İlya Mustafa Nuhi
150180710

HOMEWORK-3 REPORT

1-) Report your problem formulation:

a) Write your pseudo-code.

```
//Filling matrix by smith-waterman algorithm with  given match, mismatch, gap values:

For (i = 1; i< (length of word1 +1); i++){
        For(j=1 ; j < (length of word2 +1); j++){
                Create int vector “values”
                İf (word2[j-1] == word1[i-1])
/*add the match value to value of the upper left cell of the current cell and push the result to values vector*/
                        Values.push_back(cell[i-1][j-1]+MATCH)
                }
                Else{
/*add the mismatch value to value of the upper left cell of the current cell and push the result to values vector*/
                        Values.push_back(cell[i-1][j-1]+MISMATCH)
                }
/*add the gap value to left cell of the current cell and push the result to values vector*/
                Values.push_back(cell[i][j-1]+GAP)
/*add the gap value to upper cell of the current cell and push the result to the values vector*/
                Values.push_back(cell[i-1][j]+GAP)

                Values.push_back(0)
/*pick the max value among this 4 values of the values vector and write it to the current cell*/
                Cell[i][j]=max element of the values vector
        }
}
```

İlya Mustafa Nuhi
150180710

//Finding the sequences in matrixes that filled with smith-waterman algorithm:

```
For (i=1; i<(length of the word1)+1; i++){
        For(j=1; j<(length of word2)+1; j++){
                İf (cell[i][j]==max_score){
                        İnitialize temp_i = i
                        İnitialize temp_j = j
                        Create char vector “letters”
                        While(cell[temp_i][temp_j]!=0){
                                //pushing corresponding letter to the letters vector
                                Letters.push_back(word1[temp_i-1])
                                Temp_i --
                                Temp_j --
                        }
                        Reverse the letters vector
                        Create a string “common” with letters of the letters vector
                        Sequences.push_back(common)
                }
        }
}
```

b) Show the complexity of your algorithm on the pseudo-code.

Time complexity is a function describing the amount of time an algorithm takes in terms of the amount of input to the algorithm.

In our homework time complexity is filling matrix Mx N with smith-waterman algorithm and finding the sequences in these matrixes. Both implementation has 2 for loops starting from 1 to length of the word +1 means the length of the compared words M and N. So it takes 2M.N operations and since constant numbers are not important in big O notation; **our time complexity is O(M.N)**

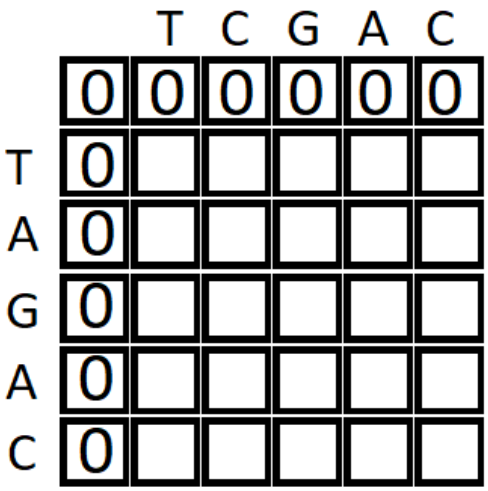
Space complexity is a function describing the amount of memory (space) an algorithm takes in terms of the amount of input to the algorithm.

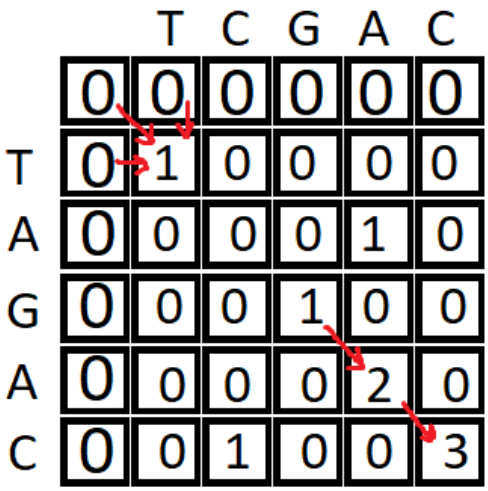
In our homework space complexity is creating memory for matrix MxN which M is the length of word1 and N is the length of word2. So **space complexity is O(M.N)**

İlya Mustafa Nuhi
150180710

2-) Analyze and compare the algorithm results with assessing all possible alignments one by one in terms of:

Smith-Waterman:





(Match:1, Mismatch:-2, Gap:-4)

Brute Force:

TCGAC- - - - - TCGA -C - - - - TCGA- -C - - - TCGA- - - C - -

- - - - - TAGAC - - - - - TAGAC - - - - - TAGAC - - - - - TAGAC

a) the calculations made,

In Smith Waterman algorithm we are scoring each element with maximum value of 4 choices which are (match/mismatch)+upper left cell, gap + left cell , gap + upper cell and 0 (3 calculations and 0) and sequence is found by tracing back from the maximum score element.

In Brute Force method we are calculating all possible allignments, score for each alignment and select the max score among all these possible allignments.

İlya Mustafa Nuhi
150180710

b) the maximum number of calculations result kept in the memory,

In my example, 5.5=25 calculated cell and 6+5=11 default 0 cells total 36 cells kept in memory with smith-waterman algorithm

It is (10! / (5!5!))^2 = 63505 calculation results kept in the memory with brute force

c) the running time

Running time can be calculated with number of calculations which i calculated it in the (b). Since i already calculated running time of the smith-waterman algorithm which is O(M.N), the running time of the brute force is O((M+N)!).