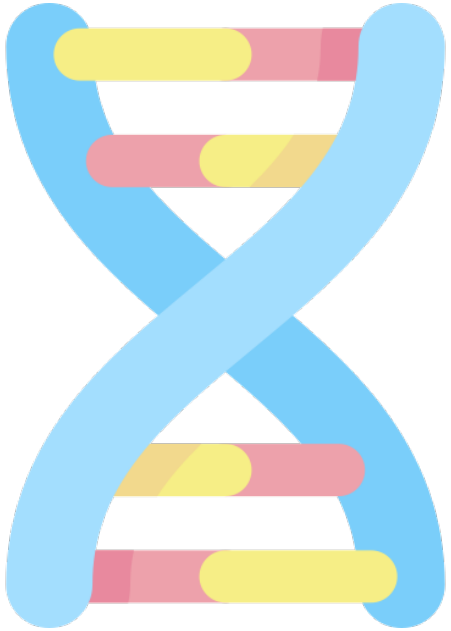
DNA복원
feat. Needleman-Wunsch algorithm

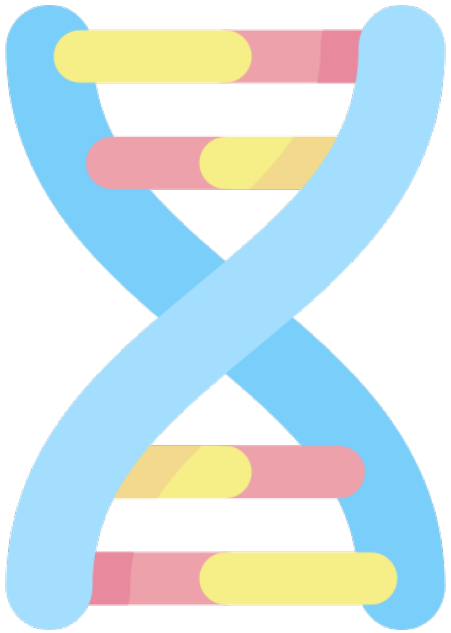
컴퓨터공학과 2019112007 권예진



문제 정의

reference 길이 MAX=100000
short read 길이 K=100
short read 개수 N=1500
reference DNA와 My DNA의 차이 50%

Needleman–Wunsch 알고리즘을
mapping시 사용하여 DNA를 복원하였다.



reference DNA 데이터 생성방식

```
random_device rn;
mt19937_64 rand(rn());
uniform_int_distribution<int> range(0, 3);
//srand(6783);
int cnt = MAX;
char* r_DNA = new char[MAX + 1];

char num_array[5] = "ACGT";

for (int i = 0; i < MAX; i++) {
    int random = range(rand);
    r_DNA[i] = num_array[random];
}
r_DNA[MAX] = 'WO';

string str(r_DNA);
r = str;
ofstream fout("Reference_DNA.txt");
fout << r_DNA;
fout.close();
 free(r_DNA);
```

메르센 트위스터 난수 생성기,
uniform_int_distribution 클래스 이용
0-3구간의 난수 생성

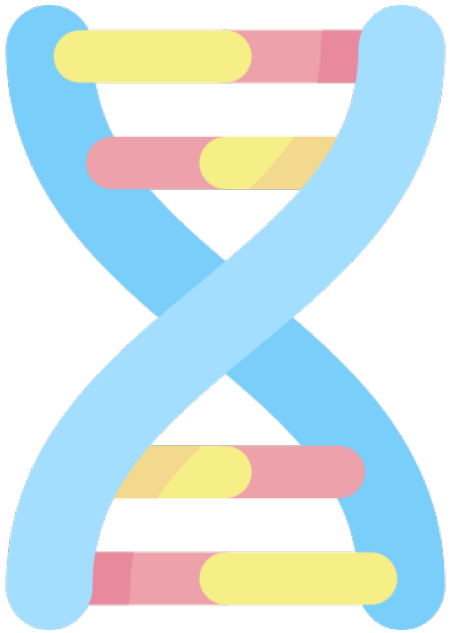
→

0: A

1: C

2: G

3: T



My DNA 데이터 생성방식

```
random_device rn;
mt19937_64 rand(rn());
uniform_int_distribution<int> range(0, 3);
char* my_DNA = new char[MAX + 1];
int diff = 2;
for (int i = 0; i < MAX; i++) {
    my_DNA[i] = r[i];
    if ((i != 0) && ((i + 1) % diff == 0))
    {
        while (my_DNA[i] == r[i]) {
            int dif = range(rand);
            switch (dif) {
            case 0:
                my_DNA[i] = 'A';
                break;
            case 1:
                my_DNA[i] = 'T';
                break;
            case 2:
                my_DNA[i] = 'G';
                break;
            case 3:
                my_DNA[i] = 'C';
                break;

            }
        }

    }
```

메르센 트위스터 난수 생성기,
uniform_int_distribution 클래스 이용
0-3구간의 난수 생성

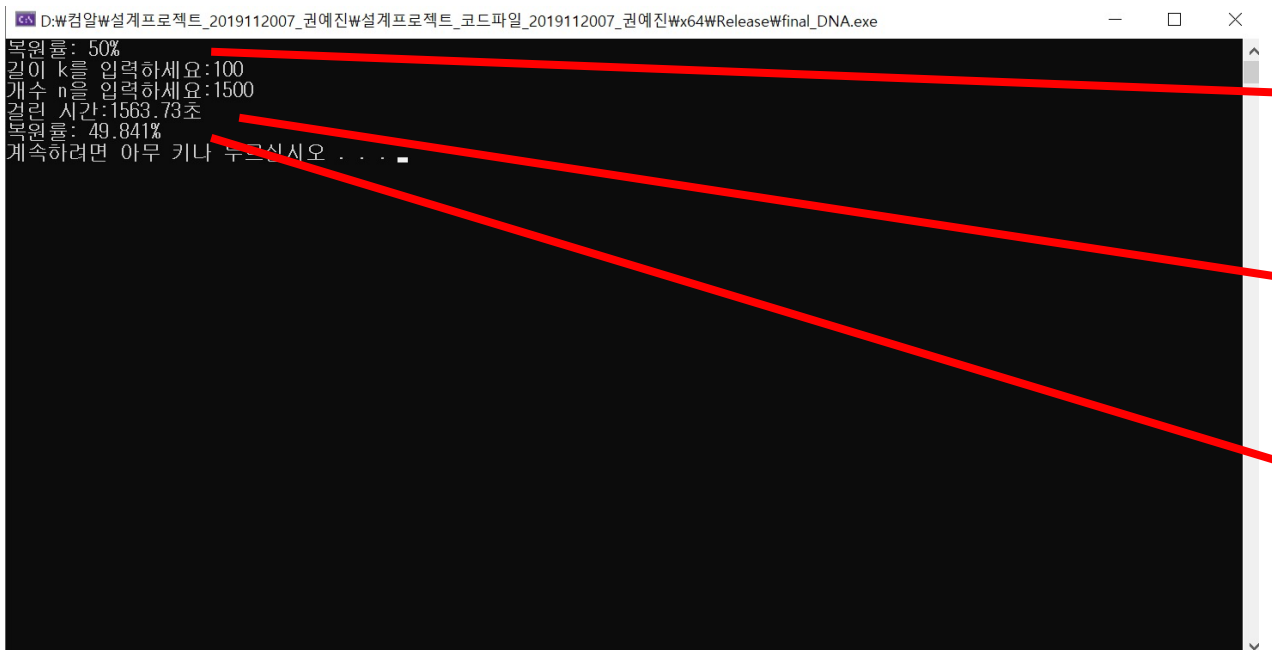
→50%차이

(reference DNA 인덱스숫자+1)

==2의 배수

현재 reference DNA 인덱스의 문자와 다른 값을 가질 때까지 값을 교체

Input and Ouput



초기 설정값
Reference DNA와 MY DNA의 일치율 50%

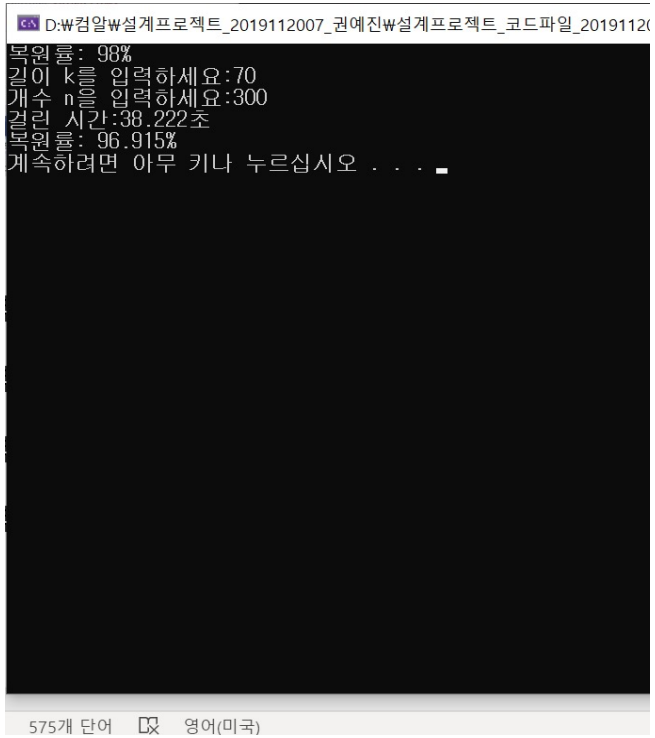
소요 시간
1563.73초

복원 완료 한
restruct my DNA와 MY DNA의 일치율
49.841%

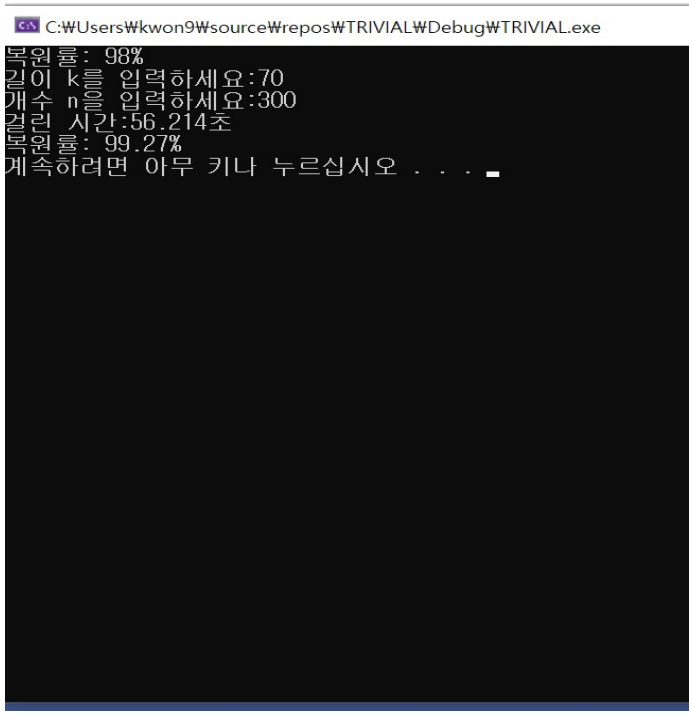
Benchmark Trivial

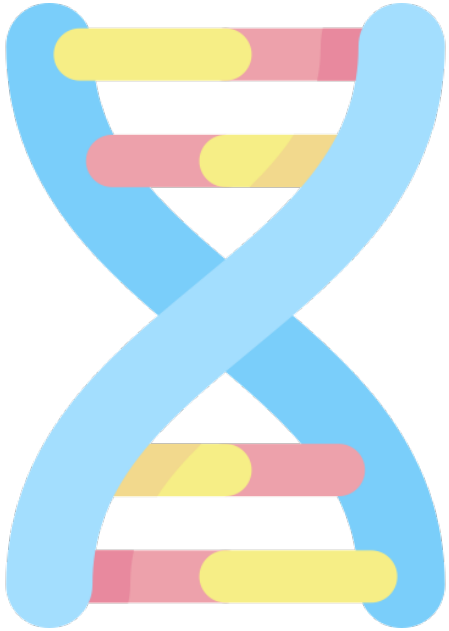
조건: Reference DNA와 MY DNA의 차이 2%
short read개수 300, 길이 70 reference 길이 20000

Needleman-Wunsch



Trivial



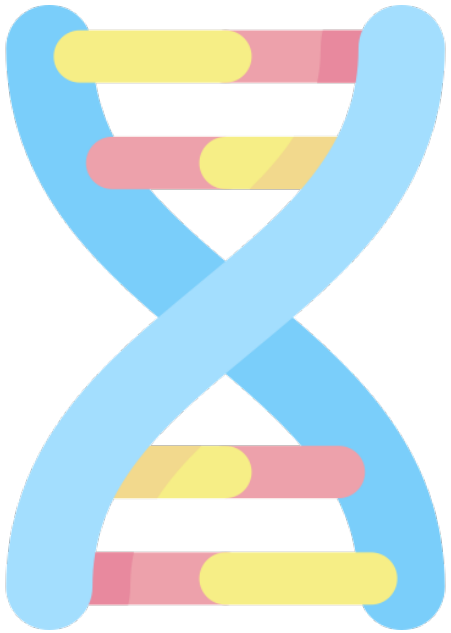


My algorithm

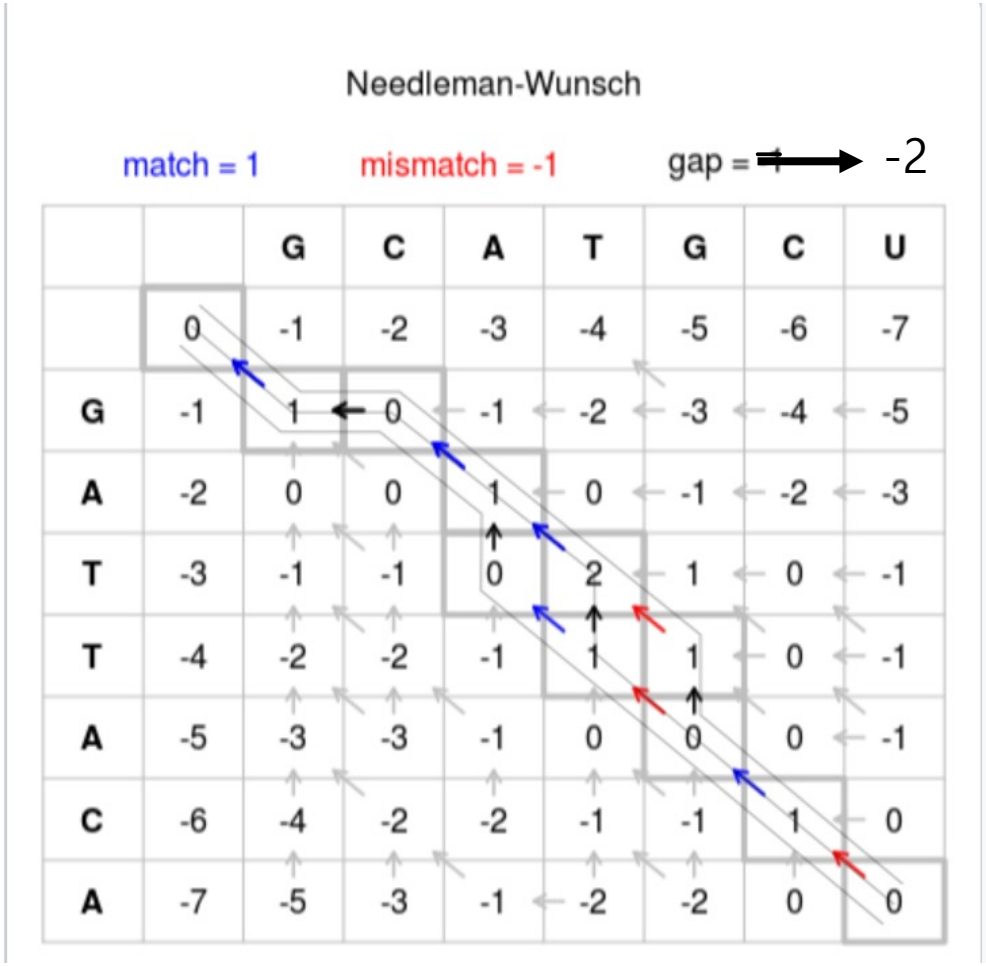
Needleman-wunsch의 Match matrix 생성

→재구성된 short read sequence에서
Reference와 가장 많은 문자가
일치하는 구간을 찾아 시작 인덱스를 저장

→해당 인덱스부터 길이 k까지
reference의 값을 교체

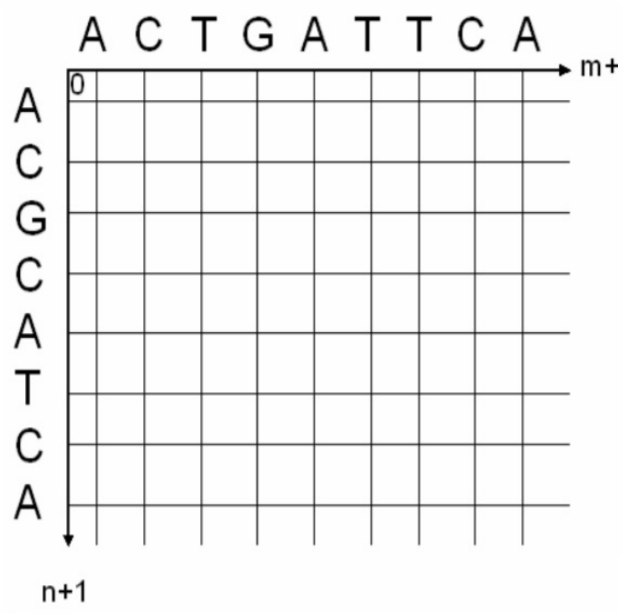


My algorithm



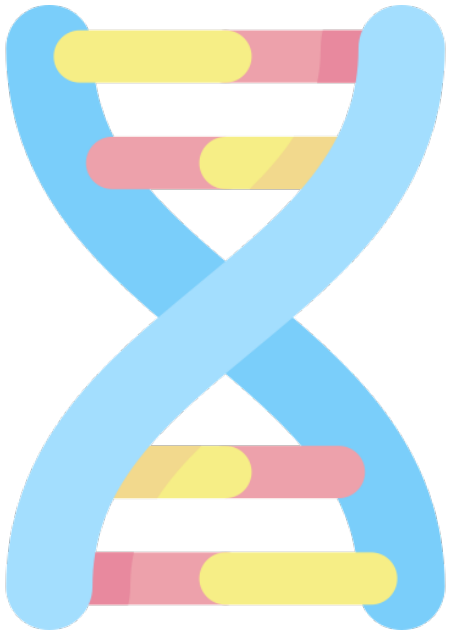
		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

My algorithm



		A	C	T	G	A	T	T	C	A
	0	-2	-4	-6	-8	-10	-12	-14	-16	-18
A	-2	2	0	-2	-4	-6	-8	-10	-12	-14
C	-4	0	4	2	0	-2	-4	-6	-8	-10
G	-6	-2	2	1	4	2	0	-2	-4	-6
C	-8	-4	0	-1	2	1	-1	-3	0	-2
A	-10	-6	-2	-3	0	4	2	0	-2	2
T	-12	-8	-4	0	-2	2	6	4	2	0
C	-14	-10	-6	-2	-4	0	4	2	6	4
A	-16	-12	-8	-4	-5	-2	2	1	4	8

		A	C	T	G	A	T	T	C	A
	0	-2	-4	-6	-8	-10	-12	-14	-16	-18
A	-2	2	0	-2	-4	-6	-8	-10	-12	-14
C	-4	0	4	2	0	-2	-4	-6	-8	-10
G	-6	-2	2	1	4	2	0	-2	-4	-6
C	-8	-4	0	-1	2	1	-1	-3	0	-2
A	-10	-6	-2	-3	0	4	2	0	-2	2
T	-12	-8	-4	0	-2	2	6	4	2	0
C	-14	-10	-6	-2	-4	0	4	2	6	4
A	-16	-12	-8	-4	-5	-2	2	1	4	8



result

My machine information

PC

프로세서:	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
설치된 메모리(RAM):	8.00GB

Time and space complexity

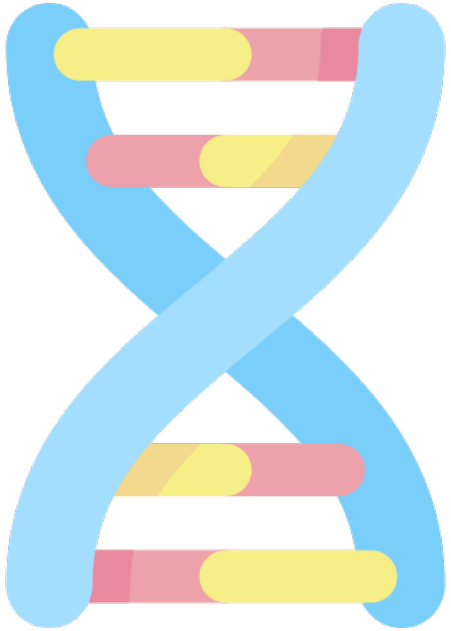
Time→O(mn)

Space→O(mn)

m: reference길이, n: short read 길이

Compare with the benchmark

Trivial→O(n^2)



Future work

개선 해야 할 사항

Short read의 길이와 reference의 길이가 긴 상황에서 복원 시간을 줄일 수 있도록 발전

개선 아이디어

Match matrix 생성시
이차원 벡터가 아닌 다른 자료구조를 고안