트라이 인덱스를 이용한 DNA 시퀀스 검색

원정임°*, 박용일*, 윤지희*, 박상현**
*한림대학교 컴퓨터공학과, **연세대학교 컴퓨터과학과
{jiwon°,taz102,jhyoon}@hallym.ac.kr, sanghyun@cs.yonsei.ac.kr

DNA Sequence Searching Using a Trie Index

Jung-Im Won°*, Yong-Il Park*, Jee-Hee Yoon*, Sang-Hyun Park**
*Dept of Computer Engineering, Hallym University
**Dept. of Computer Science, Yonsei University

요 약

본 논문에서는 대규모 DNA 시퀀스를 위한 트라이 인덱싱 기법을 기반으로 하는 효율적인 부분 시퀀스 검색 기법을 제시한다. 제안된 인덱싱 방안에서는 저장 공간 감소를 위하여 시퀀스의 각 문자를 최소 비트 정보로 표현하며, 저장 구조로서 포인터를 사용하지 않는 디스크 기반의 이진 접미어 트라이 구조를 사용한다. 질의 처리 방안에서는 포인터가 없는 이진 트라이 구조 상에서 질의 시퀀스를 검색하기 위하여 이진 정보 기반의 연산과정을 필요로 하며, 또한 단말 정보를 효율적으로 검색하기 위하여 별도의 단말정보 테이블과 인덱스 구조를 사용한다. 실험 결과에 의하면 제안된 방식은 기존의 접미어 트리 인덱싱 방식에 비하여 약 30~50%의 저장 공간 감소 효과를 가질 뿐 아니라, 평균 질의 처리 시간에 있어 약 20배까지의 성능 개선 효과를 갖는 것으로 나타났다.

1. 서 론

유전체 데이터베이스는 그 크기가 매우 크며, 증가율 또한 지속적으로 증가하고 있다. 이와 같은 유전체 데이터베이스의 급속한 증가 추세와 활용범위의 확대에 따라, 유전체 데이터베이스를 위한 유사 시퀀스 검색 문제에 대한 연구의 중요성이 더욱 강조되고 있으며, 최근 데이터베이스 분야에서도 인덱싱 기반 기술을 활용한 연구[1][2][3]가 활발히 진행되고 있다. 그러나, 기존의 인덱싱 기술을 이용하여, 대형 유전체 데이터베이스의 인덱스를 구축하는 경우, 상당한 인덱스 구축 시간과 저장 공간을 필요로 한다.

본 논문에서는 대규모 DNA 시퀀스를 위한 부분 시퀀스 검색 문제에 관하여 논한다. 참고 문헌 [4]에서는 DNA 시퀀스의 각 문자를 최소 비트 정보로 표현하고, 저장 구조로서 포인터 없는 디스크 기반의 접미어 트라이를 사용하는 새로운 인덱스 구조와 질의 처리 방식을 제안한 바 있다. 실험 결과에 의하면 제안된 인덱스 방식은 기존의 접미어 트리 방식에 비하여 약 50%의 저장 공간 감소 효과를 갖는 것으로 나타났다. 본 연구는 [4]의 선행 연구 결과를 활용, 확장한 후속 연구로서, 그 주요 내용은 다음과 같다. 제안된 인덱스 구조에서는 모든 정보가 포인터 없는 연속된 이진 형태로 표현되므로 인덱스의 트라이 구조를 직접 파악할 수 없다. 따라서, 페이지의 레벨정보 등을 활용한 구조 파악 및 질의처리가 이루어져야 한다. 또한, 질의 처리 알고리즘에서는 인덱스 상에서 해당 질의와 일치하는 부분 시퀀스를 검색한 후, 해당 노드의 모든 서브 트리를 검색하여 모든 단말 노드 정보를 얻는 과정을 필요로 한다. 이 과정은 인덱스의 검색 공간을 확대시켜, 알고리즘 성능 저하의 주요 원인으로 작용할 수 있다. 본 연구에서는 이진 트라이 인덱스 구조 상에서 단말 노드 정보를 효율적으로 검색하기 위한 두 가지의 서로 다른 접근 방식을 제안하고 그 성능을 비교, 분석한다. 또한, 제안한 기법의 유용성을 평가하기 위하여 실제의 대규모 DNA 시퀀스를 대상으로 하는 다양한 실험을 수행하여 그 성능을 비교, 분석한다.

2. 인덱스 구조

2.1 관련 연구

최근, 참고문헌 [2]에서는 DNA 부분 시퀀스를 웨이브렛 변환(wavelet transformation) 하여 얻어진 변환 계수를 이용하여 다차원 정수 공간으로 매핑한 후, MBR(Minimum Bounding Rectangle)을 이용하는 디스크기반 인덱싱 기법을 제안하고 있다. 한편, 참고문헌 [1]에서는 기존의 텍스트 데이터베이스를 위한 부분 시퀀스 검색에 대하여 가장 좋은 효율을 보이는 것으로 알려진 접미어 트리 인덱스 구조에 착안하여, 이를 대규모 유전체 데이터베이스 인덱싱에 활용하고 있다. 참고문헌 [1]에서는 기존의 접미어 트리 구성 알고리즘을 확장하여 디스크 기반의 인덱스 구성 방식을 제안, 접미어 트리가 실제로 대규모 유전체 데이터베이스 인덱싱 기술로 활용될 수 있음을 보이고 있다. 또한 접미어 트리의 최대 단점인 인덱스 크기의 문제를 보완하기 위한 작업을 소개하고 있으나, 인덱스의 크기는 여전히 데이터베이스의 크기보다 매우 큰 단점을 가지고 있다.

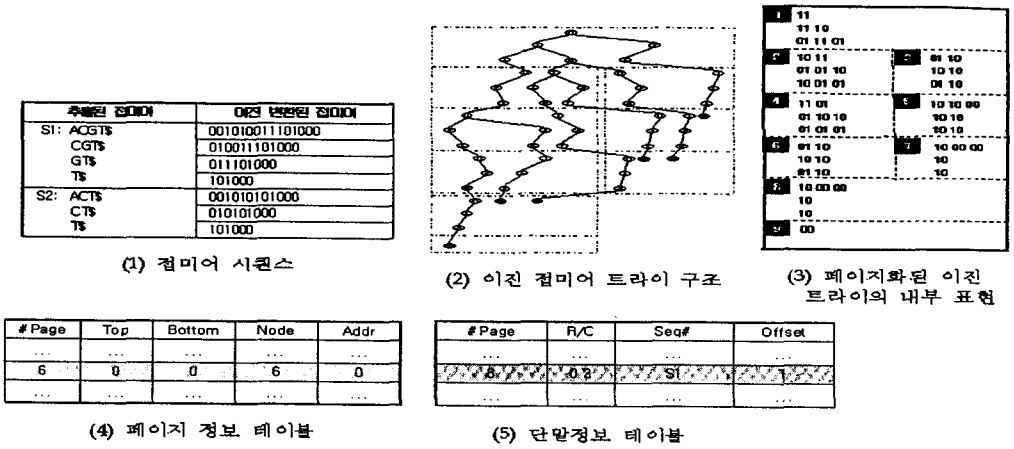
2.2 트라이 인덱스 구조

2.2.1 트라이 : 트라이[5] 구조는 다수의 시퀀스들을 인덱싱하기 위하여 사용되며, 주어진 질의 시퀀스와 정확히 일치하는 시퀀스의 위치를 신속하게 찾도록 하는데 유용하다. 트라이 구조를 이진 트라이 구조로 제한하여 이진 데이터 저장 구조로 사용할 수 있다. 또한, 이진 트라이 구조에서 에지 정보가 '0'이면 왼쪽 에지에, '1'이면 오른쪽 에지에 정보가 저장되는 것을 가정함으로써 에지 정보를 생략 표현할 수 있다. 이와 같이 형성된 이진 트라이 구조는 포인터를 사용하지 않는 이진 데이터 표현이 가능하다. 즉 '01'은 노드에 오른쪽 에지만이 연결된 형태를 표현하고, '10'은 노드에 왼쪽 에지만이 연결된 형태를 표현하고, '11'은 노드에 왼쪽 에지와 오른쪽 에지가 모두 연결된 형태를 표현하고, '00'은 에지가 연결되지 않은 단말노드의 형태를 표현한다.

2.2.2 인덱스 구성 전략 : 부분 시퀀스 검색을 지원하기 위하여 인덱스의 기본 구조로서 접미어 트라이[5] 구조를 사용한다. 여기에서는 S1='ACGT'와 S2='ACT'의 두 개의 DNA 시퀀스에 대한 경우를 예를 들어 단계적으로 인덱스 구성 절차를 설명한다.

(1) 인덱스 압축 효과를 얻기 위하여 DNA 시퀀스가 소수의 문자만으로 구성된 시퀀스라는 점에 착안하여 시퀀스 내에 출현하는 각 문자를 최소의 비트량으로 표현한다. 각 시퀀스로부터 모든 접미어 시퀀스를 추출하여 각 문자당 3트를 할당하여 이를 이진 시퀀스로 변환하면 (그림 1)의 (1)과 같은 결과를 얻을 수 있다. 여기에서 '$' 문자가 각 시퀀스를 유일하게 구분하기 위하여 사용됨을 알 수 있다.

(2) 각 이진 접미어 시퀀스를 정렬하여 순서대로 트라이 구조에 삽입한다. (그림 1)의 (2)는 이와 같은 과정에 의하여 얻어



(그림 1) 트라이 인덱스

진 이진 트라이 구조를 보이고, (3)은 페이지 단위로 저장된 이진 데이터를 나타낸다. 여기에서 (그림 1)의 (2)와 (3)의 점선은 디스크 내의 페이지 분할 저장 상황을 나타낸다. 예를 들어 페이지 크기를 16비트로 가정하여, 한 페이지에 저장할 수 있는 최대 트리의 레벨 수를 3으로, 페이지 내에 저장될 수 있는 최대 노드 수를 8로 가정한 경우, 그림에서 보인 바와 같은 디스크 분할 결과를 얻게 된다.

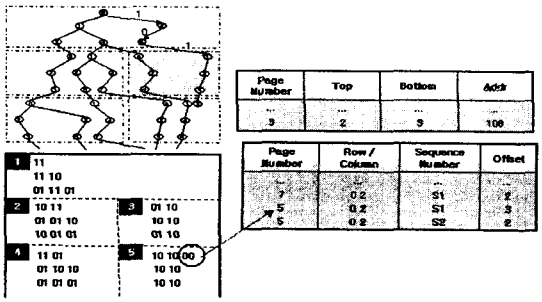
(3) 접미어 트라이 인덱스는 디스크 페이지에 분할되어 저장되므로, 이 인덱스를 이용하여 임의의 경로를 검색하기 위하여는 노드와 노드 사이의 페이지 연결 상태를 나타내는 정보가 필요하다. 페이지 연결 정보는 페이지 번호(#Page), 각 페이지에 유입된 에지의 수(Top), 각 페이지로부터 나간 에지의 수(Bottom) 등으로 이루어지며, (그림 1)의 예의 경우 (4)와 같은 페이지 정보가 필요하다.

(4) 마지막으로 접미어 시퀀스의 출현을 나타내는 단말 노드 정보(시퀀스 번호, 오프셋 정보)를 저장한다. 이진 데이터 상에는 시퀀스의 단말정보를 직접 기입할 수 없으므로 이를 별도의 테이블 형태로 저장하여야 한다. (그림 1)의 (2)의 트라이 구조에서 검게 표시된 노드는 단말 노드를 나타내며, 이들 정보를 (5)의 형태로 저장한다. 여기에서 #Page는 페이지 번호를 나타내며, R/C는 단말 노드가 페이지 내에 출현한 위치(행/열) 정보를 나타내고, Seq#와 Offset은 시퀀스 식별자와 오프셋 정보를 나타낸다.

3. 질의 처리

3.1 질의처리 과정

2장에서 제안한 접미어 트라이 인덱스 구조는 노드 사이의 포인트 정보를 내부적으로 포함하지 않으므로, 트라이 구조 상의 노드 연결 정보 등을 직접적으로 얻을 수 없다. 따라서, 주어진 질의와 일치하는 부분 시퀀스를 인덱스 내에서 검색하기 위하여는, 트라이의 레벨 정보, 페이지 사이 혹은 페이지 내에서의 노드의 연결 상황 등을 인덱스 정보를 이용한 연산과정 등을 통하여 얻어야 한다. 예를 들어, 인덱스에 저장된 2비트의 이진 노드 정보로부터 각 노드에 연결된 에지 정보를 얻을 수 있으므로, 이를 이용하여 트라이의 각 레벨에 존재하는 노드 수 등을 계산하여 낼 수 있다. 2장에서 사용한 예제를 이용한 질의 처리 과정을 (그림 2)에 보인다. 질의 시퀀스로서 'T'가 주어지면, 이 질의는 '101'로 변환되며, 루트 페이지의 처음 노드로부터 검색을 시작한다. 이 경우, 질의의 처음 비트가 '1'이고 루트 노드 정보가 '11'이므로 다음 검색 대상 노드는 현재의 루트 노드로부터 2만큼 떨어진 2번 노드임을 연산에 의하여 얻을 수 있다. 이와 같은 과정에 의하여 질의 비트와 인덱스 정보를 비교하여 가게 되며, 인덱스의 페이지 교체 연산에는 페이지 연결정보(페이지 정보 테이블)를 이용한다. 이 예에서 질의비트를 마지막까지 처리하여, 3번 페이지의 1번 노드에 도달하면, 질의 시퀀스 검색이 완성되고, 그 다음 단계에서는 현재의 3번 페이지 1번 노드의 모든 서브 트리를 검색하여 단말 노드 정보를 취합하는 과정이 필요하다. 즉 검색된 질의 시퀀스를 부분 시퀀스로 포함하는 모든 시퀀스 정보와 오프셋 정보가 최종 결과가 되게 된다. 이 경우, 단말 정보 테이블로부터 (S1, 3)과 (S2, 2)를 얻게 된다.



(그림 2) 질의 처리 과정의 예

3.2 단말노드 정보 검색

위 질의 처리 과정 중 이진 트라이의 중간 노드 N에서 질의 시퀀스가 검색 완료되어, 해당 노드 N의 모든 서브 트리를 검색하여 단말 노드 정보를 가져오는 단계에 주의를 기울일 필요가 있다. 그 이유는 검색 대상의 인덱스 크기가 방대하고, 질의 시퀀스의 길이가 짧은 경우, 이 과정은 인덱스의 검색 공간을 확대시켜, 알고리즘 성능 저하의 주요 원인으로 작용할 수 있기 때문이다. 이진 트라이 인덱스 구조 상에서 이와 같은 단말 노드 정보를 효율적으로 검색하기 위하여 다음 두 가지의 서로 다른 접근 방식을 사용할 수 있다.

(1) B$^+$-트리를 이용한 단말 노드 정보 검색

앞의 예제에서 사용한 단말 정보 검색 방식을 그대로 사용하

는 기법이다. 중간 노드 N의 서브 트리를 검색하여 모든 단말 노드를 검색하는 과정은 깊이 우선 방향으로 진행된다. 각 노드의 방문 순서를 고려하여 단말노드 테이블을 B$^+$-트리 구조로 구성하여, 단말 정보의 고속 검색을 지원한다.

(2) 다차원 인덱스를 이용한 단말노드 정보 검색

이진 트라이 인덱스를 사용하여 중간 노드 N 밑에 있는 서브 트리를 순차적으로 검색하는 과정을 생략하고, 단말 정보를 직접 얻는 기법이다. 단말 노드 정보를 별도의 다차원 인덱스에 저장하여, 단말 정보의 고속 검색을 지원한다. 단말 정보는 시퀀스 식별자와 오프셋 정보를 가지는 단말 정보 리스트와 이를 위한 다차원 인덱스로 이루어진다. 다차원 인덱스의 구성 방식과 이를 이용한 정보 검색 방식은 다음과 같다.

(2-1) 각 단말 노드 정보를 정해진 수 k개의 정수로 표현하여 k 차원의 다차원 인덱스에 저장한다. 만약 단말노드를 나타내는 비트 패턴이 k개의 정수 표현보다 짧을 때는 그 뒤에 0을 계속 붙여 표현한다. 만약 단말 노드를 나타내는 비트 패턴이 k개의 정수 표현보다 길 때는 그 뒷 부분을 잘라 버린다.

(2-2) 트라이 인덱스의 루트로부터 중간 노드 N을 연결하는 경로가 주어진 질의를 만족한다면, 그 경로의 길이에 따라 다음 3가지 중 하나를 선택하여 질의를 처리함으로써, 단말정보를 얻는다.

① 경로 p의 길이가 k-정수 보다 짧을 때: p 뒤에 0을 계속 붙여 만든 k-정수 값을 p0라고 하고, p뒤에 1을 계속 붙여 만든 k-정수 값을 p1 이라고 하자. 다차원 인덱스를 검색하여 p0와 p1 사이에 존재하는 모든 단말 노드들을 검색한다.

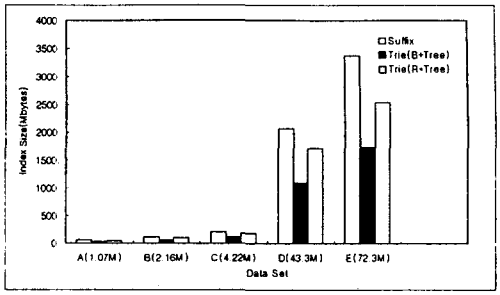
② 경로 p의 길이가 정확하게 k-정수일 경우: 다차원 인덱스를 검색하여 p 값을 가지는 모든 단말 노드들을 검색한다.

③ 경로 p의 길이가 k-정수 보다 길 때: p를 k-정수로 표현한 값을 pt 라고 하자. 다차원 인덱스를 검색하여 pt 값을 가지는 모든 단말 노드들을 검색한다. 그 후 후처리과정에 의하여 잘못 검색된 단말노드를 제거한다.

4. 성능 평가

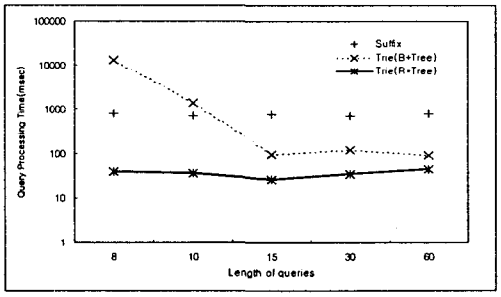
제안된 기법의 성능을 평가하기 위하여 인덱스의 크기 및 질의 처리 시간을 기존 방식과 실험을 통하여 비교 분석한다. 비교 대상의 인덱스 구조로는 디스크 기반의 접미어 트리를 사용한다. 실험 데이터로는 GenBank[6]로부터 다운 받은 Human Chromosome 18~21번의 5가지 DNA 시퀀스를 사용하였다. 시퀀스 내에 출현하는 서로 다른 문자수는 7~8개이다. 실험을 위한 하드웨어 플랫폼으로는 Windows 2000 Server를 운영체제로 사용하고, 1GB의 주기억장치, 40GB 디스크를 갖는 Pentium IV 2GHz의 PC를 사용한다.

실험 1 (인덱스 크기 비교) : 제안된 인덱스 구조에서는 접미어 트라이 인덱스 외에 페이지 정보와 단말노드 정보를 필요로 한다. 또한 단말노드의 효율적 검색을 위하여 두가지의 서로 다른 인덱스를 사용할 수 있다. B$^+$-트리를 이용한 방식에서는 단말노드의 페이지 정보에 대하여 B$^+$-트리 인덱스를 구성하였다. 다차원 인덱스를 이용한 방식에서는 다차원 인덱스서 현재 시퀀스 데이터베이스 분야에서 가장 널리 사용되고 있는 R*-트리[7]를 사용하였으며, 인덱스 엔트리로서 각 단말 노드를 2개의 정수로 표현하였다. (그림 3)에 제안된 인덱스의 크기를 기존의 접미어 트리 인덱스 기법에 의하여 생성된 인덱스 크기와 비교한 결과를 보인다. (그림 3)에서 Trie(B+Tree)와 Trie(R*Tree)는 단말노드 검색 방식으로 각각 B$^+$-트리 방식과 R*-트리 방식을 사용한 경우를 나타내며, Suffix는 비교 대상의 접미어 트리 검색 방식을 나타낸다. 실험 결과로부터 제안된 방식 중, Trie(B+Tree)는 Suffix에 비하여 약 50%의 저장 공간 감소 효과를 나타내고, Trie(R*Tree)는 Suffix에 비하여 약 30%의 저장 공간 감소 효과를 나타내는 것으로 나타났다.



(그림 3) 데이터 크기에 따른 인덱스 크기 비교

실험 2 (질의처리 시간 비교) : 제안된 두 가지 방식에 의한 질의 처리 시간을 기존의 접미어 트리 방식과 비교한 결과를 (그림 4)에 보인다. 실험 데이터로는 D(43.3M)를 사용하였으며, 질의 길이 변화에 따르는 질의 처리 시간을 비교한 결과이다. Y축은 로그 스케일링 되어 있으며, 측정 단위는 msec이다. 실험 결과에 의하면 Trie(B+Tree)는 질의 길이가 짧은 경우, 단말노드 검색을 위한 트라이 인덱스 검색 시간이 증가하여 전체 검색 시간이 증가하지만, 질의 길이가 긴 경우, 질의처리 시간이 현저히 감소함을 볼 수 있다. 반면에 Trie(R*Tree)는 질의 길이에 무관하게 거의 일정한 질의 처리 시간을 나타내며, Suffix에 비하여 약 20배의 성능 감소 효과를 나타내는 것으로 나타났다.



(그림 4) 질의 길이 변화에 따른 질의처리 시간 비교

참고 문헌

[1] E. Hunt, R. W. Irving, and M. P. Atkinson, "Persistent Suffix Trees and Suffix Binary Search Trees as DN Sequence Indexes," Technical report, Univ. of Glasgow Dept. of Computing Science. TR-2000-63, 2000.

[2] T. Kahveci, A. K. Singh, "An Efficient Index Structur for String Databases," In Proc. Int'l. Conference on Very Large Data Bases, VLDB'01, pp. 351-360, 2001.

[3] H. E. Williams and J. Zobel, "Indexing and Retrieval for Genomic Databases," TKDE, 2002.

[4] 원정임, 박진만, 윤지희, 박상현, "대규모 DNA시퀀스를 위한 인덱스 구조", In Proc. KDBC 2003, pp. 87-94, 2003.

[5] G. A. Stephen, String Searching Algorithms, World Scientific Publishing, 1994.

[6] http://www.ncbi.nlm.nih.gov

[7] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger "The R*-tree: An Efficient and Robust Access Method f Points and Rectangles," In Proc. Int'l conf on Management of Data, ACM SIGMOD, pp. 322-331, 1990.