문제

주어진 두 개의 염기 서열에서 local alignment를 실행하여 score가 가장 높은 다섯 부분을 출력하는 문제이다. sequence alignment시 scoring하는 방법은 다음과 같다.

* matched : + 3
* mismatched : -2
* first gap penalty : -5
* continuous gap penalty : -2

만약 mismatched나 gap이 5회 연속 발생한다면 부분서열이 끝났다고 본다.

풀이

- 주어진 각각의 data들의 염기 서열의 길이가 약 300,0000정도임으로 만약 일반적인 방법으로 300,0000*300,0000 이차원 배열을 만든다면 적어도 수십 테라바이트의 메모리가 필요하게 된다. 메모리를 많이 쓰지 않으면서 데이터를 처리하기 위하여 다음과 같은 방법을 사용했다.

이차원 배열을 만들어 모든 score를 저장해두는 이유는 이후 만들어진 배열을 따라 alignment를 하기 위해서이다. 단지 score를 위해서라면 모든 score값을 저장해둘 필요가 없이, 특정 row나 col을 업데이트하기 위해 row-1 또는 col-1의 배열 값만 저장해두면 된다. 따라서 이러한 방법을 이용하여 모든 배열을 저장하지 않고 우선 가장 score가 높은 5부분을 찾는다.

(아래 코드에서는 main()에서 findMaxPoint(DNA1_size, DNA2_size, maxC);가 이러한 역할을 해준다.)

- 이렇게 가장 high score를 가지는 sequence를 끝을 앞의 과정을 통해 알 수 있다. 이 sequence의 끝의 index가 DNA1에서의 i번, DNA2에서의 j번이라고 칭한다면 이후는 DNA1[i-1000:i], DNA2[j-1000:j]의 부분만을 참고하여 이차원 배열을 선언하여 다시 dynamic programming 방법으로 이차원 배열에 score를 채우고, 이차원 배열[1000][1000]에서 거슬러 올라가면서 alignment를 진행하면 된다. 물론 이 과정에서는 local sequence의 길이가 1000이 넘지 않는다는 것을 가정한다.

(아래 코드에서는 main()에서 getSequence(maxC.getCoo(i), DNA1_size, DNA2_size, out); 부분이 그 역할을 해준다.)

- 그 외에 gap이 처음으로 나타났는지, gap/mismatch가 5회 연속 일어났는지 판단하기 위하여 scoring을 하기 위한 이차원 배열을 만들 때, score뿐만 아니라 처음 나타난 gap인지 아닌지 판단하기 위한 gap_frist와 match되지 않은게 몇 번이나 연속으로 진행되었는지 확인하기 위한 loseCount등도 같이 저장하였다.

(아래 소스코드에서 coo class참고)

시간복잡도와 공간복잡도

우선 공간 복잡도는 두 가지 부분에서 생각할 수 있다.

1) 프로그래머가 임의로 가정한 local common sequence의 길이를 k라고 둔다면, k*k의 2차원 배열을 사용하게 된다.

2) 주어진 염기서열의 길이가 l이라면 처음 최대 score를 가진 부분을 찾기 위해 길이가 l인 1차원 배열을 2개 사용하게 된다.

따라서 공간복잡도는 O(2*l + k*k) 이다.

(l은 주어진 염기서열의 길이, k는 프로그래머가 임의로 설정한 common sequence의 최대길이)

시간복잡도는 세 가지 부분에서 생각할 수 있다.

1) 처음 가장 high score를 찾기 위해 연산하는 부분에서는 주어진 염기서열의 길이가 l1, l2라면 l1*l2의 번의 연산을 진행하게 된다.

2) 이후 찾은 high score기준으로 k*k 2차원 배열을 만들는 과정에서는 k*k번의 연산을 진행하게 된다.

3) k*k 2차원 배열을 만든 후 거슬러 올라가는 alignment 과정에서는 최대 (k+k)길이의 sequence를 얻을 수 있고, 각 염기서열을 결정하는데는 3번의 비교연산을 하게된다. 따라서 6k번의 연산이 진행된다.

따라서 시간복잡도는 O(l1*l2 + k(k+6))으로 계산된다.

(l1, l2는 주어진 염기서열의 길이, k는 프로그래머가 임의로 설정한 common sequence의 최대길이)

소스코드

```
#include <iostream >
#include <fstream >
#include <algorithm >
#include <string >
#define MATCH 3
#define MISMATCH (-2)
#define GAP_first (-2)
#define GAP (-5)
#define size 1000
#define resultNum 5
#define lineSet 70
using namespace std;
char * DNA1;
char * DNA2;
/* Element
  Element is unit to store(for dynamic programming)
*/
class Element{
public:
  short value; //save score
  short  loseCount;  //save  continuity  of  not  match,  to  find  out
continuity of gap or mismatch
  bool first_gap; //true => first gap, false => not first gap
                //to find out this gap is first or not. first gap penalty
is -5, whereas other gap penalty is -2.
  Element(){
    value =0;
    loseCount =0;
    first_gap =true;
  }
};

/* coo
  this class is to save information about maximum 5 score point.
*/
class coo{
public:
  int row;  //row index (row is associated with DNA2)
  int col;  //col index (col is associated with DNA1)
  int value; //save score
  string res;
  coo(){
    row =0; col =0;
    value =0;
  }
  coo(const coo & a){
    row = a.row; col = a.col; value = a.value;
  }
  coo(int _row, int _col, int _value){
    row = _row; col = _col; value = _value;
  }
};
```

```
bool operator <= (coo & co1, coo & co2){
  if(co1.value <= co2.value)
    return true;
  return false;
}
bool operator < (coo & co1, coo & co2){
  if(co1.value < co2.value)
    return true;
  return false;
}
/* MaxCoo
  this class is for data structure to manage five maximum score point(=name
class coo)
  this class is shell for coo array[resultNum].
  this class help insert to coo array, and get element from coo array.
*/
class MaxCoo{
public:
  coo cooArray[resultNum];
  void insert(coo newCo){//insert newCo if newCo's value(=score) is larger
than cooArray
    if(newCo <= cooArray[resultNum -1]){
      return;
    }
    int i;
    for(i =0; i < resultNum; i ++){
      if(cooArray[i] < newCo)
        break;
    }
    for(int j = resultNum -1; j > i; j --){
      cooArray[j] = cooArray[j -1];
    }
    cooArray[i] = newCo;
  }
  void print(){//this is for debugging
    cout <<"========= max coo ==========="<<endl;
    for(int i =0; i < resultNum; i ++){
      cout <<"("<< cooArray[i].row <<","<< cooArray[i].col <<") :: ";
      cout << cooArray[i].value;
      cout <<endl;
    }
  }
  coo getCoo(int index){//get coo from cooArray by index
    return cooArray[index];
  }
};
```

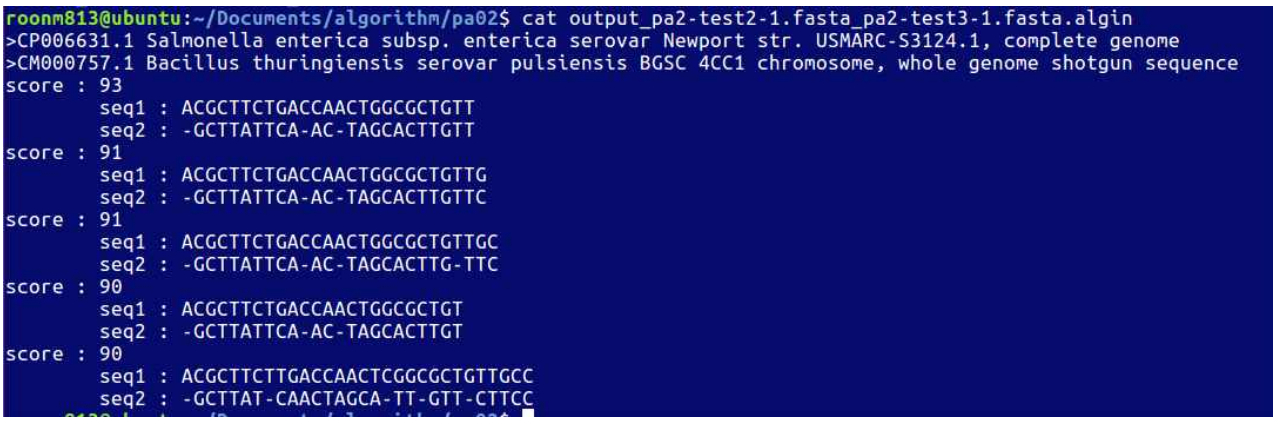
```
/*printDNA || this is for debugging.
  usage : print out DNA char array
  arg : char* DNA (= dna array)
        : int DNA_size (= size of dna array)
*/
void printDNA(char * DNA, int DNA_size);
/*initDNA
  usage : get DNA sequence size and get DNA sequence. also print out input
file info.
  arg : char** DNA (=get DNA sequence in this array)
         int* DNAsize (= get DNA sequence size int this variable)
*/
void initDNA(char ** DNA, ifstream & in, int * DNAsize, ofstream & out);
/*maxInt
  usage : get maximun integer between val1, val2 and val3
*/
int maxInt(int val1, int val2, int val3);
/*fillLine
  usage : fill dp2 from dp1, and also fill maximum score information in maxC
*/
void fillLine(int DNA1_size, Element dp1[], Element dp2[], MaxCoo & maxC, int
l);
/* findMaxPoint
  usage : maximum score index and score value , this information is stored in
maxC and pass through maxC;
*/
void findMaxPoint(int DNA1_size, int DNA2_size, MaxCoo & maxC);
/*fillTarget
  usage : fill dp(=target) using information DNA1[row:row+row_size],
DNA2[col:col+col_size]
*/
void fillTarget(Element target[][size +1],int row, int row_size, int col,
int col_size);
/* traceBack
  usage : find sequence alignment from Element target[][], and print out
result in ofstream& out
*/
void traceBack(Element target[][size +1], int row_base, int col_base, int
row_size, int col_size, ofstream & out);
/* getSequence
  usage : from co(which is storage for max score and and max score index),
find out sequence alignment
       : this function call fillTarget(to fill dp) -> traceBack(to find and
print alignment)
*/
void getSequence(coo co, int DNA1_size, int DNA2_size, ofstream & out);
```

함수 정의와 함수에 관한 설명

```
int main(int argv, char * argc[]){
  string filename1, filename2, filename3;
  if(argv <3)
    cout     <<"to     few     argument"<<endl     <<"$pa02     "<<"[filename1]
[filename2]"<<endl;
  else{
    filename1 = argc[1];
    filename2 = argc[2];
    filename3 ="output_"+ filename1 +"_"+ filename2 +".algin";
  }
  ifstream in1(filename1);
  ifstream in2(filename2);
  ofstream out(filename3);
  int DNA1_size;
  int DNA2_size;
  initDNA(&DNA1, in1, &DNA1_size, out);
  initDNA(&DNA2, in2, &DNA2_size, out);
  MaxCoo maxC;
  findMaxPoint(DNA1_size, DNA2_size, maxC);
  for(int i =0; i < resultNum; i ++)
    getSequence(maxC.getCoo(i), DNA1_size, DNA2_size, out);
       return 0;
}
```

구체적인 함수 내부는 소스코드 따로 첨부.

실행 결과 예시



c++14 기준으로 컴파일 해야 에러 없이 컴파일 된다.

pa2-test2-1.fasta와 pa2-test3-1.fasta를 비교한 결과는 다음과 같다.