|  |
| --- |
| **알고리즘 9조 보고서**  기말 프로젝트 |

원, 폰트, 상징, 로고이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

|  |  |
| --- | --- |
| **과 목** | 알고리즘 3분반 |
| **교 수** | 주종화 교수님 |
| **제출일** | 6월 16일 |
| **조 이름** | 9조 |
| **조장** | 2022110151 이주연 |
| **조원** | 2023111033 김태은  2021112504 박지우  2022113556 정태호 |

|  |
| --- |
|  |

|  |
| --- |
| **목차**   1. **서론** 2. **문제 정의** 3. **데이터 출처** 4. **벤치마킹 알고리즘** 5. **개별 구현 알고리즘** 6. **이주연** 7. **구현 알고리즘 구조** 8. **컴퓨터 사양에 따른 결과 분석** 9. **성능 분석** 10. **구현 알고리즘의 장단점** 11. **김태은** 12. **구현 알고리즘 구조** 13. **컴퓨터 사양에 따른 결과 분석** 14. **성능 분석** 15. **구현 알고리즘의 장단점** 16. **박지우** 17. **구현 알고리즘 구조** 18. **컴퓨터 사양에 따른 결과 분석** 19. **성능 분석** 20. **구현 알고리즘의 장단점** 21. **정태호** 22. **구현 알고리즘 구조** 23. **컴퓨터 사양에 따른 결과 분석** 24. **성능 분석** 25. **구현 알고리즘의 장단점** |

1. **서론**
   1. **문제 정의**

본 프로젝트의 핵심은 아이의 전체 유전자 시퀀스를 복원하고, 이를 부모 시퀀스와 비교해 친자를 판별하는 알고리즘을 설계하는 데 있다. 이를 위해 본 조는 부모 유전자를 참조 시퀀스로 설정하고, 여러 후보 아이를 두어 각 아이로부터 짧은 리드(short reads)를 추출하고자 한다. 주어진 후보 아이들의 짧은 리드를 바탕으로 전체 시퀀스를 복원하여, 참조 시퀀스인 부모 시퀀스와 비교하고 부모의 친자가 누구인지 후보 아이들 중에 판별하고자 한다.

이를 위해 설정한 고려사항은 개인 알고리즘 별로 상이하다. 부모 시퀀스 길이이자 복원할 아이 시퀀스 길이 N, 한 아이로부터 추출되는 짧은 리드 개수 M, 짧은 리드당 길이 L, 허용할 미스매치 수 D는 추후 알고리즘 구조와 함께 개별로 설명하고자 한다.

* 1. **데이터 출처**

부모 시퀀스의 경우, 염기서열 A,C,G,T 4가지 글자를 활용해 임의로 N길이만큼 생성하고자 한다. 파이썬을 활용한 알고리즘 경우에는 random, math 모듈을 활용하여, c언어와 C++를 활용한 알고리즘 경우에는 <stdlib.h>와<time.h>, <cstdlib.h>와 <ctime.h> 헤더 파일 속 rand() 함수를 사용하여, 난수를 생성하였다.

후보 아이의 short read는 알고리즘마다 서로 다른 방식으로 추출되었으며, 각 알고리즘 구조 설명 시 생성 방식을 함께 상세히 설명하고자 한다.

* 1. **벤치마킹 알고리즘**

본 조는 Trivial mapping algorithm을 구현 알고리즘 성능을 평가하는 기준 알고리즘으로 설정하였다. Trivial mapping algorithm은 short read를 부모 시퀀스 전체에 슬라이딩하며 하나하나 일치 여부를 단순 비교해, 정렬 없이 매핑하는 가장 기본적인 알고리즘이다. 동작 방식이 직관적이고 단순하다는 장점이 있지만, 연산 시간이 오래 소요되고 메모리 사용량이 많다는 점에서 큰 한계를 지닌다. 본 프로젝트는 알고리즘을 스스로 설계하고 직접 구현하는 과정인 만큼 단순 구현보다도 시간과 메모리 효율성의 중요성을 부각시키기 위해 Trivial Mapping 알고리즘과 비교하고자 한다.

이에 따라, 본 보고서에서는 알고리즘 성능 분석과 관련하여, 각 개별 팀원이 구현한 알고리즘의 최악의 시간 복잡도, 공간 복잡도(제자리성), 안전성을 중심으로 분석한 내용을 설명하고자 한다.

1. **개별 구현 알고리즘**
   1. **이주연**
      1. **구현 알고리즘 구조**

알고리즘 구조 설명에 앞서, 설계를 위한 초기 설정은 다음과 같다.

* N=3,000,000: 부모 및 아이 시퀀스 길이
* M=10,000: 한 아이로부터 추출되는 short read 개수
* L=32: short read의 길이
* D=3: 허용되는 최대 mismatch 수
* NUM\_CANDIDTAES = 101: 아이 수 (친자 1명 + 비친자 100명)

컴퓨터 사양 내에서 유의미한 결과값을 도출하기 위해, 5분 이내로 실행 시간이 제한될 수 있는 범위 내에서 위와 같이 임의 값으로 초기값을 설정하였다.

본 알고리즘은 “2-bit 인코딩 처리”, “Majority Vote기반 복원”의 2가지 핵심 기법을 바탕으로, 다음과 같은 4단계로 이루어진다.

* 1단계) 부모 DNA 시퀀스를 생성하는 단계
* 2단계) 친/양자 아이의 short read를 생성하는 단계
* 3단계) 아이 DNA 시퀀스를 복원(reconstructing)하는 단계
* 4단계) 복원한 아이 시퀀스와 부모 시퀀스를 비교(compare)하고, 유사도를 계산해 최종 친자 여부를 판별하는 단계

먼저 첫 번째 단계는 전체 친자 판별의 기준이 되는 부모 시퀀스를 무작위로 생성하는 단계이다. 시퀀스 길이 N은 3,000,000bps을 초기값으로 설정했고, ‘A’, ‘C’, ‘G’, ‘T’ 염기를 난수로 선택해 배열에 저장했다. 이때, 메모리 효율성을 높이고자 각 1byte(8비트)를 차지하는 염기 문자를 2비트로 인코딩해 parent\_seq 배열에 저장하였다. A의 경우에는 00, C는 01, G는 10, T는 11로 변환해 시퀀스 전체를 압축하여 처리할 수 있도록 설정하였다. 이때, 부모 시퀀스가 성공적으로 생성되었는지를 확인하기 위해 parent.txt파일로 저장해 확인하였다.

두 번째 단계는 후보 아이들의 short read를 생성하는 단계이다. 첫 초기 값으로 총 101명의 아이 후보를 두었고, 이 중 1명은 실제 친자, 나머지 100명은 일반 아이로 가정하여 데이터를 구성하였다. 실제 친자와 친자가 아닌 후보는 서로 다른 방식으로 short read가 구성된다.

친자의 경우, 친자 short reads는 부모 dna로부터 일정간격으로 추출돼, 염기에 일부 돌연변이를 넣는 방식으로 생성된다. 부모 시퀀스 전체 길이(N)를 short read 개수 (M)만큼 균등하게 분할한 후, 각 분할 구간의 시작위치(pos)로부터 길이 32의 short read를 추출해 생성된다. 이때, 시작 위치(pos)와 간격(step)은 다음과 같이 계산한다.

=

이렇게 생성된 short read는 다시 4개의 파트로 나누고, 4 파트 중 3개의 파트에 무작위로 돌연변이를 삽입해, 부모와 유사하면서도 돌연변이로 인해 독립적인 아이의 short read 데이터가 생성될 수 있도록 설정하였다.

세 번째 단계는, 각 후보 아이의 short reads를 가지고, 아이 전체 dna 시퀀스를 복원하는 단계이다. 이때, 1981년 보이어 무어에 의해 제안된 majority vote (최빈값을 선택하는 알고리즘) 아이디어를 착안하여 복원하는 방식을 고려하였다.

구체적으로, 하나의 위치를 여러 read가 동시에 덮을 수 있다는 점을 활용하여, 각 위치에 놓이는 A,C,G,C 염기의 빈도를 배열에 누적한 후, 가장 많이 등장하는 염기를 해당 위치의 최종 복원값으로 설정하는 방식이다. 예를 들어 특정 위치에 READ가 총 40번 겹쳤고, 이 중 ‘A’는 35번, ‘C’는 2번, ‘G는 3번 겹쳐졌다면, 해당 위치를 A로 복웒나다. 이와 같은 방식을 활용하여, 일부 READ에 돌연변이 혹은 노이즈가 발생하더라도, 안정적으로 복원을 할 수 있도록 표현하였다.

마지막 단계는 복원한 아이의 시퀀스와 부모 시퀀스를 비교해 유사도를 게산하고, 친자인지 최종 판별하는 단계이다. 이때, 비교 시 복원된 시퀀스에서 복원되어 채워진 위치만을 가지고 유사도를 계산하며, 동일한 염기의 수를 전체 비교 가능한 염기 수 로 나누어 백분율 형태로 유사도를 표현한다. 모든 후보 아이들에 대해 유사도를 계산한 후, 가장 높은 유사도가 나오는 후보를 골라 친자로 판별한다. 그 결과를 result.txt에 저장하여, 가장 높은 유사도가 나온 후보가 누구인지 가시적으로 확인할 수 있도록 하였다.

* + 1. **컴퓨터 사양에 따른 결과 분석**

**컴퓨터 사양**

* CPU : Intel 11세대 i5 (4코어 8스레드)
* RAM : 16GB
* SSD : 256GB
* 운영체제 : Windows 10 (64bit)

본 알고리즘은 위와 같은 사양 환경 내에서 이루어졌다. 노트북 내에서 성능은 준수해 중간 규모 데이터 처리에는 무리가 없지만, 고용량 데이터 특히 배열을 반복적으로 처리할 때에는 메모리 및 연산 측면에서 무리가 갈 수 있는 수준이다.

**메모리 사용량 및 N 설정 기준**

본 알고리즘에서 복원을 위해 핵심적으로 사용하는 Majority Vote(최빈값) 기반 변형 아이디어는 모든 short read가 덮는 위치마다 ACGT염기의 빈도 수를 저장하는 배열을 필요로 한다(dna\_counts). 각 위치마다 카운트를 저장하는데 4 \* sizeof(int)로 총 16byte가 필요하며, 8btye인 read 정보와 4바이트인 위치 정보까지 고려 했을 때 N=3,000,000일 때 약 36MB 이상의 메모리가 필요함을 알 수 있었다. 따라서, 초기값 3,000,000을 기준으로 10배를 줄이고, 10배를 늘리는 방식을 통해 분석을 진행하였다. 텍스트, 스크린샷, 라인, 도표이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

위 표는 시퀀스 길이 N이 변화함에 따라 소요되는 알고리즘 수행 시간을 나타낸 표이다. M을 1만으로 고정했을 때, N이 30만일 때에는 2.793초가, N이 300백만일 때에는 19.22초가, N이 3000만일 때에는 121초가 소요된 모습을 볼 수 있다.

텍스트, 스크린샷, 도표, 직사각형이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

이와 반대로 N을 고정하고, short reads의 개수인 M=10,000 초기값을 10배 줄이고, 늘렸을 때에 위와 같이 시간 소요가 발생한다. M을 1,000으로 설정했을 때는 17.51초로 조금 더 빠르게 수행되었고, M=10,000일때는 19.22초, M=100,000일 때에는 22.34초로 조금 더 오래 걸린 결과를 확인할 수 있었다. 이는 M이 작을 때에는 대부분의 배열 특히 dna\_counts최빈값을 기록하는 배열이 cpu 캐시에 머무르기 때문에 처리 속도가 빨랐기 때문이었는데, 반대로 M이 커질수록 처리할 short reads 양이 많아져, CPU 캐시를 넘는 메모리 접근이 종종 발생해 속도가 저하되었음(메모리 병목 현상)을 알 수 있었다. read수가 많아질수록 실제 복원 대상 위치가 늘어나고, 최빈값을 저장하는 카운트 배열 접근 횟수도 증가하며 이와 같은 결과가 나타났음을 확인할 수 있었다.

다만, 유사도 결과를 비교하였을 때 실험 결과는 다음과 같은 경향을 보였다. 친자로 설정한 후보는 항상 90% 이상의 유사도를 보였고, 친자가 아닌 후보들 100명은 대부분 24-25% 수준의 유사도를 기록하였다.

텍스트, 스크린샷, 폰트, 흑백이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.텍스트, 스크린샷, 메뉴, 흑백이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

이와 같은 결과는 다음과 같은 이유 때문인데, DNA 염기 조합은 총 4종류이고, read를 완전히 랜덤으로 생성할 때에 단일 염기가 부모 dna와 일치할 확률이 1/4로 약 25%다보니, 실제 친자 검사시 친자가 아니더라도 기본 검사에서 평균적으로 80%이상 나오는 것에 비해 확연히 낮은 유사도 비율이 나온 것을 분석할 수 있다. 비친자의 경우 유사도 비율은 낮았지만, 알고리즘 구조를 고려하였을 떄, 이론적으로 기대한 랜덤 dna의 기댓값과 유사하였기 때문에 복원은 제대로 되었음을 확인할 수 있었다.

* + 1. **성능 분석**

본 프로젝트는 기존 trivial mapping 알고리즘에 비교했을 때 다음과 같은 성능이나타났다.

**텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

시간 복잡도의 경우, short read 데이터를 재구성할 때에 short read개수(M)만큼 순회하고, parent DNA 길이(N)만큼 순회하다보니 이를 더한 (M+N)에 후보 수만큼 곱한 O(NUM\_CANDIDATES\*(M+N))의 시간 복잡도가 나타나게 되었다.

공간 복잡도의 경우에는 short read M개 \* L 길이만큼 메모리에 저장되는 것에 더해, 재구성을 위한 seq배열과 dna\_counts(최빈값 횟수를 저장하는)배열이 추가로 필요해 +N하여 O(N+M\*L)만큼 공간 복잡도가 나타나는 것을 알 수 있었다. 추가 배열이 필요한 만큼 추가 메모리를 사용하기 때문에 제자리성을 만족하지 못했고, 안전성의 경우, 재구성할 때에 majority vote 아이디어를 바탕으로 구현하였기에 정렬과 개념이 달라 모호하다고 판단하였지만, 편의성을 위해 최빈값 등장시 A,C,G,T 염기서열 순서대로 저장되기 때문에 만족하는 방향임을 분석할 수 있었다.

* + 1. **장단점 및 향후 개선 방안**

먼저, short read가 부모 시퀀스에 정해진 위치(start\_pos)에 매핑되기 떄문에 슬라이딩 윈도 탐색이 불필요하여 분석이 빠르다는 장점이 있다. 더욱이 read들이 여러 번 겹쳐 덮기 때문에 오류나 노이즈가 발생해도 복원시 이를 해결할 수 있다는 장점이 있다.

다만, read 위치(start\_pos)이 반드시 필요하기 때문에 데이터를 전처리하는 과정에서 복잡할 수 있고, 추가메모리로 인해 제자리성이 만족되지 않는다. 더욱이 아이 후보 수가 많아질수록 메모리 사용량이 증가하며, 최악의 경우에는 trivial방식과 비슷한 시간 복잡도를 가질 수 있어 위험할 수 있다. 더욱이 동점 처리를 할때에 염기 순서대로 되어 있어 이를 수정할 방안 또한 필요하다.

따라서 개선 방안은 다음과 같다. Tie-break 동점일때 순서를 랜덤으로 선택하여 이를 수정할 수 있고, 메모리 사용 효율을 높이기 위해 n\*4 고정된 크기를 해시맵을 활용하거나, read가 한번도 덮지 않는 위치에 대해서는 메모리 할당을 생략하도록 하여 효율을 높일 수 있다.

* 1. **김태은**
     1. **구현 알고리즘 구조**

본 알고리즘은 크게 두 가지 핵심 기법을 활용한다.

첫째는 K-mer 해싱 기능이다. FastLookup을 통해 참조 서열을 8bp 단위의 작은 조각들로 나누어 해시 테이블에 저장한다. 이와 같이 저장할 경우, 각 k-mer가 어디에 나타나는지를 O(1) 시간에 찾을 수 있어, 전체 검색 시간이 k-mer 개수만큼의 상수 시간으로 줄어든다.

둘째는 비둘기집 원리를 활용한 필터링이다. bidulgiZip 함수에서 구현한 이 방법은 간단하면서도 효과적이다. n개의 오류가 있는 서열을 n+1개 조각으로 나누면, 적어도 하나의 조각은 완전히 일치한다는 수학적 원리를 이용한다. 이 프로그램의 경우에는 mismatch를 3개까지 허용하므로 패턴을 4개 조각으로 나누어 각각을 순차적으로 검사한다.

전체적인 프로그램 흐름을 보면, 먼저 참조 서열로부터 K-mer 해시 테이블을 만들고, Short Read가 입력되면 K-mer 매칭을 통해 후보 위치들을 빠르게 찾는다. 그 다음 비둘기집 필터링으로 불가능한 후보들을 걸러내고, 마지막에 정확한 차이를 계산해서 서열을 재구성한다.

사용 데이터의 경우 부모 DNA는 랜덤생성하며 자식 DNA는 부모 DNA에 랜덤하게 결정된 돌연변이율만큼의 변화를 준다. Short read는 자식 DNA로부터 추출한다.

* + 1. **컴퓨터 사양에 따른 결과 분석**

실제 프로그램 실행 결과는 다음과 같다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

비둘기집 원리의 수학적 효과

비둘기집 원리가 실제로 얼마나 효과적인지 수치로 살펴보자. mismatch를 3개까지 허용할 때 패턴을 4개 조각으로 나누는데, 잘못된 위치에서 8bp가 완전히 일치할 확률은 (1/4)⁸ = 1/65,536이다. 4개 조각 중 하나라도 일치할 확률은 약 4/65,536 정도다.

이는 곧 99.99%의 잘못된 위치 후보들이 조기에 제거된다는 뜻이다. 이 때문에 대부분의 무의미한 계산을 피할 수 있어서 전체적인 성능이 크게 향상된다.

* + 1. **성능 분석**

**시간 복잡도**

시간 복잡도의 경우, 기존의 단순 선형 알고리즘과 비교해보면 성능 차이가 극명하다.

단순 선형 알고리즘의 시간 복잡도는 O(n × m × r × l)이다. 여기서 n은 참조 서열 길이, m은 short read 개수, r은 short read 길이, l은 허용 오차를 의미한다.

반면 본 최적화 알고리즘은 O(n × k + m × r × c × l)의 복잡도를 가진다. k는 k-mer 크기이고, c는 평균 후보 위치 수다.

**실제 성능 측정**

코드에서 사용한 실제 매개변수를 기준으로 계산해보면 성능 개선 효과를 명확히 알 수 있다.

* 참조 서열 길이: 300,000bp
* Short read 개수: 10,000개
* Short read 길이: 32bp
* 허용 mismatch: 3개
* K-mer 크기: 8bp

단순 알고리즘에서는 총 2.88 × 10¹¹번의 연산이 필요하다. (10,000 × (300,000-32) × 32 × 3)

최적화 알고리즘에서는 다음과 같이 계산된다:

* 전처리 단계 (해시 테이블 생성): 3 × 10⁵번
* 후보 검색: 1.25 × 10⁶번
* 필터링: 5 × 10⁶번
* 정확한 비교: 8 × 10⁶번
* 총 연산 횟수: 1.48 × 10⁷번

결과적으로 약 2만배의 성능 개선을 달성했다.

* + 1. **장단점 및 향후 개선 방안**

**장점**

뛰어난 시간 효율성이 가장 큰 장점이다. K-mer 해싱 덕분에 후보 위치를 찾는 시간복잡도가 O(n)에서 O(c)로 대폭 줄어들었다. 실제로는 평균 5-6개의 후보 위치만 검사하면 되므로 계산량이 매우 적다.

또한 효과적인 조기 필터링도 큰 강점이다. 비둘기집 원리를 통해 불가능한 후보의 99.99%를 미리 걸러내므로, 전체 비교 연산을 수행하기 전에 대부분의 불필요한 계산을 방지할 수 있다.

**단점**

메모리 사용량 증가가 주된 단점이다. 해시 테이블 크기가 참조 서열 길이에 비례해서 커지기 때문에, 8bp k-mer 기준으로 약 8배의 추가 메모리가 필요하다.

특정 상황에서의 성능 저하도 문제가 될 수 있다. 반복적인 서열에서는 해시 충돌이 자주 발생하고, 동일한 k-mer가 여러 위치에 나타나면 후보 위치 수가 급격히 늘어난다. 또한 매우 짧거나 긴 서열에서는 k-mer 크기를 최적화하기 어렵다는 점도 있다.

**향후 개선 방안**

k-mer 크기의 동적 조정

현재는 k=8로 고정되어 있지만, 입력 데이터의 특성에 따라 유연하게 조정할 수 있다면 더 좋을 것이라고 분석하였다. 서열의 복잡도가 높을 때(반복 구간이 적을 때)는 k를 늘려서 해시 충돌을 줄이고, 복잡도가 낮을 때는 k를 줄여서 후보 위치를 늘리는 전략을 생각해볼 수 있다.

병렬 처리를 위한 구조 개선

전체 해시 테이블을 여러 개의 독립적인 서브 테이블로 나누어 각각을 다른 스레드에서 처리하는 방법도 효과적일 방법일 것이라 분석하였다. 예를 들어 k-mer의 첫 2bp를 기준으로 AA, AC, AG, AT, CA 등 16개의 서브 테이블로 분할하면, 각 스레드가 메모리 경합 없이 독립적으로 후보 검색을 수행할 수 있다. 이렇게 하면 병렬 처리 효율성을 극대화할 수 있을 것이다.

* 1. **박지우**
     1. **구현 알고리즘 구조**

구현한 알고리즘은 부모의 전체 유전자 서열과 아이의 short reads를 기반으로 아이의 유전자 배열을 추정한 이후, 이를 부모의 유전자와 비교하여 일치율을 계산하고 친자 여부를 판별한다. 이번 장에서는 알고리즘 작동 방식을 단계별로 상세히 설명한다.

먼저 알고리즘에 사용할 데이터를 생성한다. 해당 데이터는 부모 유전자와 아이 short reads로 구성된다. 부모 유전자는 약 300만 개의 염기로 이루어지며, A, G, C, T 중 하나를 무작위로 선택하여 이를 300만 번 반복해 배열한 후 텍스트 파일로 저장한다. 이후 아이 수와 short read 수를 입력받으면, 입력된 수만큼 부모 유전자의 임의 위치에서 32개 염기를 추출하여 short read를 생성한다. 생성된 각 short read는 0.7에서 1.0 사이의 무작위 일치율을 기준으로 일부 염기가 무작위로 치환된다. 이렇게 변형된 short read들은 하나의 아이에 대한 데이터로 구성된다. 이 과정을 아이 수만큼 반복하여 각 아이에 대한 데이터를 생성하고, 각 아이의 식별 번호, short reads, 일치율에 대한 정보를 포함한 텍스트 파일을 생성하여 저장한다.

알고리즘의 메인 함수는 check\_paternity\_anchor\_based() 이다. 메인 함수는 먼저 부모와 아이의 텍스트 파일을 불러들여 읽는 것으로 시작한다. 그 다음, 아이 1명씩 복원 및 비교 과정을 반복한다.

해당 과정은 먼저 get\_words\_and\_neighborhoods()을 사용하여 32자의 short read를 8자 단위로 4개 word로 분할한 이후, 각 word에 대해 1개의 mismatch를 허용한 모든 neighborhood를 생성한다. 여기서 word란 short read를 일정 길이의 연속된 서열로 분할한 배열을 의미하며, neighborhood는 각 word에 대해 1개의 mismatch를 허용하여 생성 가능한 모든 변형 서열 집합이다. 예를 들어 word가 ACGT라면 neighborhood는 ACAT, AGGT, ACGA 등이 될 수 있다.

그 다음 find\_word\_positions()를 호출하여 부모 유전자에서 해당 word와 neighborhood가 등장하는 위치를 라빈-카프 해시 방법을 통해 탐색한다. 탐색된 위치를 기반으로 short read가 정렬될 수 있는 anchor 후보 위치를 탐색한다. 여기서 anchor란, short read가 부모 유전자 서열 상에 정렬될 수 있는 후보 시작 위치를 의미한다. anchor 위치는 복원된 배열의 기준점이 되어 정렬 정확도에 큰 영향을 미친다. anchor 후보 위치는 부모 유전자와 비교했을 때, mismatch 수가 3개 이내인 조건을 만족해야한다. 이 방식을 각 short read마다 적용하여 여러 anchor 후보를 수집한다. 수집한 anchor 후보들은 anchor\_counts = Counter(all\_candidate\_anchor\_positions), best\_anchor\_start = anchor\_counts.most\_common(1)[0][0] 코드를 통해 가장 빈번하게 등장한 위치를 선택하여 가장 가능성 높은 유전자 위치를 찾는다.

선택된 anchor 최종 후보들은 reconstruct\_best() 함수에 전달한다. 해당 함수에서는 한 개 또는 다수의 최종 anchor 후보들을 각각 정렬 및 복원한다. 선택된 anchor의 위치를 중심으로 부모의 유전자에서 양쪽으로 일정 범위를 포함한 추정 부모 shortread를 생성한다. 이때 복원 범위는 아이의 short reads가 모두 연속된 위치에 정렬된다고 가정할 때의 최대 길이, 즉 short read 수 × 32로 설정된다. 이때 조각들의 위치 정보는 사전에 계산된 해시 테이블 또는 위치 매핑을 통해 추출된다. 다음엔 align\_and\_reconstruct() 을 호출하여 수집된 부모 short read 조각들을 anchor 기준으로 정렬한다. 각 아이 short read 는 부모 short read와 비교되며, mismatch 수가 3개 이내인 조건을 만족해야 유효한 정렬로 간주된다. 정렬된 short read 조각들의 각 염기 위치마다 등장한 염기를 집계하여, 가장 빈도수가 높은 염기를 선택함으로써 복원 배열을 복원한다. 해당 방식을 예를 들어 설명하자면, 특정 위치에서 ACCT, ACCG, ACTT, ACCT가 매칭되었었다면 각 위치에서 가장 많이 등장했었던 것 선택하여 복원된 배열은 ACCT가 된다. 이렇게 복원된 배열은 부모 short read와 비교하여 일치율을 평가한다. 예측 일치율은 (일치한 염기 수) / (전체 비교된 염기 수) 로 계산한다. 예를 들어, 복원 배열이 ‘ACCTGA’이고, ‘부모 short read 중 비교할 부분이 ‘ACGTGA’이면, 일치한 염기 수가 5이고 전체 비교된 염기 수가 6이므로 예측 일치율은 약 0.833이 된다. 예측 일치율은 친자 판별에 사용하며, 설정된 친자 판별의 기준은 80%이므로 예측 일치율이 80% 이상이면 친자로 판별한다. is\_paternity = identity >= (1 - mismatch\_threshold)라는 조건식을 통해 친자 여부를 판별하며, 여기서 mismatch\_threshold는 0.2로 설정되어 있다. mismatch\_threshold은 판별 기준인 0.2이다. 복수의 anchor 후보로 복원이 시도된 경우, 각 복원 결과의 일치율이 다를 수 있다. 이때 친자인 경우를 최대한 보존하기 위해, 계산된 일치율 중 가장 높은 값을 선택하여 최종 친자 여부를 판별한다.

이후 시작했을 때 start\_time = time.time() 로 기록했던 시작 시간과 end\_time = time.time()로 기록한 종료 시간을 elapsed\_time = end\_time - start\_time로 알고리즘의 성능을 평가하기 위해 실행 시간을 함께 출력한다.

* + 1. **컴퓨터 사양에 따른 결과 분석**

알고리즘을 돌린 환경은 구글 코랩 (Google colab) 이다. 코랩 서버 스펙은 Intel ® Xeon ® CPU @ 2.30GHz (Dual-Core) 인 CPU, Nvidia Tesla T4 인 메모리 8GB의 GPU로 구성되어 있으며, 단정도 연산 기준 5.5 TeraFLOPS의 성능을 가진다.

Colab 환경에서 알고리즘을 실행한 뒤, 연산 시간과 정답률을 아이 수와 short read 수에 따라 측정하였다. 연산 시간은 알고리즘을 구현한 코드가 데이터를 읽고 일치율과 친자 판별 결과를 출력하기까지의 시간이고, 정답률은 예측된 친자 여부가 실제 친자 여부와 일치한 횟수를 총 아이 수로 나눈 값이며, 이를 통해 알고리즘의 정확도를 평가할 수 있다. 연산시간과 정답률의 현실성을 위해 5번 정도 같은 연산을 수행하여 나온 정답률을 참고하였다. 하지만, 데이터의 크기가 매우 큰 D 조건의 경우에는 실험 시간 부족으로 인해 반복적인 관측은 하지 않았다. 실험 조건의 일관성을 위해 short read의 길이는 32자로, 부모 유전자의 길이는 300만자의 염기로 고정하였다.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

아이의 수와 short reads 수가 모두 10인 A 경우, 연산시간은 약 11분이었고 정답률은 80~90% 내외였다. 아이 수가 50명이고 short read 수가 10개인 B 경우 연산 시간은 약 59분이었으며, 정답률은 80~90%를 유지했다. 반대로, 아이 수가 10명이고 short read 수가 50개인 C 경우에도 연산 시간은 약 56분으로 유사한 연산 시간과 정답률을 보였다. 아이 수와 short read 수가 모두 50개인 D 경우, 약 30명의 데이터를 처리하는 데 1시간 30분이 소요되었으므로, 전체 연산 시간은 약 2시간 30분에서 3시간 사이로 추정된다. 또한 해당 연산의 결과를 통해 정답률을 계산하면 정답률은 마찬가지로 85%로 나왔다. 따라서 모든 연산을 마쳤을 때 정답률 또한 80~90%로 추정된다.

Trivial mapping 방식을 적용할 경우 예측 정확도는 향상되었으나, 연산 시간이 크게 증가하였다. 또한 mismatch 허용 범위를 확장하여 anchor 후보를 늘리는 방식 역시 메모리 사용량과 연산 비용이 급증하여 본 실험에서는 생략하였다. 따라서 일치율이 0.8 이하인 경우 예측 실패가 많았으며, 본 실험에서는 일치율이 0.8 이하일 경우 비친자로 간주하도록 설정하였다.

두 번째 실험에서는 부모 유전자 길이를 300만 염기에서 30만 염기로 감소시켜 연산 시간 및 정답률의 변화를 관찰하였다. 나머지 조건(A, B, C, D)은 동일하게 유지하였다.

텍스트, 스크린샷, 도표, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

부모의 유전자 수를 처음의 10분의 1로 줄인 결과, 연산 시간은 대체적으로 매우 줄어드는 모습을 보였다. 아이의 수가 10명이고 short reads의 수가 10개인 A 경우, 연산시간은 1~2분 정도로 줄었으며, 정답률이 80~90%정도로 이전 실험과 비슷한 결과를 냈다. B와 C의 경우에도 연산 시간은 5~6분 정도로 감소하였다. 하지만, 이번 실험에서는 아이 수가 10명인 C의 경우에서는 정답률이 80~90%로 높은 수치를 보여주었지만, 아이 수가 50명인 B의 경우에서는 정답률이 40~70% 정도로 다소 낮아졌다. 또한 D의 경우에서도 정답률이 50% 정도의 낮은 수치를 보여주었다.

해당 실험에서 나온 결과로 부모 유전자 길이를 감소한 것이 정답률에 영향을 줄 수 있다는 추측을 할 수 있다. 특히, 아이 수가 많아질수록 정답률이 감소하는 경향이 뚜렷하게 나타났으며, 이는 후보 anchor 수의 감소와 관련이 있을 가능성이 있다. 즉, 부모 유전자 길이를 줄이면 anchor 탐색 범위가 줄고, 정보량 감소로 인해 정답률이 낮아지는 경향을 실험에서 관측할 수 있었다.

* + 1. **성능 분석**

알고리즘의 시간 복잡도는 유전자 데이터 생성 과정까지 포함할 경우, 전체적으로 O(N × M × L + N × M × W)로 추정된다. 여기서 N은 아이의 수, M은 각 아이가 가진 short read 수, L은 부모 유전자 길이, W는 short read 탐색 시 부모 유전자에서 비교하는 범위의 절반이다. 이 복잡도는 라빈-카프 해시 기반 탐색과 Trivial mapping 방식을 결합하여 구현된 구조에서 비롯된다. 각 short read에 대해 후보 anchor 위치를 탐색하고, 해당 anchor를 기준으로 복원할 배열을 생성하고 비교하는 데 많은 연산이 소요된다.

또한 알고리즘은 리스트와 딕셔너리를 활용하여 데이터를 저장하고 처리하므로, 메모리를 지속적으로 사용하게 된다. 따라서 제자리(in-place) 알고리즘은 아니며, 추가적인 메모리 공간을 필요로 한다.

그리고 복원 과정에서는 anchor 탐색 실패 시 복원이 이루어지지 않는다. 이러한 실패는 주로 일치율이 0.8 이하일 때 자주 발생했으며, 여러 번의 실험을 통해 이를 확인할 수 있었다. 따라서 anchor 탐색에 실패하면 해당 short read는 일치율이 0.8 이하인 것으로 간주하고, 비친자로 판별한다. 이러한 예외 처리를 통해 anchor 탐색 실패나 복원 불가 상황에서도 알고리즘의 전체 흐름이 중단되지 않도록 구성하였다. 복원 중단 시 빈값을 그대로 두지 않고, 결과로 이어지도록 처리함으로써 알고리즘의 안정성을 높였다. 이로 인해 모든 아이에 대해 친자 판별 결과를 안정적으로 얻을 수 있었다.

텍스트, 번호, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

* + 1. **장단점 및 향후 개선 방안**

알고리즘은 트리비얼 매핑(trivial mapping) 방식과 라빈-카프(Rabin-Karp) 해싱 알고리즘을 결합하여, short read 기반 친자 판별을 수행하였다. 탐색 정확도와 연산 속도 간의 균형을 목표로 구현되었으며, 복원 및 판별 전반에서 몇 가지 두드러진 장점과 한계가 나타났다. 먼저, 라빈-카프 해싱을 활용한 슬라이딩 윈도우 기반 탐색을 통해, short read 검색 속도가 빠르며, 트리비얼 매핑 방식을 통해 구현 복잡도는 낮고 구조는 단순하다. anchor 기준으로 다수의 read들이 겹쳐지는 구간을 중심으로 정렬한 뒤, 동일 위치에서 겹치는 염기들은 하나로 통일하여 중복 제거 및 간결한 복원을 실현하여 정확하고 짧은 서열을 복원할 수 있었다. 또한 자식 수, short read 수, mismatch 허용치 등을 사용자 정의로 설정할 수 있어 유연한 실험 환경을 제공하였다. 그리고 mismatch 허용치 및 데이터 설정을 다양화하더라도, 전체적인 친자 판별 정답률은 80~90% 수준으로 비교적 안정적인 성능을 나타냈다.

하지만 전체적인 정확도는 약 80% 수준에서 정체되었으며, 일부 테스트에서는 일관된 오답이 발생하였다. 이는 복원된 서열이 실제 부모 유전자와 다소 다르게 재구성되는 경우 때문이다. 그리고 복수의 anchor로부터 여러 일치율이 도출되는 경우, 가장 높은 값을 기준으로 친자 여부를 판단하는 것은 판별 민감도를 높이지만, 일부 비친자의 경우에서도 높은 일치율이 나올 수 있어 오판 확률이 소폭 증가할 수 있다. 또한 확률적으로 부정확한 위치에 anchor가 선택될 경우, 이후 정렬 및 복원 단계에서 정렬이 실패하고, 일치율이 실제보다 과도하게 낮게 예측되는 오류가 발생하였다. 일치율이 0.8 이하로 낮아지면, 다수결 기반 복원이 불가능하거나 무의미해지므로, anchor 기반 복원이 실패하는 임계점이 존재한다. 따라서 아이가 비친자일 경우, 해당 아이의 유전자 배열을 복원하는데 실패할 확률이 크다. 또한 복원된 배열이 짧거나 일부 서열이 누락되거나 변형된 경우, 계산된 일치율이 실제와 다르게 왜곡될 가능성이 있다. 이러한 문제를 보완하기 위해, anchor 기반 복원이 실패할 경우에는 기존의 trivial mapping 방식으로 대체 복원을 시도함으로써 정확도를 향상시키는 방향으로 개선이 가능하다. 또한, neighborhood 조건을 확장하여 허용되는 mismatch 수를 기존의 1개에서 2개로 증가시키는 방식도 복원 성공률을 높이는 대안이 될 수 있다. 이 경우, 비교해야 할 후보 서열의 수가 기하급수적으로 증가하게 되어 저장해야 할 데이터량과 탐색 횟수가 늘어나지만, 복원 가능성이 높아지고 정확한 anchor를 찾을 확률 또한 향상될 수 있다. 다만, 허용된 mismatch 수가 많아질수록 오류탐지 가능성 또한 함께 증가하므로, 이 방식은 탐색 정밀도와 계산 자원 간의 균형을 고려하여 적용해야 한다.

* 1. **정태호**
     1. **구현 알고리즘 구조**

본 프로젝트에서 구현한 알고리즘은 "이진 탐색 기반 시드 매칭 알고리즘 (Binary Search-Based Seed Matching Algorithm)"으로, 부모 DNA 시퀀스에서 추출한 k-mer 인덱스를 정렬하고 ,각 read의 일부를 시드로 사용하여 이진 탐색을 수행함으로써 자식 DNA(reads)를 빠르게 매핑하는 방식이다. 기존 벤치마크 알고리즘인 트리비얼 매핑(trivial mapping)은 부모 DNA 전체에 대해 전수 비교를 수행하는 비효율적인 구조였으나, 본 알고리즘은 사전 필터링(pre-filtering), 조기 중단(early abort), 보완 탐색(local fallback)이라는 세 가지 핵심 전략을 조합함으로써 효율성과 정확도를 동시에 확보하고자 하였다.

구현 과정은 다음과 같은 구조를 가진다.

1. **k-mer 인덱스 생성**  
   부모 DNA는 길이 N의 염기서열로 주어지며, 이로부터 길이 k인 모든 연속된 부분 문자열(k-mer)을 추출하고, 각 k-mer에 해당하는 위치 정보를 구조체 배열에 저장되며 그 후 정렬한다. 이 배열을 사전순으로 정렬하여 전체 시퀀스를 빠르게 탐색할 수 있는 사전 인덱싱을 만들고, 이 인덱스를 통해 특정 시드를 이진 탐색으로 빠르게 검색할 수 있게 된다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. **시드 기반 사전 필터링 (multi-seed prefiltering)**  
   각 read는 길이 L의 문자열이며, 각 read에서 시작, 중간, 끝 세 위치에서 시드를 선택한다. 이는 read 내 특정 위치에만 변이가 존재해도 전체 read가 실패하지 않도록 하기 때문이다. 선택한 시드들을 통해 k-mer 인덱스를 이진 탐색하여 후보 위치를 좁히며 찾을 수 있다. 다시 말해서 다중 시드를 사용함으로써, 하나의 시드가 변이에 의해 손상되었더라도 나머지 시드를 통해 복원 가능한 후보 위치를 찾을 수 있는 방법이다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. **mismatch 계산 및 조기 중단 (early abort)**  
   후보 위치가 정해지면, 해당 부모 DNA 구간과 read를 비교하며 mismatch 개수를 계산한다. mismatch 수가 허용된 오차 D를 넘는 경우 비교를 즉시 중단(early abort)한다. 이로 인해 불필요한 비교 연산을 줄일 수 있으며, 대부분의 경우 빠르게 실패 여부를 판단할 수 있다. 기준 이내이면 read를 해당 위치에 child DNA에 삽입하여 복원한다.

텍스트, 스크린샷이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. **fallback 탐색 (local fallback)**  
   세 시드를 통해 후보 위치를 찾지 못했거나 모든 후보가 mismatch 조건을 만족하지 못한 경우, 시드 기준 위치를 중심으로 fallback 탐색이 실행된다. 이때 가장 유력한 seed 기준 위치를 중심으로 ±64bp 범위를 전체 탐색한다. 이 fallback 탐색 역시 mismatch 수가 D를 초과하면 비교를 중단한다.

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

(불필요한 반복을 줄이고, 코드 흐름을 간단하게 처리하기 위해 goto사용)

본 알고리즘은 시드 기반의 효율적인 후보 필터링 구조와 early abort 기법, local fallback 탐색 전략이 유기적으로 결합된 구조로 구성되어 있다.

* + 1. **컴퓨터 사양에 따른 결과 분석**

실험은 다음과 같은 노트북 환경에서 수행되었다.

* **CPU:** Intel Core i7-12700H (12세대)
* **RAM:** 16GB DDR4
* **OS:** Windows 11 64-bit

본 프로젝트는 위와 같은 사양의 노트북 환경에서 실행되었고 대량의 read(M), 긴 부모 DNA 시퀀스(N), 다양한 파라미터 설정 등이 연산량을 급격히 증가시키므로 메모리 및 연산 시간의 병목이 발생했다.

특히 초기 실험과정에서 다음과 같은 한계가 발견되었다.

* N이 30만 이상, M이 5만 이상일 경우, malloc()에 의한 메모리 할당 실패, 프로그램 종료 현상 발생
* fallback 탐색이 자주 발생할 경우 시간 지연이 발생하며, O(MNL) 수준의 최악 복잡도가 드러나기도 함

이에 따라 본 실험에서는 다음과 같은 파라미터 범위를 설정하여 결과를 측정하였다:

* N (부모 DNA 길이): 100,000 ~ 200,000
* M (read 개수): 10,000 ~ 20,000
* k (seed 길이): 5 ~ 9
* D (허용 mismatch 수): 3 ~ 9
* read 길이 L = 32로 고정

특히 메모리 최적화를 위해 read 자체는 복사하지 않고 포인터로 관리하였고, child DNA는 처음에 'N'으로 초기화 후 매핑된 read만을 삽입하는 구조로 구현하였다. 이는 메모리 사용량을 절감하고, 실험 범위 내에서 안정적인 실행을 가능하게 했다.

또한, 콘솔 로그를 통해 매핑 진행률(mapping progress)을 10% 단위로 출력하며 실시간 확인이 가능하도록 하여, 대규모 입력 실험의 진행 상태를 파악할 수 있게 하였다.

* + 1. **성능 분석**

성능 평가는 세 가지 측면에서 진행되었다.

* **실행 시간** (총 mapping 시간)
* **복원율 (Coverage)**: 전체 N 중 child DNA에서 N이 아닌 문자의 비율
* **정확도 (Identity)**: 부모/자식 DNA와 복원된 DNA 간 일치율 비교

1. **read 수 M 증가에 따른 영향**  
   read의 수가 많아질수록 복원율은 당연히 증가했으며, fallback 발생률도 낮게 유지되면서 실행 시간은 M에 대해 선형적으로만 증가했다. 이는 탐색 효율성을 유지하기 위한 다중 시드 기반 필터링 구조가 대부분의 read를 빠르게 처리했기 때문이다.

텍스트, 스크린샷, 라인, 도표이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. **부모 DNA 길이 N 증가에 따른 영향**  
   부모 DNA가 길어질수록 k-mer 인덱스가 커지고, fallback 탐색 범위도 넓어지기 때문에 실행 시간은 증가했다. 탐색 시간은 log N 수준으로 증가하는 반면, read 수가 고정된 상태에서 시퀀스가 길어져 coverage가 낮아지면서 복원율은 감소했다.

텍스트, 라인, 스크린샷, 도표이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. **seed 길이(k) 변화에 따른 영향**  
   k가 작을 경우 시드 충돌이 많아 후보 위치가 많아지고, 탐색 시간이 증가했다. 이 경우 정확도는 좋지만 속도는 저하되었다. 반대로 k가 클 경우 후보 위치 자체를 찾지 못하여 시드 일치 확률이 낮아 대부분 fallback으로 넘어가거나 바로 실패 처리되어 실행 시간은 급격히 감소하면서 복원율도 하락했다.

텍스트, 스크린샷, 디스플레이, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. **허용 mismatch 수(D) 변화 실험:**  
   D가 클수록 더 많은 오차를 허용하므로 원칙적으로 비교량이 늘어야 하지만, early abort 전략으로 인해 실행 시간은 오히려 감소하여 개선되는 경향을 보였다. 다만 D가 너무 크면 잘못된 매핑이 늘어 정확도는 낮아질 수 있어, 적절한 D의 선택이 중요했다.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

* + 1. **장단점 및 향후 개선 방안**

**장점**

* **효율적인 후보 필터링:** 시드를 이진 탐색으로 빠르게 찾아 전체 전수 탐색 없이 매핑 가능
* **평균적인 성능 안정성:** read 수 M이나 부모 DNA 길이 N이 커져도 커져도 fallback 없이 대부분 탐색이 log N 복잡도 내에서 해결되어 평균 성능이 안정적으로 유지
* **오차 허용 기반 복원 가능:** mismatch를 일정 수준 허용하여 일부 손상된 read도 복원할 수 있어 실제 데이터에 강함
* **유연한 확장 가능성:** read 단위 병렬 처리나 seed 캐싱과 같은 구조적 확장이 쉬운 구조

**단점**

* **메모리 사용량:** k-mer 인덱스와 child DNA를 별도로 관리해야 하므로, 메모리 점유가 트리비얼 방식보다 높음
* **결과의 비일관성:** read의 순서나 중복 여부에 따라 복원 결과가 달라질 수 있어 실행 시마다 결과가 달라질 가능성 존재
* **fallback 의존성:** seed 매칭 실패 시 fallback에 의존하게 되어, 구조적으로 일부 read에 대해 속도 저하가 불가피

**데이터 출처 및 생성 방법**

**부모 DNA (parent sequence)**

* 염기 A, C, G, T 중 하나를 무작위 선택하여 길이 N만큼 생성
* random\_base() 함수를 통해 생성된 시퀀스를 parent.txt에 저장

**자식 DNA (child sequence)**

* 부모 DNA를 그대로 복사한 후, mutation rate(3~7%)를 기반으로 무작위 위치에서 다른 염기로 치환
* random\_diff() 함수로 염기 변형 후 child.txt에 저장

**read 생성 (short reads)**

* child DNA에서 길이 L만큼 연속된 구간을 무작위로 추출 (중복 허용)
* 총 M개의 read를 생성하며, reads.txt에 저장

**실험 실행**

* k-mer 인덱스 생성 → 매핑 실행(map\_reads) → identity/coverage 계산 → 콘솔 및 파일 출력
* 전체 과정은 하나의 main.c로 구현되어 있으며, 모든 핵심 함수는 직접 구현되었고 외부 라이브러리는 사용되지 않았음

이와 같은 구조는 실제 친자 검사를 포함한 유전체 분석의 핵심 로직을 간단한 방식으로 생성한 것으로, 변이 위치는 랜덤이지만 전체 mutation rate를 일정 수준으로 제어하여 실험 간 일관성을 유지했고 read 복원 정확도 향상과 빠른 매핑을 목표를 위해 설계했다.

**참고 그림**

**부모 DNA Sequence**

AGCTTAGTGATCTTTAGCCCTAGTCACCTAGTCTCCATCATGCTACGGCTAAGCATATTATCATGATACAGGTATTC….. …………CATG

**길이 : N = 3,000,000**

**아이 한 명의 DNA sequence로부터**

**랜덤 추출된 M=10,000개의 short reads**

**1st short read**

8개 | 8개 | 8개 | 8게

8개 | 8개 | 8개 | 8게

**2nd short read**

8개 | 8개 | 8개 | 8게

**3rd short read**

**.**

**.**

**.**

**M=10,000th short read**

8개 | 8개 | 8개 | 8게

**허용 mismatch 수 D <= 3개**

**( short read를 4파트로 나눴을 때 4파트 중 한 파트는 완전히 일치하는 파트)**