Istanbul Technical University
Faculty of Computer and Informatics

BLG336E Analysis of Algorithms 2
Project 2
Report

Cem Yusuf Aydoğdu
150120251

a) Rule: $S_{ij} = max \begin{cases} S_{i-1,j-1} + m_{ij} \\ S_{i-1,j} + g \\ S_{i,j-1} + g \end{cases}$, i for rows, j for columns

where $m_{ij} = \begin{cases} 4, & match \\ -1, & mismatch \end{cases}$ and $g = -4$ for gap

According to Needleman-Wunsch algorithm, initially first row and first column of scoring matrix is filled, according to gap score.

Then, scores of inner cells are calculated according to the rule above; by selecting the maximum value of left, up and upper left cells.

Finally, alignment of two strings are calculated by back tracking scores starting from the bottom right position of the score matrix.

Back tracking method is based on selecting a cell which is responsible for score of the current cell. Selecting left or upper cell corresponds to a gap, but selecting the upper left cell is interpreted as take a character from both strings.

It should be considered that there may be more than one backtrack paths, depending on the priority of selecting the corresponding responsible cell. Also, the backtracking stage can be accomplished by holding location of each cell's score processor.

In the example below, the back tracking stage is focused to obtain maximum score, therefore priority of left, up and upper left cells are arranged for this purpose.

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

Result of the alignment:

```
--ISALIGNED
..|.|.|.|.
THIS-LI-NE-
```

Total score of the alignment: $4 + 8 + 4 + 4 - 4 - 4 = 12$

Implementation Details

The implementation consist **SequenceAlignment** class for algorithm (*alignSequences()* function) and data reading operations, **Fasta** struct for FASTA data format and **SubstitutionMatrix** struct for BLOSUM file, which holds match/mismatch/gap score values in STL map data type.

In the implementation of the algorithm, calculating scoring table is same as the example in the a) above. However, back tracking stage is not flexible since it checks upper left, left and upper cells respectively and selects the corresponding cell immediately when it discovers that cell is responsible to score value of current cell. Pseudocode of the algorithm is given below. Latex source of the pseudocode is also given with the project.

Algorithm 1 Needleman-Wunsch for Global Alignment of DNA sequences

Initialize strings $str1$, $str2$ with gap symbol in front
Initialize 2D scores array $S \leftarrow 0$ with string sizes
for each cell in the first row of S **do**
    Assign gap scores
end for
for each cell in the first column of S **do**
    Assign gap scores
end for

for $i := 1$ **do** size of $str2$ ▷ Calculate scoring table S
    for $j := 1$ **do** size of $str1$

$$S_{i,j} = max \begin{cases} S_{i-1,j-1} + m_{i,j} & \triangleright\ m \text{ is match/mismatch score} \\ S_{i-1,j} + g & \triangleright\ g \text{ is gap penalty score} \\ S_{i,j-1} + g \end{cases}$$

    end for
end for

Initialize empty strings $seq1$ for $str1$, $seq2$ for $str2$ ▷ Backtrack
Set $i \leftarrow$ length of str2, $j \leftarrow$ length of str1, $totalScore \leftarrow 0$
while $i > 0$ and $j > 0$ **do**
    if Score of $S_{i,j}$ calculated from $S_{i-1,j-1}$ **then**
        Append str1[j] to front of seq1
        Append str2[i] to front of seq2
        Decrement i and j by 1
        Increment $totalScore$ by $S_{i,j}$
    else if Score of $S_{i,j}$ calculated from $S_{i-1,j}$ **then**
        Append gap symbol to front of seq1
        Append str2[i] to front of seq2
        Decrement i by 1
        Increment $totalScore$ by $S_{i,j}$
    else if Score of $S_{i,j}$ calculated from $S_{i,j-1}$ **then**
        Append str1[j] to front of seq1
        Append gap symbol to front of seq2
        Decrement j by 1
        Increment $totalScore$ by $S_{i,j}$
    end if
end while

for each character of aligned sequences **do** ▷ Calculate metrics
    Count $identity$, $similarity$ and gap
end for

Output aligned sequences $seq1$, $seq2$

b) Complexity of the algorithm is determined from calculating scoring table part through lines 10-14 in the pseudocode above, which is $O(nm)$ where n and m are sizes of given strings.

Principle of the dynamic programming approach is based on dividing the main problem into easier computable sub-problems and solving the main problem using solutions of these sub-problems. The algorithm applies this principle in the nested for loops through lines 10-14.

Standard methods without dynamic programming would have time complexity exponentially related with sizes of sequences, for this problem.

An output from the program is shown below:

```
########################################################################################
#
# Length:       118 (with gaps)
# Identity:     23/118 (0.194915%) ("|")
# Similarity:   23/118 (0.194915%) ("|" and ":")
# Gaps:         85/118 (0.720339%)
# Score:        379
#
#
########################################################################################

human             1      ----KVK--AH---G---K-K---------V----L-G-----A----FS      50
                         ....|....|........|.............................|.
mouse             1      MSLMK-NERA-IIMSMWEKMAAQAEPIGTETLERLFCSYPQTKTYFPHF-      50

human             51     D---G---L-AHLDN-L----------KG---TF--A-T-LSELH---CD      100
                         |...|...|.||.............|........|.|.|||||.....
mouse             51     DLHHGSQQLRAH---GFKIMTAVGDAVK-SIDNLSSALTKLSELHAYI--      100

human             101    KLH-VDP-ENF----R--       118
                         .|..|||..||.......
mouse             101    -L-RVDPV-NFKLLS-HC       118


########################################################################################
```