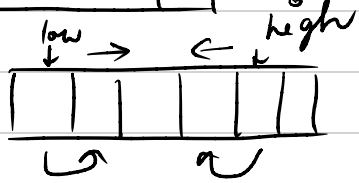
August 10, 2025

TWO POINTERS FORM 1

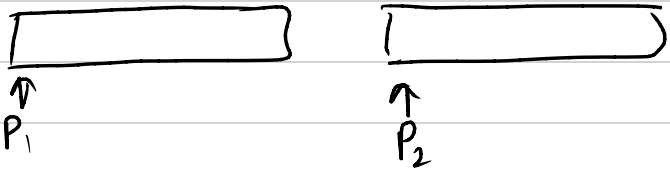
Form 0 : Sliding Window (Ex : Monotone Deque ques in STL)

Form 1 : Variable Window

Form 2 : 3Sum Type (opposite side)



Form 3 : Multi-Sequence



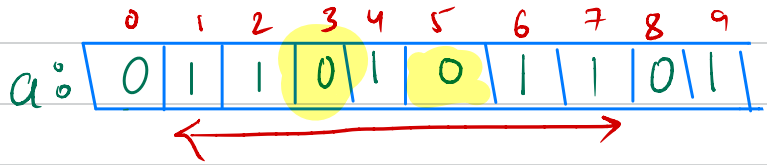
* Form 1, what kind of problems can we solved using form 1?

For subarrays :
- Finding max length
- Finding min length
- count of subarray
- Sum of length

→ (given some condition that needs to be satisfied)

Ques 1. Given an array of 0's and 1's. Find the max length of 1's that can be created by flipping at most K positions with 0's.

N=10, K=2



For K=2, that means we can flip at most 2 zeroes in one array.

Answer = 7

We will be using form 1 to solve this question. For that, first we have to find the condition that should be satisfied at all times:

Find max length of subarray with <=K 0's → condition

Step 2:

If a condition is true for l to r then it must hold true for l to r-1 as well.

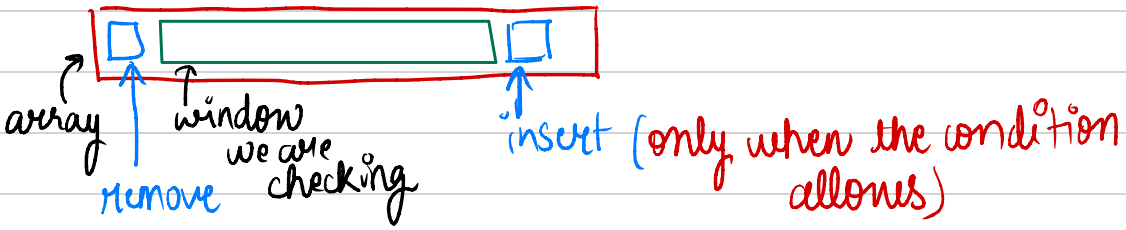
Here,

[L, R] satisfy
[L, R-1] Satisfy

Step 3:

Design a data structure which can check for the identifiable condition.

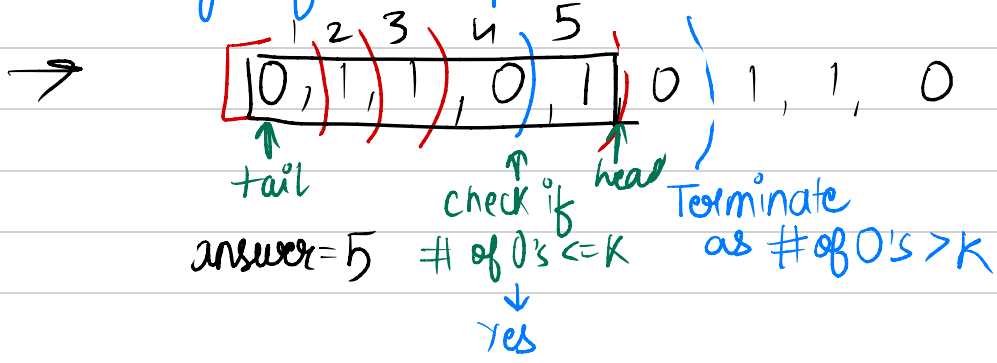
In this case, our check fxn will check that if we take an element in our window, then will the condition still remain true?



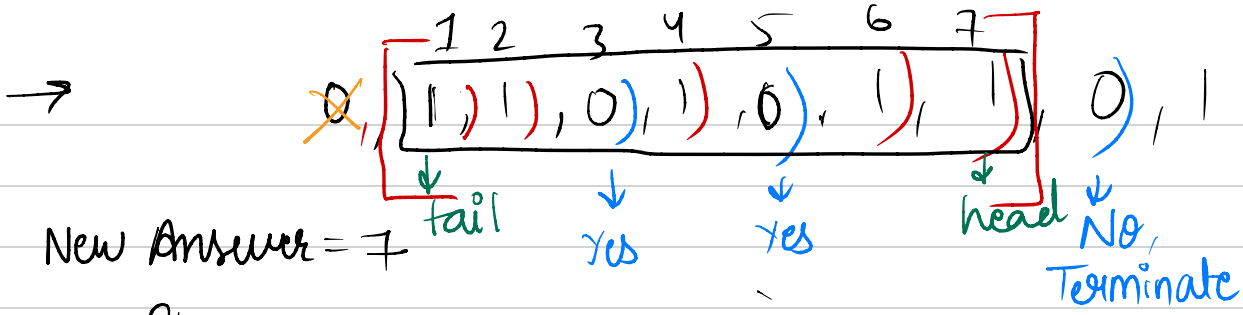
Can we take the next element or not? If it is 1, then we can continue to take it but if its 0 then we shall see if this incoming 0 is <=K, if it is then we can continue otherwise return.

Our check fxn is essentially counting no. of 0's in a window.

- Runthrough for example



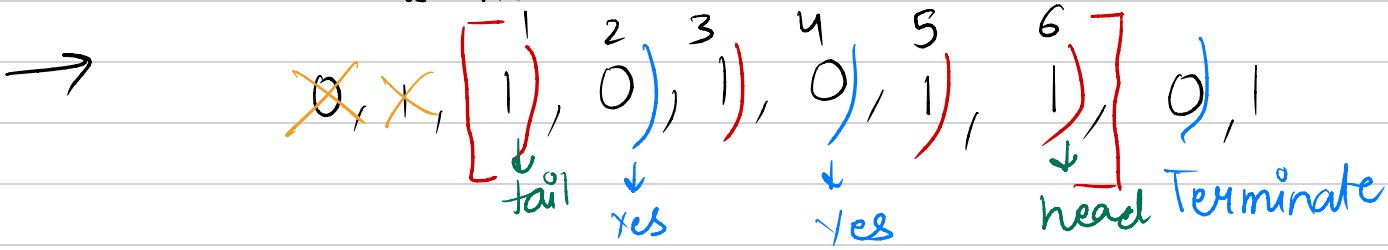
Red parantheses indicates that it can be taken.



New Answer = 7

Since new answer > answer

ans = new answer



New answer = 6

6 < 7, So answer stays same.

Similarly the window slides through using the head & tail pointer and the best answer is returned

```
// tp
int ans = 0;
int head=-1,tail=0;
while(tail<n){ //O(N) Amortized
    while(head+1<n && (arr[head+1]==1 && cnt0<=k)||
                      (arr[head+1]==0 && cnt0<k)){
        head++;
        // insert head.
        if(arr[head]==0)cnt0++;
    }
    // update answet
    ans = max(ans, head-tail+1); O(1)

    // remove element from tail
    if(tail<=head){        //O(1)
        // remove from DS.
        if(arr[tail]==0)cnt0--;
        tail++;
    }else{
        tail++;
        head = tail-1;
    }
}
```

Ques 2. Find count of number of subarrays with # of distinct elements $\leq K$.

Just like last question, here our check would be:
If we take next element, will the # of distinct element will remain $<= K$.

1/2

```
// ds
int distinct = 0;
int freq[1000100];<- used to keep count of distinct

void insert(int x){
    if(freq[x]==0)distinct++;
    freq[x]++;
}

int check(int x){
    int cnt = distinct;
    if(freq[x]==0)cnt++;
    return cnt;
}

int erase(int x){
    freq[x]--;
    if(freq[x]==0)distinct--;
}
```

2/2

```
while(tail<n){
    while(head+1<n && check(arr[head+1])<=k){
        head++;
        // insert head.
        insert(arr[head]);
    }
    // update answer
    ans += head-tail+1;

    // remove element from tail
    if(tail<=head){
        // remove from DS.
        erase(arr[tail]);
        tail++;
    }else{
        tail++;
        head = tail-1;
    }
}

cout<<ans<<endl;
```