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

Algorithm design

Time complexity analysis

Conclusion

Reference

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 diffrent 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 three 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

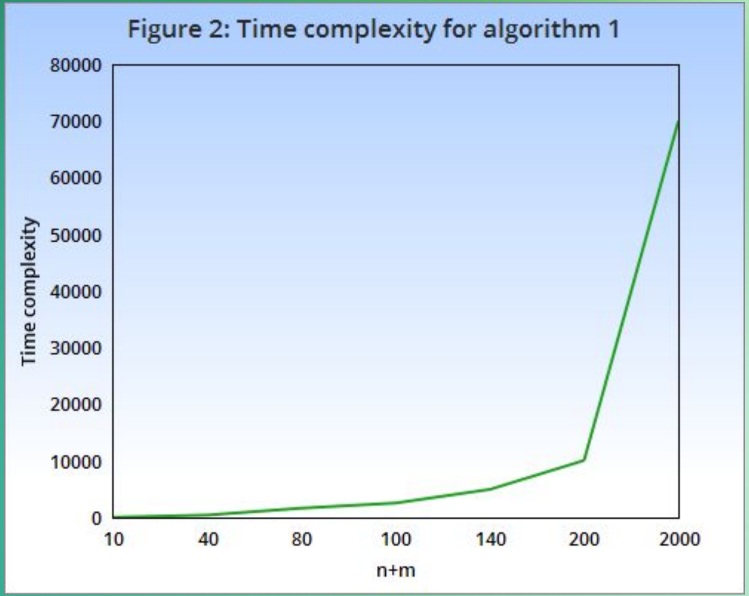
In this algorithm we are storing the length of the sorted subarrays of the array in a vector count using nested for loop . Then we are sorting it printing the largest length of the sorted subsequence.

1. Brute Force

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
9 8 6 5
0 6 7 1
```

OUTPUT:

```
4
```

2. Searching and Sorting

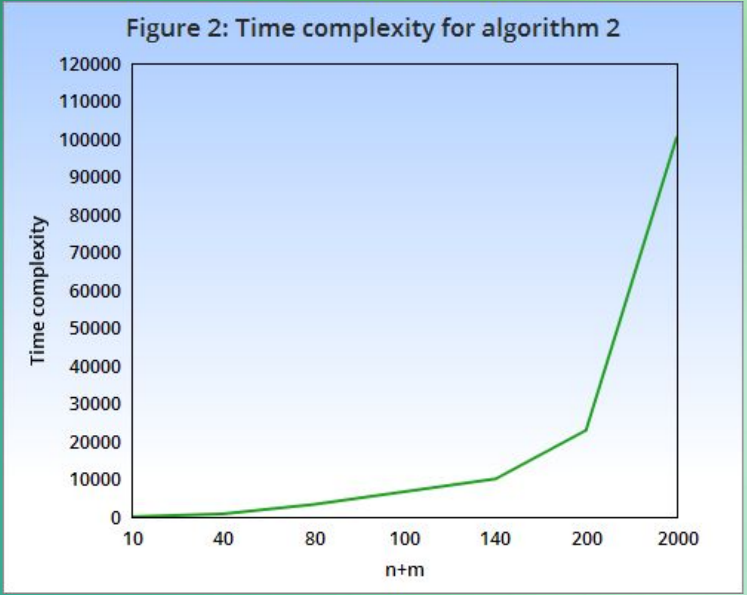
Time complexity

Best case : Ω(n^2)

Average case : θ(n^2.logn)

Worst case : O(n^2logn)

Space Complexity : O(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:

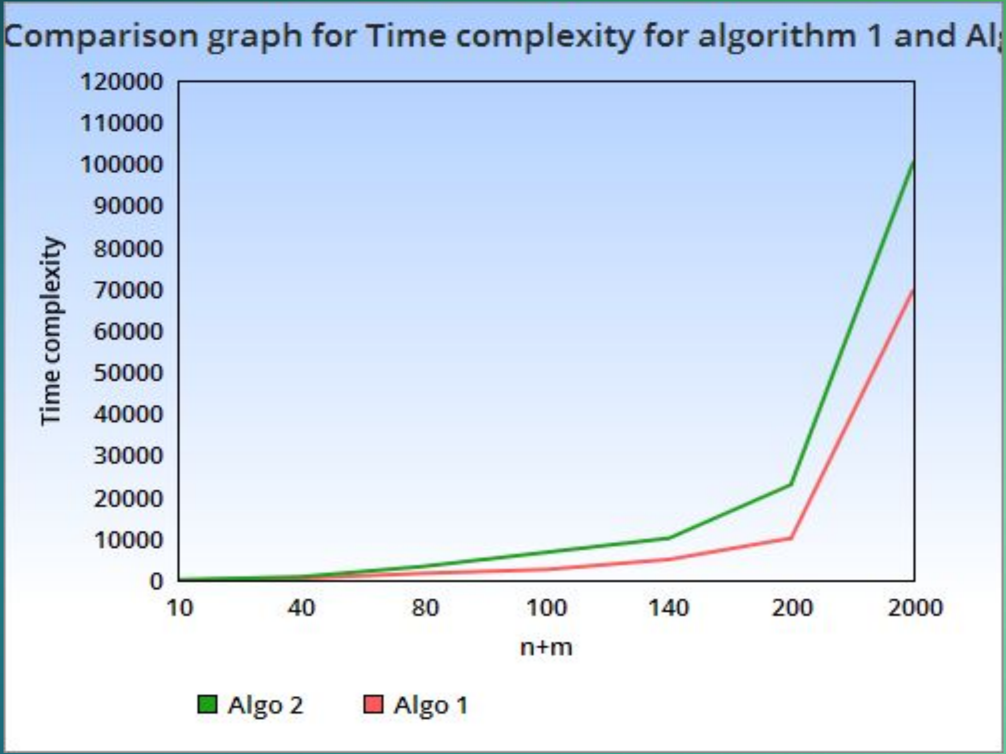
INPUT:

```
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/