Design and Analysis of Algorithms

Assignment No. 2

Group members:

Harsh Meena -IIT2019005

Asha Jyothi Donga -IIT2019006

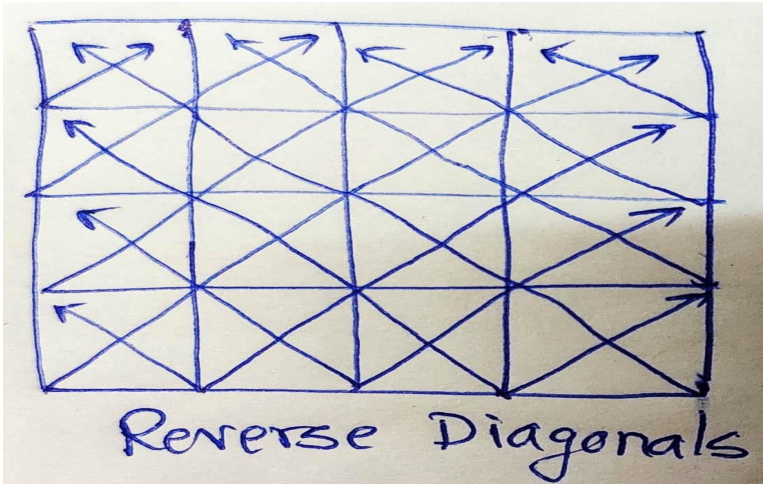
Sampada Kathar -IIT2019007

Contents

Longest Increasing component Reverse Diagonally

Problem:

Create a matrix of size 50×50 of numbers ranging from 0 to 9. Find the length of the largest sorted component reverse diagonally.



Algorithm

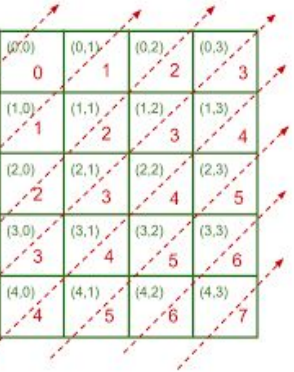
1. Random numbered matrices are created by the function rand().

2. For a given N x N size matrix, the number of Reverse Total Diagonals is 4n-2.

3. For each reverse diagonal:

   Traverse second element from the diagonal.

Diagram-1



if current element is greater than previous element.Then this element helps in building up the previous increasing subarray encountered so far, else check if 'max' length is less than the length of the current increasing subarray.

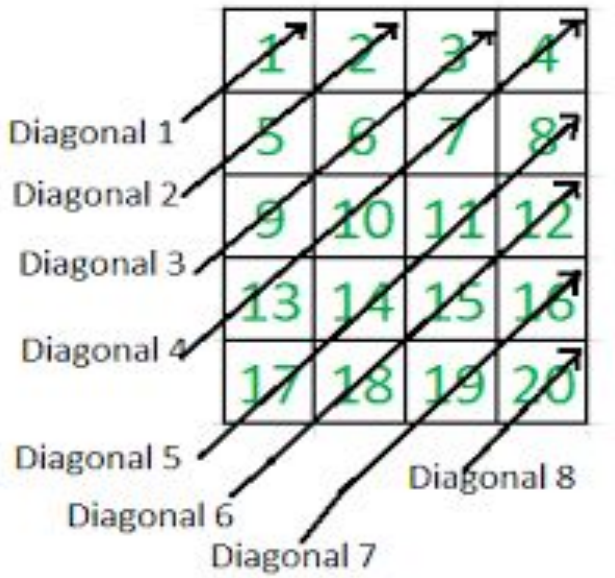
If true, then update 'max'.comparing the length of the last increasing subarray with 'max'.

Pseudo Code :

Int maxlen=1;

Traverse the diagonals of direction from bottom-left to top-right

```
For i=0 to i=n-1
      j=n-2,k=i+1; len=1;
      while(k<=n-1 && j>=0)
        if(v[j+1][k-1]<v[j][k])
              len++
        Else
            maxlen=max(maxlen,len)
        len=1
        k++ j--
   maxlen = max(maxlen,len)
   j=i-1,k=1
   len=1
   While (k<=n-1 && j>=0)
        if(v[j+1][k-1]<v[j][k])
           len++
        Else
           maxlen=max(maxlen,len)
        len=1
        k++ j--
   maxlen = max(maxlen,len)
```



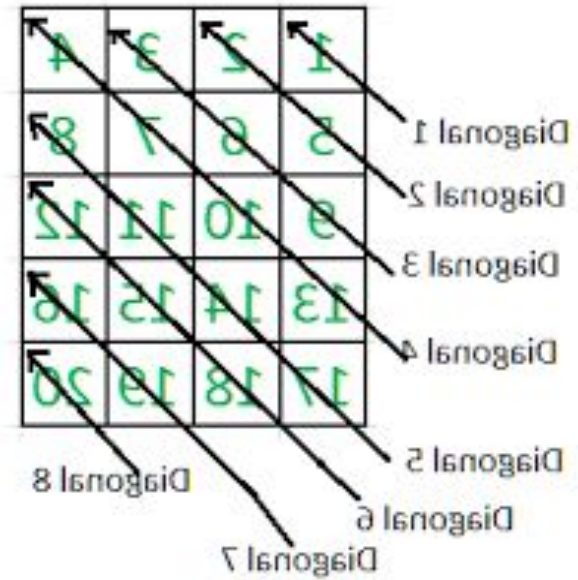
https://media.geeksforgeeks.org/wp-content/uploads/zigzag-1.png

Diagram-2

Traverse the diagonals of direction from bottom-right to top-left

```
for( i=0; i<n;i++)
   j=n-2,k=i-1 len=1
   while(k>=0 && j>=0)
      if(v[j+1][k+1]<v[j][k])
         len++
      Else
        maxlen=max(maxlen,len)
      len=1
      k-- j--
   maxlen=max(maxlen,len)
   j=i-1,k=n-2 len=1
   while(k>=0 && j>=0)
     if(v[j+1][k+1]<v[j][k])
       len++
     Else
       maxlen=max(maxlen,len)
     len=1
     k-- j--
   maxlen=max(maxlen,len);
```

Diagram-3



https://media.geeksforgeeks.org/wp-content/uploads/zigzag-1.png

Time Complexity and Space Complexity Analysis

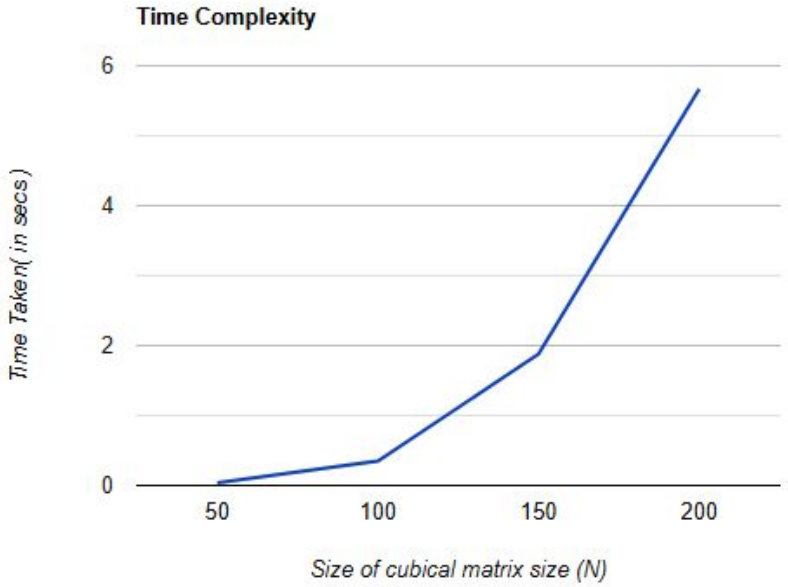
Time Complexity

For travelling to the starting cells of reverse Diagonal we need TC of order N

And then for travelling through these diagonals and finding the largest sorted component it takes TC of order of N

TC becomes : O(N^2)

- For best case: n=1 , hence Ω(1)
- Worst Case TC O(N^2)



Experimental Graph 1

Space Complexity

We haven't occupied any extra space in this algorithm

Hence SC is O(1)

Conclusion

From the analysis of our algorithm we can conclude that the algorithm devised by us is efficient to solve this problem.

It takes no extra space which makes it efficient memory wise, and with a worst case complexity of only O(n^2), it is efficient enough to process a 50*50 matrix which is what was required according to our problem statement.

References:

1. https://www.geeksforgeeks.org/longest-increasing-subarray/
2. https://www.geeksforgeeks.org/maximum-length-of-strictly-increasing-sub-array-after-removing-at-most-one-element/
3. https://stackoverflow.com/questions/14492047/longest-subarray-that-has-elements-inincreasing-order