Tracing sorted subsequence in a random matrix row-wise and column-wise

Harsh Kedia
IIB2019028

Milap Anwani
IIB2019029

Kandagatla Meghana Santhoshi
IIB2019030

***Abstract*—In this paper ,we have devised an algorithm to trace all the sorted subsequences in a random matrix both row-wise and column-wise. We have used recursion to solve the problem and also analysed the time and space complexity of the program.**

Keyword : Subsequence, Recursion, Time complexity, Space complexity.

I. Introduction

In this problem, we have to find all the sorted subsequences, by traversing through each row and each column.

For example, let's suppose the matrix is:
0 2
5 4
Then all the sorted subsequences in the matrix are:
{(2), (0), (4), (5), (0,2)}

We will use recursion to solve the problem.

Recursion is the process in which a function calls itself directly or indirectly and the corresponding function is called as recursive function.
In the recursive program, the solution to the base case is provided and the solution of the bigger problem is expressed in terms of smaller problems.

The algorithm is described in detail in **part II**.

II. Algorithm Description and Analysis

Driver function:

1) We are generating a random matrix in the main and is sent to driver function as input.

2) In driver function we are initialising a vector of int "ls1" as each row one by one is forwarded to find function to calculate the sorted sub sequences.

3) For each row after computation it is stored in 2D vector 'st'.

4) To find column wise, we find store each column as rows in trans matrix and then similarly we compute sorted subsequences for each column in matrix or each row in trans matrix.

5) In this way we compute sorted subsequences for a matrix, row-wise and column-wise.

Find function:

1) In find function, if input row is empty and output row is not empty then, storing the result in 2D vector "st" and return.

2) Else , initalise temp vector and push back value first element of row and erase that value from input row.

3) Call find function and if output row is empty then call it again with output row as temp.

4) Else if first element of temp is greater then last element of output row , then push back that temp value in output row.

5) Again call find function.

III. Pseudo Code

2D vector st is used to store results after computation.find function has two vectors inp and out to store the current input and current output respectively in its prototype as we are using recursion.

```
Function driver :

       For : i -> 0 to R
          initialise vector<int>ls1
          ls1=matrix[i]
          Declare vector<int>ls2
       find(ls1,ls2)
       print st
       clear st

   For: i->0 to C
     For : j -> 0 to R
     transmatrix[i][j] = matrix[j][i];

   For : i -> 0 to C
          initialise vector<int>ls1
          ls1=transmatrix[i]
          Declare vector<int>ls2
       find(ls1,ls2)
```

```
        print st
        clear st

Function find :

    if(inp.empty()&&out.size()!=0)
    st.push_back(out)
    else return

    intialise vector temp
    temp.push_back(inp[0])
    erase the first element in inp
    find(inp, out)

    if(out.size() == 0)
                find(inp, temp)

    else(temp[0] > out[out.size() - 1])
                out.push_back(temp[0])
                find(inp, out)
```

IV. Time Complexity

The time complexity of finding subsequence in each row/column is O(2^n) and thus time complexity of the whole program is O(n·2^n).

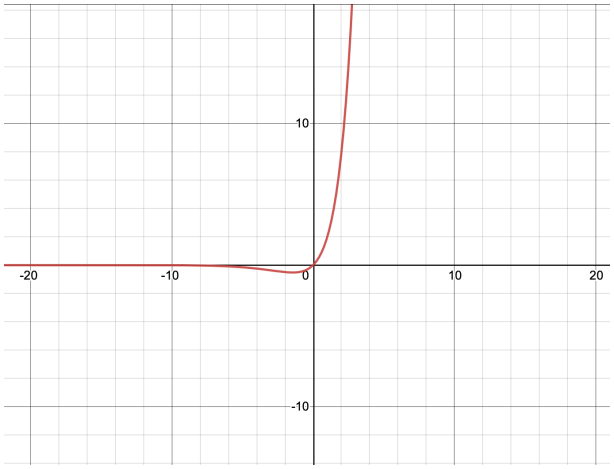

Fig. 1. Time complexity curve

V. Auxiliary Space Complexity

The space complexity of the program is O(n^2). This is because we are using matrices here.

VI. Conclusion

We have arrived at the solution of the problem by recursion, where we are operating on each row and column, thus the time complexity is O(n·2^n).

VII. Applications

Recursion is a wide variety of algorithmic technique which believes on the strategy of calling the other problems and combine them.This methodology has many predefined algorithms which work on the basis of Recursion. Some of the applications of the Recusrion Approach are:

1) **DFS traversal of a graph**

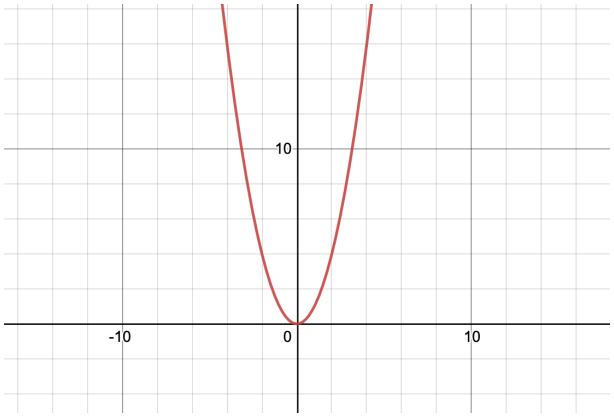

Fig. 2. Space complexity curve

2) **Tower of Hanoi problem**
3) **Tree traversals like inorder,postorder and preorder**
4) **All Backtracking algorithms are based on recursion**

etc many more.

VIII. Acknowledgment

We are very much grateful to our Course instructor Mr.Rahul kala and Dr.Javed and our mentor, Mr.Md Meraz, who have provided the great opportunity to do this wonderful work on the subject of Data Structure and Algorithm Analysis.

IX. References

We have referred [3] to clear the basic concepts of Recursion.Reference [2] helped us , to develop solution of this question.[1] helped us to generate random matrix.

References

[1] https://www.codespeedy.com/generate-a-matrix-of-random-numbers-in-cpp/
[2] https://www.geeksforgeeks.org/number-of-longest-increasing-subsequences/
[3] https://www.inf.unibz.it/ calvanese/teaching/06-07-ip/lecture-notes/uni11.pdf

X. Appendix

A. Project link on Github:

https://github.com/maggi2k19/DaaAssignments/Assignment - 06