**LAB [5], [2/7/2019] MCS 253P**

**Name: Yu Qin,**

**Partner: Ethel Hoshi**

**General Problem Description**

Optimal local align two DNA sequences. Input is two DNA sequence strings, and outputs the strings with dash “-” inserted where another nucleotide must be inserted to make them match.

Hint: use [Smith-Waterman algorithm](http://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm) or [Needleman–Wunsch algorithm](https://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm).

e.g.,

Input

S1 = ACACACTA

S2 = AGCACACA

Output

S1 = A-CACACTA

S2 = AGCACAC-A

Input

S1 = A

S2 = B

Output

S1 = A

S2 = B

**Additional Problem Specifics**

* Define [substitution matrix](https://en.wikipedia.org/wiki/Substitution_matrix) and the [gap-scoring](https://en.wikipedia.org/wiki/Gap_penalty) scheme, I’d better use as API function.
* How will the above parameters be set?
* negative scoring matrix cells are set to zero, for Optimal local align. I agree that in regions of low similarity between distantly related biological sequences, when mutations have added too much 'noise' over evolutionary time, it allows for a meaningful comparison of those regions.
* Traceback procedure starts at the highest scoring matrix cell and proceeds until a cell with score zero is encountered, yielding the highest scoring local alignment.
* Don't require of O(n) space complexity, which means I can first use a matrix for saving the middle results, which, of course, can be recorded to obtain O(n).
* When traceback, begin with the highest score, end when 0 is encountered; rather than begin with the cell at the lower right of the matrix, end at top left cell.
* The only max result;
* The only median value in traceback;
* The ending point is set to be 0, whether it is selected to be the upper, the left, the upper left, it doesn’t matter in S-W since it will disregard the last letter.

**Proposed Algorithm**

#include <iostream>

#include <cstdio>

#include <math.h>

#include <vector>

#include <math.h>

using namespace std;

float SubstitutionMatrix(char a, char b ){

float sub = (a==b)? +3: -3;

return sub;

}

float Penalty(){

float pnl = 2;

return pnl;

}

bool isstart = false;

float MyMax3(float a, float b, float c){

float tmp;

tmp = max(a,b);

return max(tmp, c);

}

float MyMax4(float a, float b, float c, int d){

float tmp;

tmp = MyMax3(a,b,c);

return max<float>(tmp, d);

}

void prev(int& i, int& j, vector<vector<float>> H, string& out1, string& out2){

float a = H[i-1][j-1];

float b = H[i-1][j];

float c = H[i][j-1];

float maxprev = MyMax3(a,b,c);

if (a == 0 || b == 0 || c == 0)

isstart = true;

else{

if (maxprev == a){

i = i-1;

j = j-1;

}

else if (maxprev == b){

i = i-1;

j = j;

out2[j] = '-';

}

else {

i = i;

j = j-1;

out1[j] = '-';

}

}

}

void BackTracePrint(vector<vector<float>> H, pair<int, int> maxlocation, string S1, string S2){

int i = maxlocation.first;

int j = maxlocation.second;

string out1 = S1.substr(0,i+1);

string out2 = S2.substr(0,j+1);

while (isstart == false){

prev(i,j,H,out1,out2);

}

out1 = out1.substr( i, out1.size() - i );

out2 = out2.substr( j, out2.size() - j );

cout<<"S1 = "<<out1<<endl;

cout<<"S2 = "<<out2<<endl;

}

void OptimalLocalAlign(string S1, string S2){

int size1 = S1.size() + 1;

int size2 = S2.size() + 1;

vector<vector<float>> H(size1, vector<float>(size2));

//initialize

float maxscore = 0;

pair<int, int> maxlocation;

for (int i = 0; i < size1; i++){

H[i][0] = 0;

}

for (int j = 1; j < size2; j++){

H[0][j] = 0;

}

for (int i = 1; i < size1; i++){

for (int j = 1; j< size2; j++){

H[i][j] = MyMax4( H[i-1][j-1] + SubstitutionMatrix(S1[i],S2[j]),

H[i][j-1] - Penalty(),

H[i-1][j] - Penalty(),

0);

if (H[i][j] > maxscore){

maxscore = H[i][j];

maxlocation = make\_pair(i,j);

}

}

}

cout<<"maxscore: "<<maxscore<<endl;

BackTracePrint(H, maxlocation, S1, S2);

}

int main(){

//string S1 = "ACACACTA";

//string S2 = "AGCACACA";

string S1 = "GGTTGACTA";

string S2 = "TGTTACGG";

OptimalLocalAlign(S1, S2);

return 0;

}