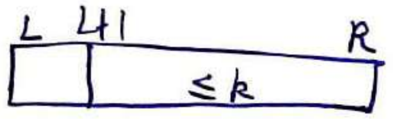
2. Find the count of # of subarray with # of distinct ele $\leq k$

$\boxed{N \leq 10^6 \quad arr[i] \leq 10^6}$

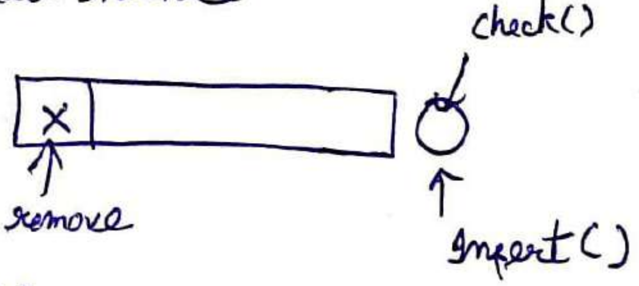
Step-1 : Identify if can be solved using 2-pointer or is 2-pointer be used here?

Step-2 :



Step 3 : Design data structure

Main check.



Use map.

Two pointers

Form 1:

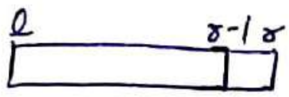
① N = 10, K = 2
Arr: [0 1 1 0 1 0 1 0 1 1 0 1]

Max length of 1's you can create
↳ flip almost K posⁿ.

Find max length subarray with $\leq k$ 0's
↳ condⁿ

Step-1: Identify the problem with the key words.

Step-2:

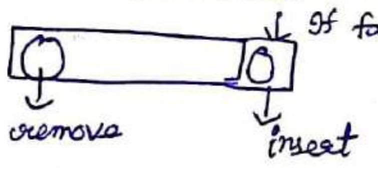


$[l, r] \to$ satisfy
⇓
$[l, r-1] \to$ satisfy

} Then we can apply 2 pointers

Is the condⁿ being asked is satisfied by 2-pointer approach?

Step-3: cr: $\boxed{<K \text{ ' } 0's}$ ←



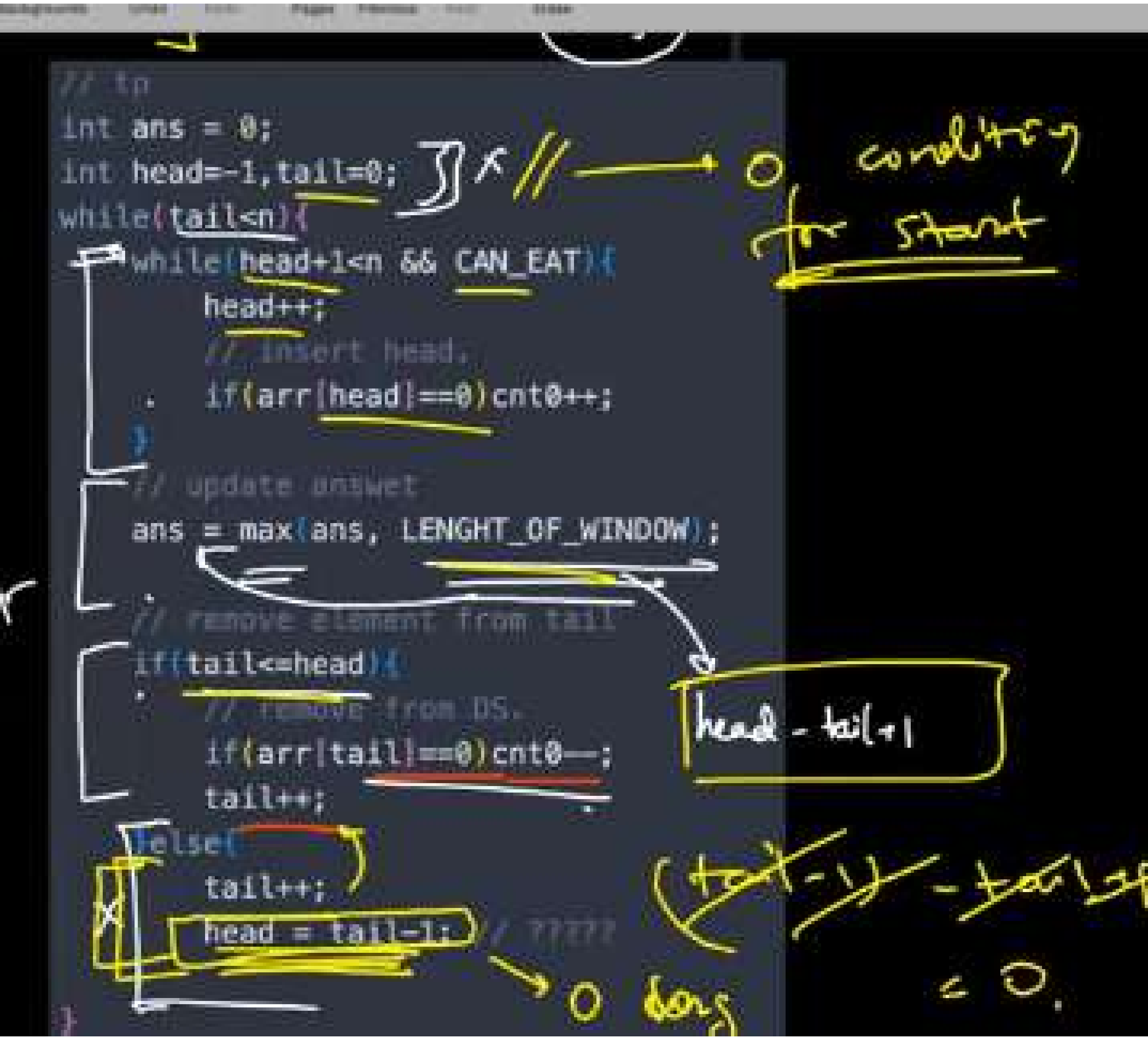
Design data structure → # 0's in window

Two pointers

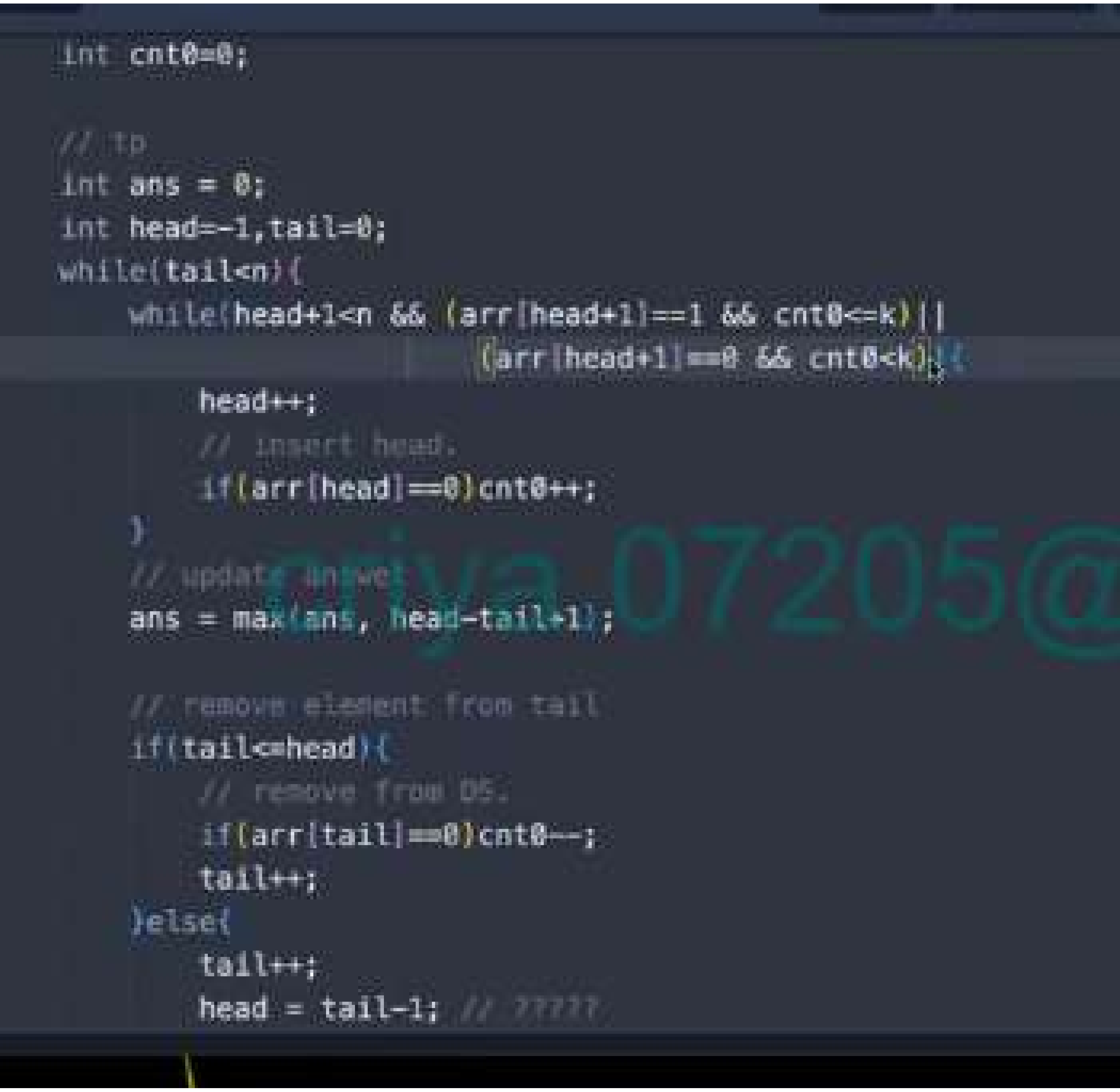
- → Form 0 : Sliding Window
- → Form 1 : Variable Window
- → Form 2 : 3 sum Type (opp. side)
- → Form 3 : Multi - sequence

Form 1
- → Max length subarray
- → Min length
- → count of subarray
- → sum of length of subarray

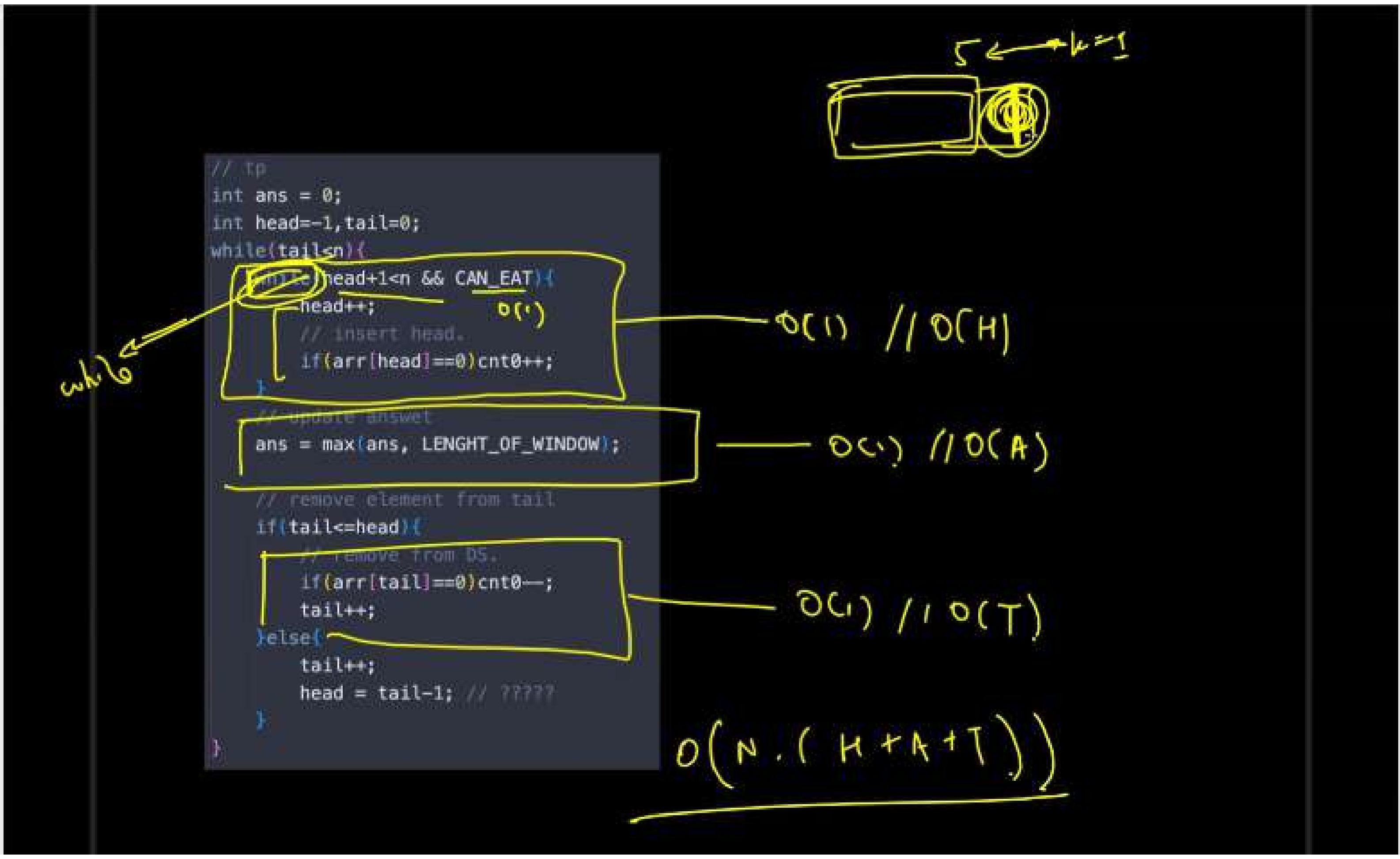
} → Satisfy some cond^n



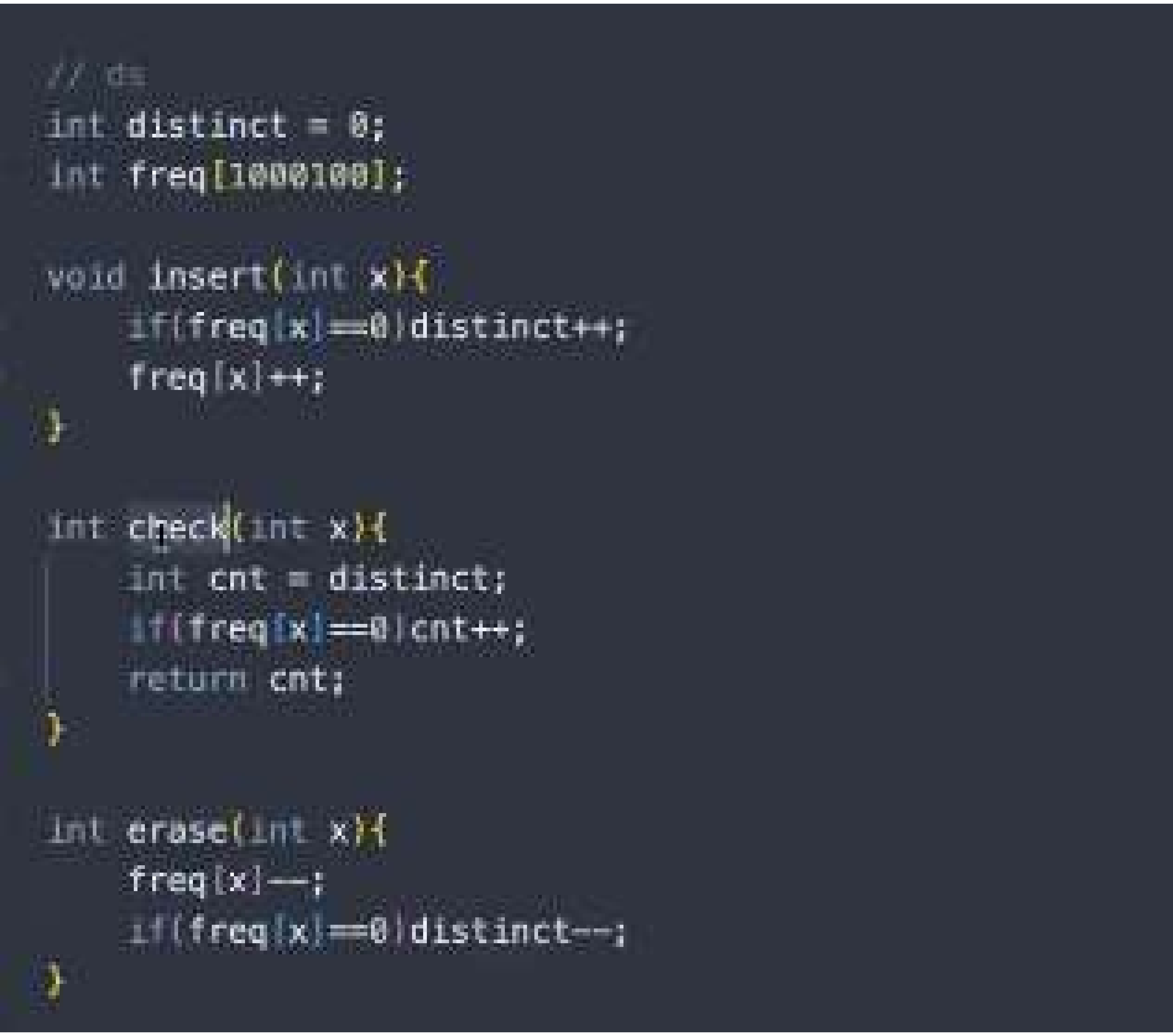
```
// tp
int ans = 0;
int head=-1,tail=0;
while(tail<n){
    while(head+1<n && CAN_EAT){
        head++;
        // insert head.
        if(arr[head]==0)cnt0++;
    }
    // update answer
    ans = max(ans, LENGHT_OF_WINDOW);
    // remove element from tail
    if(tail<=head){
        // remove from DS.
        if(arr[tail]==0)cnt0--;
        tail++;
    }else{
        tail++;
        head = tail-1; // ?????
    }
}
```



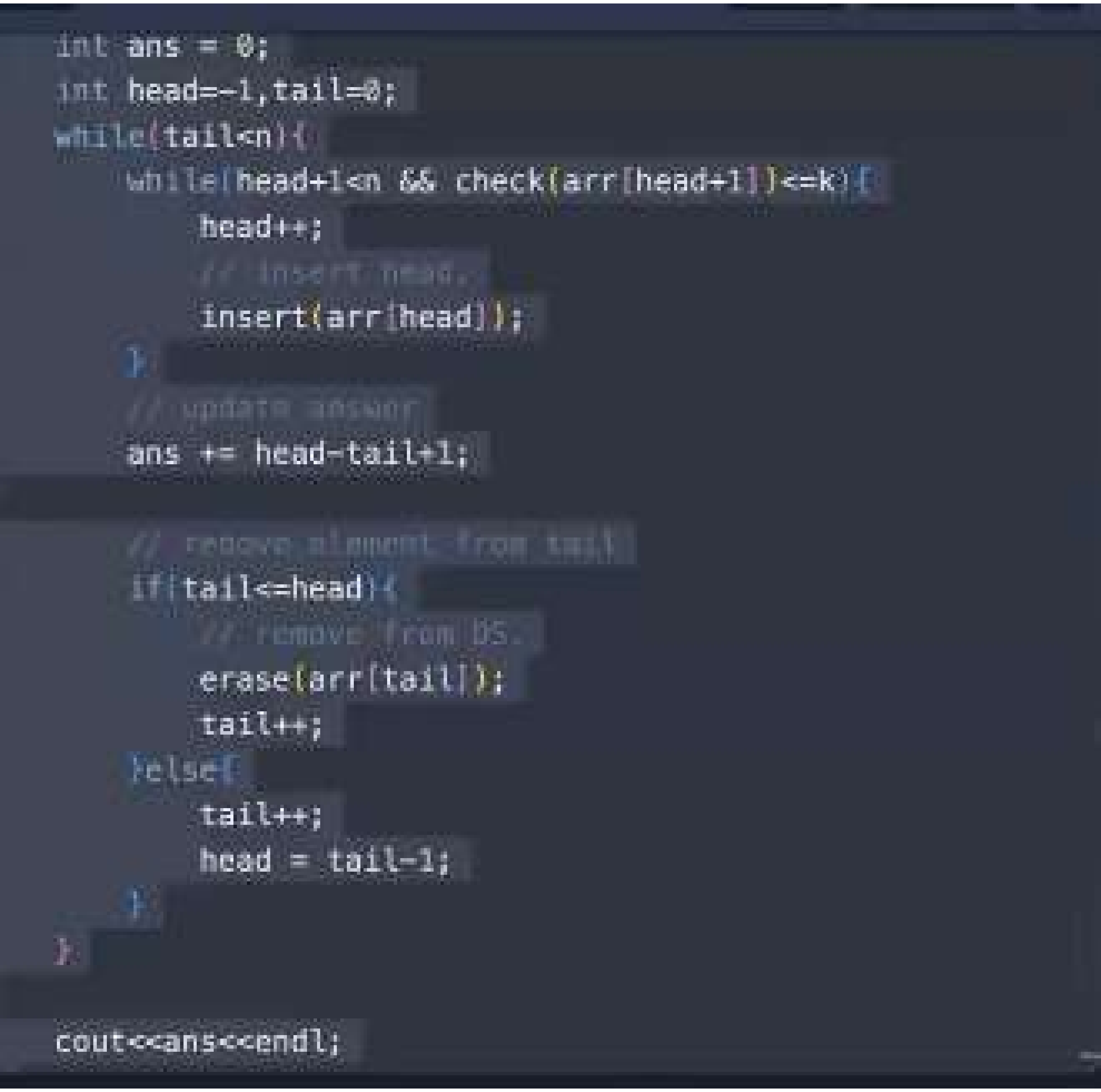
```
int cnt0=0;

// tp
int ans = 0;
int head=-1,tail=0;
while(tail<n){
    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 answe
    ans = max(ans, head-tail+1);

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



```
// tp
int ans = 0;
int head=-1,tail=0;
while(tail<n){
    while(head+1<n && CAN_EAT){
        head++;
        // insert head.
        if(arr[head]==0)cnt0++;
    }
    // update answer
    ans = max(ans, LENGHT_OF_WINDOW);

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



```
// ds
int distinct = 0;
int freq[1000100];

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--;
}
```



```
int ans = 0;
int head=-1,tail=0;
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;
```