BLG 336E – Analysis of Algorithms II

Homework 3

Local Sequence Alignment with Smith-Waterman

150170092 : Barış İncesu

19 MAY 2020

Part 2. Report

2.1.a

```
// Local Alignment Runthrough

M = length of sequence 1 // lengthOfs1
N = length of sequence 2 // lengthOfs2
S = dynamic programming matrix of size M x N

// smith_waterman function

S[0,0] = 0
for i = 1 to M do
        S[i,0] = 0
for i = 1 to N do
        S[i,0] = 0
        for i = 1 to M
                       0
                       S[i-1,j-1] + match or mismatch
S[i,j] = max
                       S[i-1,j] + gap
                       S[İ,j-1] + gap

return S[M,N]

// Local Alignment Traceback

// Traceback_part1
maxNumber = maximum value in S initialized with 0
for i = 0 to M do
        for i = 0 to N do
        if (S[i,j > maxNumber)
                maxNumber = S[i,j]
return maxNumber

// Traceback_part2
targetNumber= maximum value number in S initialized with 0
for i = 0 to M do
        for i = 0 to N do
        if (S[i,j] == maxNumber)
                targetNumber++
return targetNumber++

// Traceback_part3
targetPosition[targetNumber] keeps every maxNumber row number.
targetLength[targetNumber] keeps each maxNumber diagonally from 0.
offset = offset counter for these arrays.

for i = 0 to M do
        for i = 0 to N do
        if (S[i,j] == maxNumber)
        tempi = i
        tempj = j
```

```
        while ( S[tempi, tempj] != 0)
                tempi = tempi -1
                tempj = tempj - 1
                targetLength[offset] = targetLength[offset] + 1
        targetLength[offset] = i
offset ++
return targetNumber and targetLength
```

For the desired output forms and file reading operations, the following operations have been made.

```
// Assignment

File reading line by line.
        push back for each line to vector
        sort the vector
for i = 0 to vector.size do
        for j = i + 1 to vector.size do
        s1 equal to vectors ith string
        s2 equal to vectors jth string
        call smith_waterman function
        call traceback_part1
        print Score = maxNumber
        call traceback_part2
        call_traceback_part3
        After detect target positions and length create substrings
        Create a set structure for keep substring without repetition.
        Print substrings
```

2.1.b.

Optimal alignment score for aligning M and N Time Complexity: **O(M*N)**

2.2.a

Calculations on brute force $4^{(m+n)}$ on the other hand, Smith-Waterman is processing as much as **m * n**.

2.2.b

Since Brute Force compares all possible substructures with each other, the smallest word substring will be maximum kept for it. This means that they are **single-length substrates, or letters**. Maximum kept equals **the number of letters of the lowest number of letters** among 2 sequences.

Smith-Waterman, on the other hand, has to keep all values in a matrix structure in order to reach the result. The size of the matrix is **m * n.**

2.2.c

Since the Bf algorithm is not applied in the study, this question can be answered according to the data in 1a. Time comparisons are also considered to be $4^{(m+n)}$ and **m * n.**