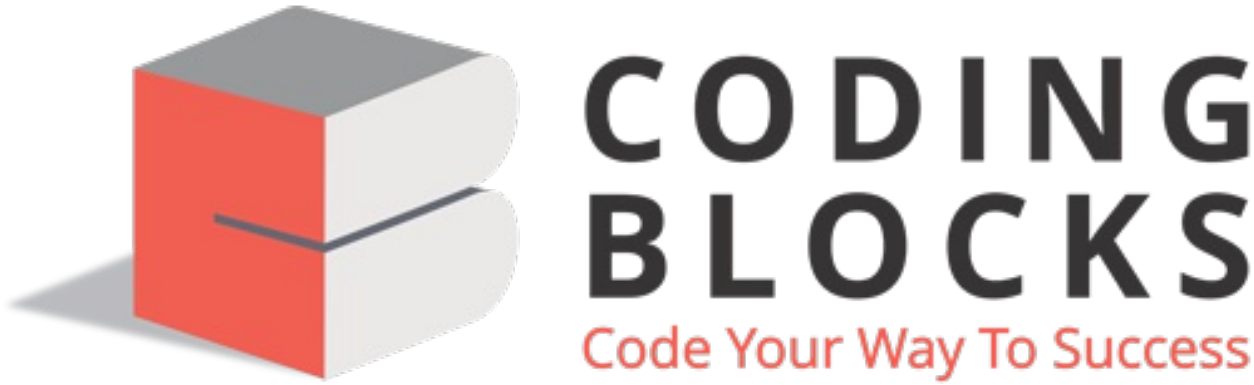

Problem Name : Maximum length Bitonic Subarray

##Approach
We create two arrays - 'inc' and 'dec'

1. inc[i] stores the length of increasing subarray till i.
2. dec[i] stores the length of decreasing subarray starting from index i.
3. Doing so gives us the length of increasing and decreasing subarray at each index in O(n) time.
4. We calculate the length of the longest bitonic subarray by finding the maximum inc[i] + dec[i] - 1
5. We subtract one since the current element at ith index is included in both the increasing and decreasing subarray lengths.
   ##Algorithm
6. Initialize inc[0] to 1 and dec[n-1] to 1
7. Creating inc[] array
   a. Till end of the array ie, i=1 to n, if arr[i] > arr[i-1] then inc[i] = inc[i-1] + 1.
   else, inc[i] = 1
8. Creating dec[] array
   a. From the end of the array ie, i = n-2 till i =0, if arr[i] > arr[i+1] then dec[i] = dec[i+1] +1
   else, dec[i] = 1
   ####Java Code

```java
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner scn = new Scanner(System.in);
        int t=scn.nextInt();
        while(t-- > 0) {
        int n = scn.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < arr.length; i++) {
            arr[i] = scn.nextInt();
        }
        System.out.println(bitonic(arr));
        }
    }

    public static int bitonic(int arr[])
    {
        int n=arr.length;
        int[] inc = new int[n];

        int[] dec = new int[n];
        int max;

        inc[0] = 1;
        dec[n-1] = 1;
        for (int i = 1; i < n; i++)
           inc[i] = (arr[i] >= arr[i-1])? inc[i-1] + 1: 1;
        for (int i = n-2; i >= 0; i--)
            dec[i] = (arr[i] >= arr[i+1])? dec[i+1] + 1: 1;
        max = inc[0] + dec[0] - 1;
        for (int i = 1; i < n; i++)
            if (inc[i] + dec[i] - 1 > max)
                max = inc[i] + dec[i] - 1;

        return max;
    }

}
```

####C++ Code

```c
int bitonic(int arr[], int n)
{
    int inc[n]; // Length of increasing subarray ending at all indexes
    int dec[n]; // Length of decreasing subarray starting at all indexes
    int i, max;

    // length of increasing sequence ending at first index is 1
    inc[0] = 1;

    // length of increasing sequence starting at first index is 1
    dec[n-1] = 1;

    // Step 1) Construct increasing sequence array
    for (i = 1; i < n; i++)
    inc[i] = (arr[i] >= arr[i-1])? inc[i-1] + 1: 1;

    // Step 2) Construct decreasing sequence array
    for (i = n-2; i >= 0; i--)
    dec[i] = (arr[i] >= arr[i+1])? dec[i+1] + 1: 1;

    // Step 3) Find the length of maximum length bitonic sequence
    max = inc[0] + dec[0] - 1;
    for (i = 1; i < n; i++)
        if (inc[i] + dec[i] - 1 > max)
            max = inc[i] + dec[i] - 1;

    return max;
}
```