전자계산학과 박사과정

관심분야 : 스트링 매칭, 생물정보학, 정보 가시화



조 환 규

e-mail : hgcho@hyowon.pusan.ac.kr

1984년 서울대학교 계산통계학과 졸업(학사)

1986년 한국과학기술원 전자계산학과 졸업(공학석사)

1990년 한국과학기술원 전자계산학과 졸업(공학박사)

1990년~현재 부산대학교 전기전자정보컴퓨터공학부 교수

관심분야 : 그래프 이론, 생물정보학, 그래픽스