Analysis of Algorithms II № 3

Mehmet Enes Gümüş, 150170047 19/05/2021

Problem 1

a) Pseudocode

Listing 1: Pseudocode

```
struct node for every index of the matrix:
    value for scores
    next for track the path
    x,y for index characters of the strings


createMatrix():
    create space for dynamic matrix
    for i 0 to row number
        for j 0 to col number
            create node
            node.x is i'th element of first strings
            node.y is j'th element of second strings
            matrix[i][j]=node

    for i 1 to row number
        for j 1 to col number
            int max for biggest value
            if strings characters equal:
                integer1=matrix[i-1][j-1]+match
            else:
                integer2=matrix[i-1][j-1]+missmatch
                integer3=matrix[i-1][j]+gap
                integer4=matrix[i][j-1]+gap

            check the biggest value between integer1 to integer4
            connect nodes with next pointer

findMaxValue():
    max = 0
    for i 0 to row number:
        for j 0 to col number:
            if matrix[i][j]>max:
```

```
                max = matrix[i][j]

    return max

printSubs():
    max=findMaxValue()
    for i 0 to row number:
        for j 0 to col number:
            if matrix[i][j]==max
            temp = matrix[i][j]
            while(temp->next)
                print temp.x
                temp = temp->next


int main():
    string vector allStrings
    get match,missmatch,gap values from user or 1,-2,-4
    get strings --> allStrings
    sort(allStrings)

    size = allStrings.size()

    for i 0 to size:
        j = i+1
        string a=allStrings[i]
        while j smaller than size:
            string b = allStrings[j]
            pathMatrix= createMatrix(a,b,match,missmatch,gap)
            printSubs(pathMatrix)
```

b) Time Complexity

For understand the time complexity of the algorithm, we can check the code below. After creating the matrix, we will check for the maximum value at the matrix. For doing this we will consider all elements which is (m+1)*(n+1), m and n are the lengths of the strings. Time complexity= O((m+1)*(n+1)).

Listing 2: Time Complexity

```
findMaxValue():
    max = 0
    for i 0 to row number:
        for j 0 to col number:
            if matrix[i][j]>max:
                max = matrix[i][j]
    return max
```

Problem 2

Considering the example: examination-medication

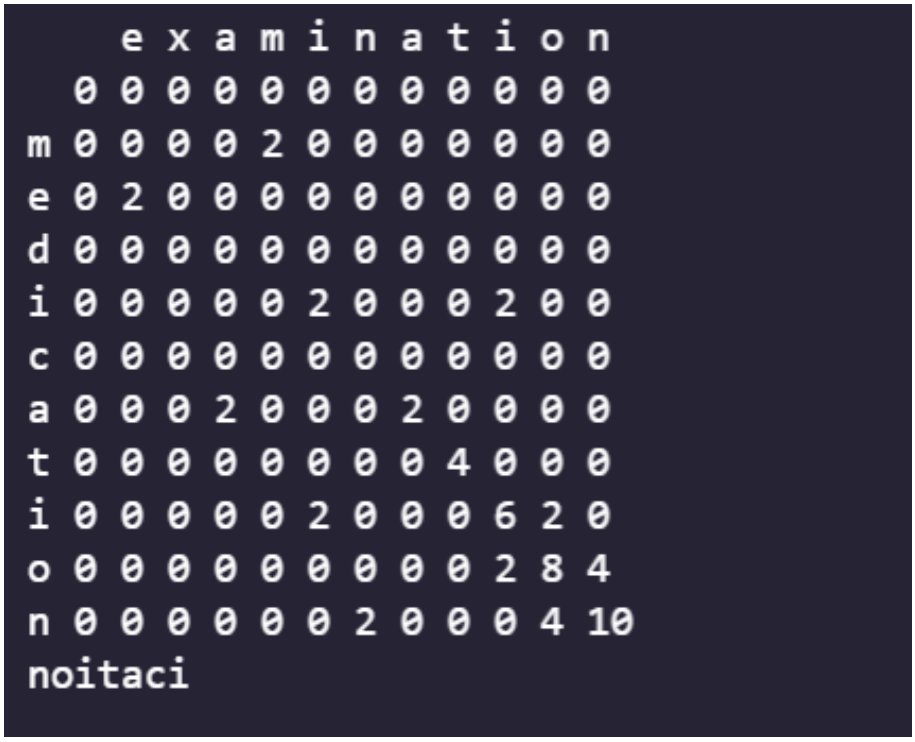

Figure 1: Example matrix

a) Calculations made

The matrix is 11x12, but first row and first columns are not calculated, they are always 0. For the rest of the matrix, we have to calculate 10x11=110 cells. While calculating these cells, we have to check for 4 different calculation results(match,mismatch,left gap, upper gap). So the total calculations made is 110x4 = 440.

b) Calculations that kept in memory

For this example, total calculations that kept in memory is 11x10 = 110.

c) Running Time

Running time is O((m+1)(n+1)) = O(12x11)= O(132).