Analysis of Algorithms

Project 2: Global Sequence Alignment

Nurefşan Sertbaş
040110078
16 Nisan 2016

Project Report

Part a:

In this part we tried to implement Needleman-Wunsch algorithm for global alignment of DNA sequences. First, we will give the procedure that represents how the algorithm works then we will give the execution results.

Sequence 1 is given as ISALIGNED and Sequence 2 is given as THISLINE.

First Step: Initialization

ISALIGNED → M=9

THISLINE → N=8

Then, we create (M+1)(N+1)=10x9 matrix and initialize it.

	0	I 1	S 2	A 3	L 4	I 5	G 6	N 7	E 8	D 9
0	0	-4	-8	-12	-16	-20	-24	-28	-32	-36
T 1	-4									
H 2	-8									
I 3	-12									
S 4	-16									
L 5	-20									
I 6	-24									
N 7	-28									
E 8	-32									

Matrix fill (scoring) & Trace back (alignment)

As stated in the Project description:

- $S_{i,j}$ = 4 (match)
- $S_{i,j}$ = -1 (miss match score)
- w = -4 (gap penalty)

Then, we compute the following value for each cell

Mi,j = MAXIMUM [

Mi-1, j-1 + Si,j (match/mismatch in the diagonal),

Mi,j-1 + w (gap in sequence #1),

Mi-1,j + w (gap in sequence #2)]

The details of the calculation is given in following tables:

M[1,1]= max(0-1, -4-4, -4-4)=-1 M[1,2]= max(-4-1,-1-4, -8-4)=-5 M[1,3]= max(-8-1, -5-4, -12-4)=-9 M[1,4]=max(-12-1, -9-4,-16-4)=-13 M[1,5]=max(-16-1,-13-4,-20-4)=-17 M[1,6]= max(-20-1, -17-4,-24-4)=-21 M[1,7]= max(-24-1, -21-4,-28-4)=-25 M[1,8]= max(-28-1,-25-4,-32-4)=-29 M[1,9]= max(-32-1, -29-4,-36-4)=-33	M[2,1]= max(-4-1,-8-4,-1-4)=-5 M[2,2]= max(-1-1,-5-4,-5-4)=-2 M[2,3]= max(-5-1,-2-4,-9-4)=-6 M[2,4]=max(-9-1,-6-4,-13-4)=-10 M[2,5]=max(-13-1,-10-4,-17-4)=-14 M[2,6]= max(-17-1,-14-4,-21-4)=-18 M[2,7]= max(-21-1,-18-4,-25-4)=-22 M[2,8]= max(-25-1,-22-4,-29-4)=-26 M[2,9]= max(-29-1,-26-4,-33-4)=-30	M[3,1]= max(-8+4,-12-4,-5-4)=-4 M[3,2]= max(-5-1,-4-4,-2-4)=-6 M[3,3]= max(-2-1,-6-4,-6-4)=-3 M[3,4]= max(-6-1,-3-4,-10-4)=-7 M[3,5]= max(-10+4,-7-4,-14-4)=-6 M[3,6]= max(-14-1,-6-4,-18-4)=-10 M[3,7]=max(-18-1,-10-4,-22-4)=-14 M[3,8]=max(-22-1,-14-4,-26-4)=-18 M[3,9]=max(-26-1,-18-4,-30-4)=-22

M[4,1]= max(-12-1,-16-4,-4-4)=-8 M[4,2]= max(-4+4,-8-4,-6-4)=0 M[4,3]= max(-6-1,0-4,-3-4)=-4 M[4,4]=max(-3-1,-4-4,-7-4)=-4 M[4,5]=max(-7-1,-4-4,-6-4)=-8 M[4,6]= max(-6-1,-8-4,-10-4)=-7 M[4,7]= max(-10-1,-7-4,-14-4)=-11 M[4,8]= max(-14-1,-11-4,-18-4)=-15 M[4,9]= max(-18-1,-15-4,-22-4)=-19	M[5,1]= max(-16-1,-20-4,-8-4)=-12 M[5,2]= max(-8-1,-12-4,0-4)=-4 M[5,3]= max(-0-1,-4-4,-4-4)=-1 M[5,4]=max(-4+4,-1-4,-4-4)=0 M[5,5]=max(-4-1,0-4,-8-4)=-4 M[5,6]= max(-8-1,-4-4,-7-4)=-8 M[5,7]= max(-7-1,-8-4,-11-4)=-8 M[5,8]= max(-11-1,-8-4,-15-4)=-12 M[5,9]= max(-15-1,-12-4,-19-4)=-16	M[6,1]=max(-20+4,-24-4,-12-4)=-16 M[6,2]=max(-12-1,-16-4,-4-4)=-8 M[6,3]=max(-4-1,-8-4,-1-4)=-5 M[6,4]=max(-1-1,-5-4,-0-4)=-2 M[6,5]=max(0+4,-2-4,-4-4)=4 M[6,6]=max(-4-1,4-4,-8-4)=0 M[6,7]=max(-8-1,0 -4,-8-4)=-4 M[6,8]=max(-8-1,-4-4,-12-4)=-8 M[6,9]=max(-12-1,-8-4,-16-4)=-12

M[7,1]= max(-24-1,-28-4,-16-4)=-20 M[7,2]= max(-16-1,-20-4,-8-4)=-12 M[7,3]= max(-8-1,-12-4,-5-4)=-9 M[7,4]=max(-5-1,-9-4,-2-4)=-6 M[7,5]=max(-2-1,-6-4,4-4)=0 M[7,6]= max(4-1,0-4,0-4)=3 M[7,7]= max(0+4,3-4,-4-4)=4 M[7,8]= max(-4-1,4-4,-8-4)=0 M[7,9]= max(-8-1,0-4,-12-4)=-4	M[8,1]= max(-28-1,-32-4,-20-4)=-24 M[8,2]= max(-20-1,-24-4,-12-4)=-16 M[8,3]= max(-12-1,-16-4,-9-4)=-13 M[8,4]=max(-9-1,-13-4,-6-4)=-10 M[8,5]=max(-6-1,-10-4,-0-4)=-4 M[8,6]= max(0-1,-4-4,3-4)=-1 M[8,7]= max(3-1,-1-4,4-4)=2 M[8,8]= max(4+4,2-4,0 -4)=8 M[8,9]= max(0-1,8-4,-4-4)=4

As a result of those calculations we have obtained below table.

	0	I 1	S 2	A 3	L 4	I 5	G 6	N 7	E 8	D 9
0	0	-4	-8	-12	-16	-20	-24	-28	-32	-36
T 1	-4	-1	-5	-9	-13	-17	-21	-25	-29	33
H 2	-8	-5	-2	-6	-10	-14	-18	-22	-26	-30
I 3	-12	-4	-6	-3	-7	-6	-10	-14	-18	-22
S 4	-16	-8	0	-4	-4	-8	-7	-11	-15	-19
L 5	-20	-12	-4	-1	0	-4	-8	-8	-12	-16
I 6	-24	-16	-8	-5	-2	4	0	-4	-8	-12
N 7	-28	-20	-12	-9	-6	0	3	4	0	-4
E 8	-32	-24	-16	-13	-10	-4	-1	2	8	4

Also, we have written C++ code for the implementation of above scenario and its execution results by using SSH is given below.

```
[sertbasn@ssh ~]$ g++ 040110078_part1.cpp -o 040110078
[sertbasn@ssh ~]$ ./040110078

Generated matrix:
0 -4 -8 -12 -16 -20 -24 -28 -32 -36
-4 -1 -5 -9 -13 -17 -21 -25 -29 -33
-8 -5 -2 -6 -10 -14 -18 -22 -26 -30
-12 -4 -6 -3 -7 -6 -10 -14 -18 -22
-16 -8 0 -4 -4 -8 -7 -11 -15 -19
-20 -12 -4 -1 0 -4 -8 -8 -12 -16
-24 -16 -8 -5 -2 4 0 -4 -8 -12
-28 -20 -12 -9 -6 0 3 4 0 -4
-32 -24 -16 -13 -10 -4 -1 2 8 4

Solution:
0 0 0 0 0 0 0 0 0 0
-4 0 0 0 0 0 0 0 0 0
-8 0 0 0 0 0 0 0 0 0
0 -4 0 0 0 0 0 0 0 0
0 0 0 -4 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 4 0 0 0 0
0 0 0 0 0 0 0 4 0 0
0 0 0 0 0 0 0 0 8 4

One of the optimal alignment is
__ISALIGNED
THIS_LI_NE_

Max global alignment score is 6

[sertbasn@ssh ~]$
```

Also, we execute the code for the sample that was given by a link in the project description. Our aim was checking the results of the code and we saw that it operates properly as given in below.

```
[sertbasn@ssh ~]$ g++ 040110078_part1.cpp -o 040110078
[sertbasn@ssh ~]$ ./040110078

Generated matrix:
0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 2 2 2 2
0 1 2 2 2 2 2 2 2 2 2 3
0 1 2 2 3 3 3 3 3 3 3 3
0 1 2 2 3 3 4 4 4 4 4 4
0 1 2 2 3 3 4 4 5 5 5 5
0 1 2 3 3 3 4 5 5 5 5 6

Solution:
0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 2 2 0 0 0 0 0 0 0
0 0 0 0 0 3 0 0 0 0 0 0
0 0 0 0 0 0 4 4 0 0 0 0
0 0 0 0 0 0 0 0 5 5 5 0
0 0 0 0 0 0 0 0 0 0 0 6

One of the optimal alignment is
GAATTCAGTTA
GGA_TC_G__A

Max global alignment score is 7

[sertbasn@ssh ~]$
```

Connected to ssh.itu.edu.tr

Part b:

In this part, we have written C++ code again and it operates as following:

```
75  void readFastaSeq(){
76      char line[256];
77      vector<char>::iterator it;
78
79      FILE *ptr_file=fopen("human-hemoglobin-sequence.fasta", "
80      if(ptr_file==NULL) {
81          printf("An error has occured: can't open file\n");
82          exit(1);}
83
84      char * fileinfo=fgets(line, sizeof(line), ptr_file);
85      char ch = 'x';
86      seq1.push_back(ch);
87
88      while (ch != EOF ) {
89          ch=(char)fgetc(ptr_file);
90          if(ch!='\n')
91          seq1.push_back(ch);
92      }
93
94      seq1.pop_back();
95
96      //reading the second sequence
97      ptr_file=fopen("mouse-hemoglobin-sequence.fasta", "r");
98      if(ptr_file==NULL) {
99          printf("An error has occured: can't open file\n");
100         exit(1);}
101
102     fileinfo=fgets(line, sizeof(line), ptr_file);
103     ch = 'x';
104     seq2.push_back(ch);
105     while (ch != EOF ) {
106         ch=(char)fgetc(ptr_file);
107         if(ch!='\n')
108         seq2.push_back(ch);
109     }
```

It reads the fasta files that belongs to human and mouse respectively. It passes first line that gives information about the content then it copies the sequence into the vectors named as ‘seq1’ and ‘seq2’ until see the end of file.

Then, it creates the seq1 and seq2 in the main by using above function. Then it initializes the matrix. After that it uses the below function in order to read the substitution matrix from the BLOSUM62.txt file.

```
void readSubstitutionMatrix(){
    char line[256];

    FILE *ptr_file=fopen("BLOSUM62.txt", "r");
    if(ptr_file==NULL) {
        printf("An error has occured: can't open file\n");
        exit(1);}

    char * buffer=fgets(line, sizeof(line), ptr_file);
    while(buffer[0]=='#')
        buffer=fgets(line, sizeof(line), ptr_file);

    char ch = 'x';
    int count=0;
    while (ch != '\n' ) {
        ch=buffer[count++];
        if(ch!=' ')
        header.push_back(ch); }

    header.pop_back(); //ARNDCQEGHILKMFPSTWYVBZX*

    int i=0,j=0,k=0;
    char tmp;
    while(i<matrixsize){//dosyanın sonuna gelmediği sürece
        buffer=fgets(line, sizeof(line), ptr_file);
        for(k=3;k<strlen(buffer);k=k+3){
                tmp=buffer[k-1];
                if(tmp==' ')
                    sub_matrix[i][j++]=atoi(&buffer[k]);
                else if(tmp=='-')
                    sub_matrix[i][j++]=-1*atoi(&buffer[k]);
                else{
                    char m[2];
                    m[0]=buffer[k-1];
                    m[1]=buffer[k];
                    sub_matrix[i][j++]=atoi(m); }}
        i++;//next row
        j=0;
    }
     fclose(ptr_file);}
```

Assume that the substitution matrix has m rows so that the readSubstitutionMatrix function runs with O(m*m) time.

```
int getSubstitutionValue(char c1,char c2){
    int ix=0;
    char elements[matrixsize];

    for (myit=header.begin(); myit != header.end(
        elements[ix++]=*myit;

    int i=0,j=0;

    while(elements[i]!=c1)
        i++;
    while(elements[j]!=c2)
        j++;

    return sub_matrix[i][j];
}
```

This function uses the outputs of the readSubstitutionMatrix function. It operates in O(m) time and it takes two char as a parameter. Then, it returns the related substitution value.

To sum up, the program reads the fasta files and calculate the global alignment of given sequences by using the same code that is used in part1. Note that there is one difference that the algorithm uses an external substitution matrix. The output of the codes is given below as a screenshot.

```
[sertbasn@ssh ~]$ g++ 040110078_part2.cpp -o 040110078
[sertbasn@ssh ~]$ ./040110078
One of the optimal alignment is
M_VHL___TPEEKSA_V__TALW_GK_____VNVDE_VG_GE_AL_GRL___LVVYP___WTQRFF__ESFGDL__STPDAVMG____NPKVKAHG_K__
|   |       |: | :    :|  |          | :|  |  |  ||      ||     |  :|    | ||          |        :||| |
MS__LMKN___ER_AIIMS__MWE_KMAAQA____EPIGT_ET_LE_RLFCS___YPQTK_T__YFPH__F_DLHH_______GSQQL____RAHGFKIM

__KVLG_A__FS_D_GL__A_______H___L__D__NLKGTF__ATLSELHC_DKLHV________D__P___E____NF____________R
   | | |   | |  |  |       |   |  |  |    |    ||  ||   | |        |  |   |     |             |
TA_V_GDAVK_SIDN_LSSALTKLSELHAYILRVDPVN____FKL__LS__HCL__L_VTMAARFPADFTPEVHEAWDK_FMSILSSILTEKYR


##########      SUMMARY ##########
Length(with gap)  :194
Identity                :47/194  24(%)
Similarity       :6/194  3(%)
Gaps             :141/194  72(%)
Score            :257
[sertbasn@ssh ~]$
```

The pseudo code of the whole program is given below.

	Operation	Complexity
1	**Line 140:** readFastaSeq()	O(M) + O(N) where M and N are the sequence lengths
2	**Line 141:** readSubstitutionMatrix()	O(m*m) where m is the number of rows in the substitution matrix
3	**Line 149-160:** creating seq1 and seq2 arrays	O(M) + O(N)
4	**Line 164-198:** initialize and fill the matrix and directions of the values Note that it calls getSubstitutionValue() for each round	O(M.N.m)
5	**Line 201-224:** Tracing back	O(M.N)+O(N)
6	**Line 241-291:** Aligned sequence operations	O(M)
7	**Line 291-322:** Printing the results and summary	O(M)

By using dynamic programming approach we saved the values in the matrix then we traced back to find the optimal solution. If we have not used the precomputed matrix, there was no difference with brute force solution. Unlike dynamic programming, brute force approach tries all the possibilities in order to find the optimal one.