Maximum Consecutive ones

BRUTE FORCE :

Generate all subarrays invert 0 to 1 and when left with no more inversions and a zero occurs then break

```
int longestSubSeg(vector<int> &arr , int n, int k){
    // Write your code here.
    int maxi=-1,ans,t;
    for(int i=0;i<n;i++)
    {
        t=k;
        if(arr[i]==1)
            ans=1;
        else
        {
            if(t>0)
            {
                t--;
                ans=1;
            }
            else
                ans=0;
        }
            for(int j=i+1;j<n;j++)
            {
                if (t==0 && arr[j] == 0)
                {
                    break;
                }
                else if(t>0 && arr[j]==0)
                {
                    t--;
                }

                ans++;
                maxi=max(maxi,ans);
            }


    }
    return maxi;
}
```

- Time Complexity : O(N^2)

- Space Complexity : O(1)

Optimal Approach :

Using two pointers left and right moving right counter until count zero is <k and incrementing count zero. As soon as count zero exceeds k we slide the window by incrementing left if at left pointer a zero is encountered we decrement cnt.

We maintain a maxlen variable which stores the maxlen of consecutive ones in a valid window.

```
int longestSubSeg(vector<int> &arr , int n, int k){
    // Write your code here.
    int l=0,r=0,cnt0=0,maxlen=0;
    for(int r=0;r<n;r++)
    {
        if(arr[r]==0)
        {
            cnt0++;
        }
        while(cnt0>k)
        {
            if(arr[l]==0)
                cnt0--;
            l++;
        }
        maxlen=max(maxlen,r-l+1);

    }
    return maxlen;
}
```

- TimeComplexity : O(N)
- Space Complexity : O(1)