To create a matrix of size 50×50 of numbers ranging from 0 to 9.Find the length of the largest sorted component horizontally.

Group Members

Aakash Anand (IIT2019015)

Parth Kataria (IIT2019016)

Shruti Nanda (IIT2019017)

Contents

Introduction

Given a 2d array with n number of rows and m number of columns where n and m is less than 50. The elements of the array will range from 0 to 9. We have to print the length of largest sorted component of the array horizontally. We will discuss different algorithm for this and the complexity of algorithms.

Example:

Input	Output:
3 4 5 6	4
4 3 1 5	
3 4 5 1	

Algorithm Design

We have designed two algorithm for finding the union of two sets of positive Integers.

1. Brute-force

In this algorithm we have checked for each row if the array is increasing or decreasing we have increased the cnt variable and else we have updated the max_cnt variable and make the cnt variable 0.

2. Searching and sorting

In this algorithm we are storing the length of the sorted subarrays of the array in a vector count using nested for loop . Then we are sorting it printing the largest length of the sorted subsequence.

1. Brute force

Algorithm:

Step 1:

Take an array as input and pass it in function.

Step 2:

We will traverse each element in the array horizontally and if the elements are increasing then we will increase the value of count

Step 3 :

If there is one element which is not sorted then we will update our maxcount variable which is storing the maximum length of sorted component horizontally and make count variable 1

Step 4

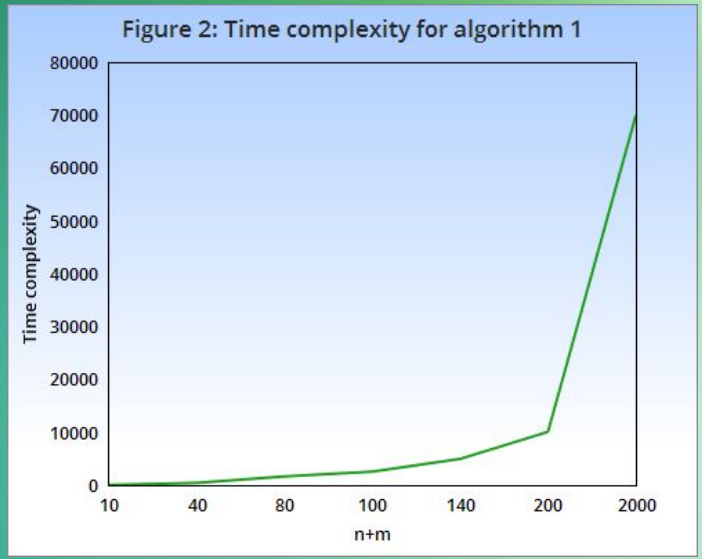
Return max-count

Complexity Analysis

Time Complexity

Best case : Ω(n^2)
Average case : θ(n^2)
Worst case : O(n^2)

Space Complexity : n*m



Pseudo Code for Brute force

```
int Brute(int a[][], int n, int m)
{
   int max_cnt=0,cnt=1;
      for(i =0 to  n){

for( j=1 to j=m)
          {
             if(a[i][j]>a[i][j-1])
                cnt++;
            else{
             max_cnt=max(cnt,max_cnt);
             cnt=1;}
          }
Return max_cnt;
}
```

Ex:
INPUT
3 4
1 2 3 4
3 8 6 5
0 6 7 1

OUTPUT:
4

In this example it will start iterating from 1 and increment the count till 4 now we will store the count value in max_count variable and make count variable 1 and we will do the same for each element and in last we will return max_count.

2. Sorting count

Algorithm

Step 1:

Take an array as input and pass it in function.

Step 2:

We will traverse each element in the array horizontally and if the elements are increasing then we will increase the value of count

Step 3 :

If there is one element which is not sorted then we will push cnt value in count vector which is storing the length of all sorted component horizontally and make count variable 1.

Step 4

Sort the count vector and return last its last element

Complexity analysis

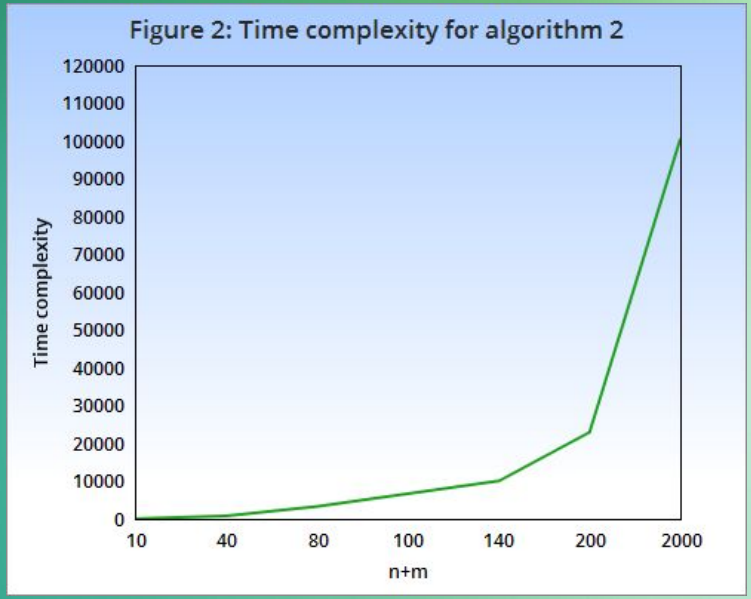
Time complexity

Best case : Ω(n^2)

Average case : θ(n^2.logn)

Worst case : O(n^2logn)

Space Complexity : n*m



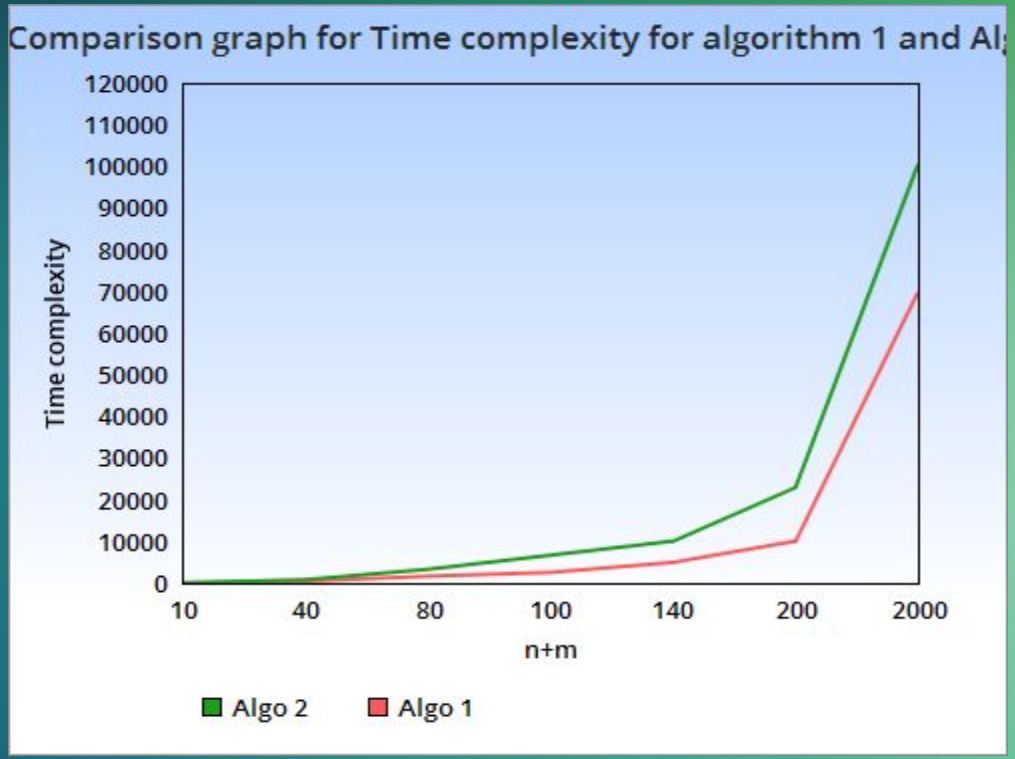
Pseudo code

```
int sorting_cnt(int a[][], int n, int m)
{
   int n, m;
   int max_cnt=0,cnt=1;
   vector<int>count;
   for( i=0 to i=n){
   for( j=1 to m)
   {
     if(a[i][j]>a[i][j-1]){
          cnt++;
          v.push_back(cnt);
    }
   else
          cnt=1;
   }}
   sort(count.begin(), count.end());
          Return count[count.size()-1];
}
```

EX:
INPUT:
3 4
3 4 5 1
1 2 3 4
9 8 7 6

OUTPUT:
4

Comparison of time Complexity of two Algorithm



Conclusion

Time complexity for finding the length of the largest sorted subarray will be minimum in brute force which is O(n^2) and for sorting count algorithm time complexity will be O(n^2logn).

References:

1. https://www.quora.com/How-would-one-use-Arrays-sort-on-a-multidimensional-array-of-ints-by-the-first-element-of-each-sub-array-in-Java
2. https://www.geeksforgeeks.org/longest-increasing-path-matrix/