프로젝트

설계보고서

<2018.6.12>

컴퓨터 알고리즘과 실습 03분반

주종화 교수님

컴퓨터공학과

2015111489 김성현

1. 팀구성

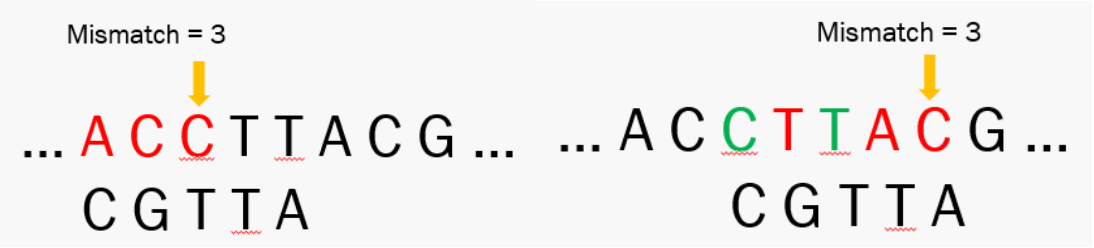
2015111489 김성현

2. 프로젝트 목표

인간의 유전자 DNA는 상상할 수 없을 정도로 많은 수를 가지고 있다. 이러한 유전자는 각 인간의 특징과 건강사항 등을 결정짓는데 큰 역할을 한다. 따라서 인간의 DNA를 분석하는 것이 대두되고 있다. 30억의 수 많은 DNA중에서, 모든 인간이 가지고 있는 99.9%가 아닌 다른 구조를 가진 0.1%의 DNA를 찾아내어 인간의 특징과 건강을 예측할 수 있게 된다. 따라서 그러한 다른 DNA를 찾아내기 위해서 이번 프로젝트에서는 A/C/G/T로 구성되어 있는 랜덤한 n개의 문자열을 생상하여 gemone을 만든 뒤, 특정 l길이의 short read를 m개 잘라내서 생성한다. 또한 genome과 0.1%가 다른 새로운 reference를 만들어, short read와 reference를 이용하여 원래의 genome을 복구하는 것이 목표이다.

3.Bench Mark Algorithm

벤치마크로 사용한 알고리즘은 KMP 알고리즘이다. KMP알고리즘은 Θ(n)의 matching time complexity를 가진다. 하지만 전수조사를 하는 Trivial Mapping 방법은 Θ(nm)의 time complexity를 가진다. 따라서 KMP 알고리즘은 전수조사의 방법에 비해서 더 좋은 시간 효율을 가지고 있다. 하지만 KMP는 mismatch를 고려해서 string matching을 실행할 때, 한번 지나간 string에 대해서는 다시 탐색하지 않아서 정확도가 떨어지는 경우가 있다. 가령 CGTTA인 pattern과 ACCTTACG의 string을 matching하고 있다면, <그림1>과 같이 mismatch가 3이 되는 순간 KMP검색 방법에 의하여 틀린 글자인 C와 같은 patter의 첫번째 글자 C로 돌아가서 <그림2>와 같이 KMP 탐색을 다시 시작한다. <그림2>에서도 mismatch가 3이되어 pattern을 찾아내지 못했다. 하지만 KMP가 건너뛴 부분을 보면 CGTTA와 mismatch가 1값을 갖는 string을 <그림3>과 같이 찾을 수 있다. 이러한 경우 때문에 KMP의 정확도가 떨어지게 된다. 정리하자면, KMP는 최악의 경우에 대한 좋은 효율성을 가지고 있기 때문에 Trivial에 비해서 정확도는 떨어지지만 시간적으로 좋은 알고리즘이라는 생각을 하여서 선택하였다.



<그림 1>

<그림2>

<그림 3>

4. My Algorithm

앞서 3.BenchMark Algorithm에서 말한 것과 같이 KMP는 시간적인 면에서 효율적이지만 정확도가 떨어진다는 단점을 가지고 있다. 하지만 우리가 목표로 하는 genome 복구는 정확도와 복구하는 정도가 굉장히 중요하기 때문에 KMP의 알고리즘을 이용하여서 genome복구를 하는 것은 좋지 못하다고 판단되었다. 따라서 KMP의 방법과 같이 시간을 빠르게 이용하여 탐색할 수 있는 방법을 토대로, KMP 검색 시에 놓칠 수 있는 부분에 대하여 Alpha skip searching Algorithm을 이용하여 탐색하는 알고리즘을 생각하게 되었다.

Alpha searching Algorithm은 pattern에 대하여 3글자씩 잘라내어 각 3글자로 잘라낸 pattern의 위치를 Tries data structure을 이용하여 저장한다. 이후에 특정 위치마다 Tries를 검색하여 pattern이 존재하는지 여부를 확인하고 존재할 시에 해당 위치에서의 string 탐색을 시도한다. Alpha searching을 실행하는 i의 위치는 다음과 같은 수식을 이용하여 구하게 하였다.

$$location_n = m + 6(n+1) - 9 \ (\mathrm{n} \geq 0)$$

다음의 수식에 존재하는 글자 뒤로 2글자를 포함한 3글자를 pattern의 위치를 저장한 자료구조에서 찾아내서, 일치하는 pattern 이전과 이후의 글자를 string matching하여 KMP가 놓칠 수 있는 pattern matching을 찾아내도록 한다.

5. Result

KMP와 Alpha skip searching을 이용한 알고리즘을 실행한 환경은 다음과 같다.

컴퓨터	**LG-PC**
운영 체제	Windows 8.1 K 64비트
BIOS	14Z950FN X64
프로세서	Intel® Core™ i5-5200U CPU @ 2.20GHz
메모리	8192MB RAM

DirectX	11

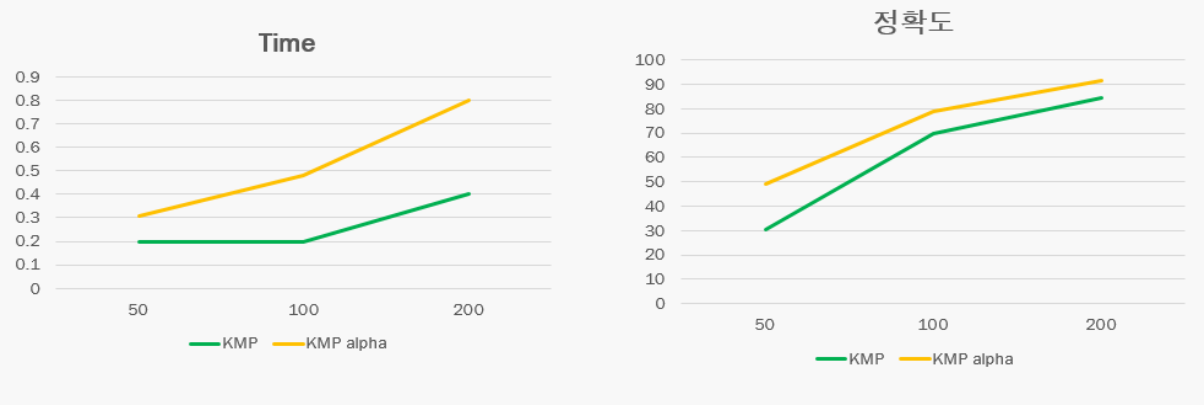
input으로 실행창에 N의 개수와 Shortread의 개수인 m을 입력하면 다음과 같이 실행하게 된다.

```
N = 1000
shortread_num = 100
```

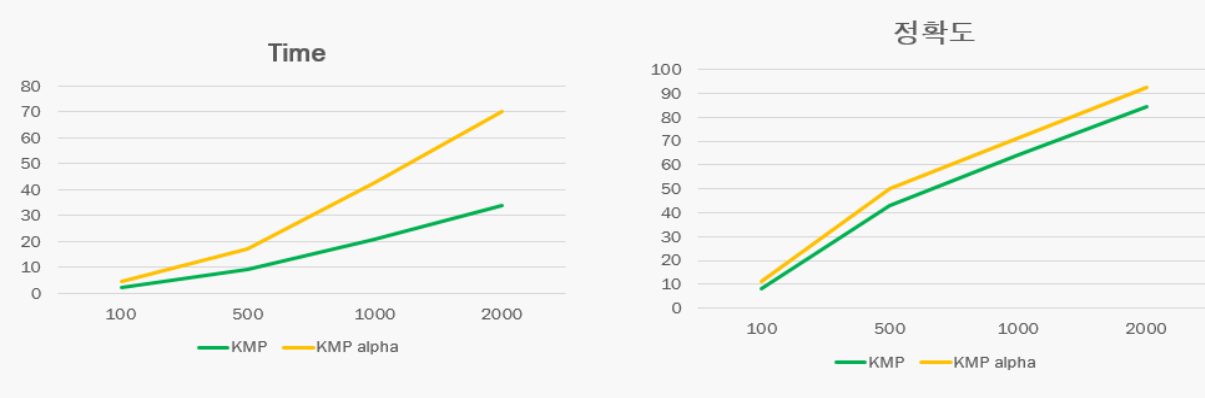
Kmp_result.txt와 kmp_alpha_result.txt를 생성하며, 생성한 txt는 다음과 같다. 이 때, B는 찾지 못한 string을 의미한다.

<kmp_alpha_result.txt> **<kmp_result.txt>**

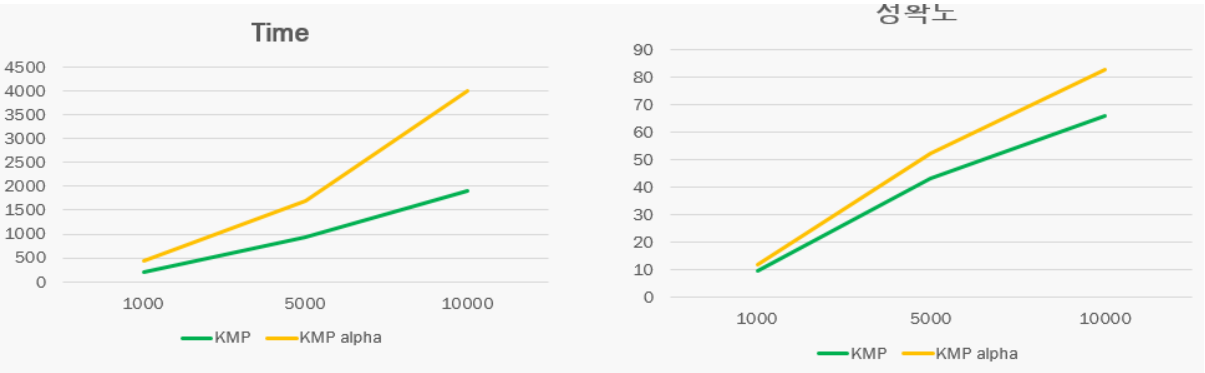
아래의 그림은 각 n과 m의 개수에 따른 KMP와 KMP-Alpha skip searching에 따라서 나타나는 것을 표로 나타낸 것이다.



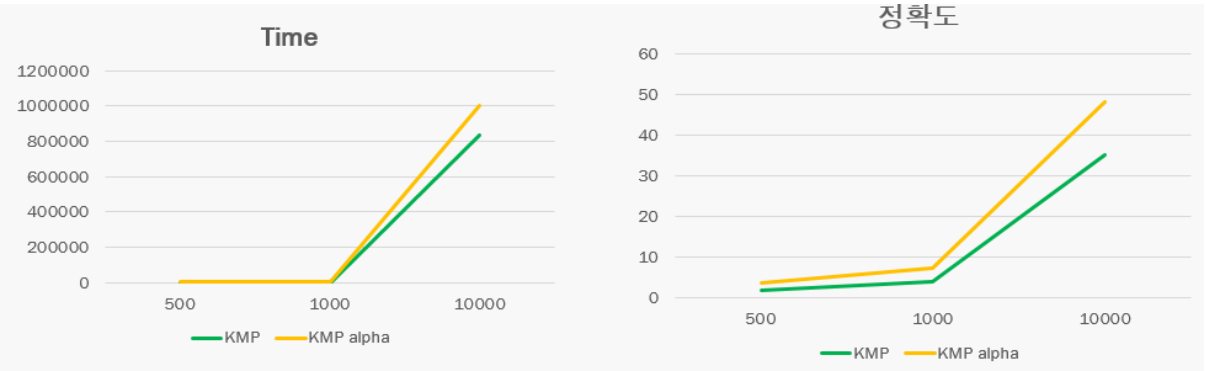
<N = 1,000, M = 50~200, L =20~30, D = 2>



<N = 10,000, M = 100~2,000, L =20~30, D = 2>



<N = 100,000, M = 1000~10,000, L =20~30, D = 2>



<N = 1,000,000, M = 500~10,000, L =20~30, D = 2>

위의 결과에 의해 time complexity는 alpha skip searching을 위한 pattern 관련 Tries 생성시간인 O(m)이며 searching time은 O(nm) 이하 이게 된다. Space는 O(m)을 사용하게 된다.

6.알고리즘의 장단점 및 향상시킬 점

KMP의 단점을 상쇄시키려 만들게 된 KMP와 Alpha skip searching을 이용한 방법은 원하던 목표인 정확도를 향상시키는 점은 만족시킬 수 있었다. 하지만 KMP의 장점이던 시간효율면에서 시간을 증가시키는 단점이 생기게 되었다. 이러한 단점이 나타나게 된 점을 생각 해 보았을 때, Alpha skip searching을 실행하기 위하여 생성한 Tries 자료구조를 만드는 방법에 시간과 메모리를 많이 사용하게 되었기 때문이라는 생각이 들었다. 해당 알고리즘을 구현할 때, class와 배열을 이용하여 만들었다. 하지만 이 방법을 이용하면 pattern의 길이가 길어질수록 메모리와 Tries 생성시간, string을 비교하는 시간이 길어지게 되었다. 따라서 앞으로 이 알고리즘을 향상시키기 위해서는 Tries생성을 하는 방법을 효율적으로 수정하는 것이다. Linked list혹은 Boolean을 이용하여 A/C/G/T를 적절하게 나누어 Tries를 생성할 수 있는 방법을 고안한다면 시간적인 효율을 향상될 것을 기대할 수 있다.