Two Pointers

1. Finding longest subarray with $\leq K$ zero.

soln:

Binary search → BS on Answer (90%.)
→ BS on every start

N
↳ Arr:

T	T	T	T	T	F	F	F	F
1	0	0	1	0	1	0	1	1	0

↑ st

$\leq$ 3 zeros.

Now,

	0	1	2	3	4	5	6	7	8	9
Arr[]:	1	0	1	0	0	1	0	1	0	1

L L' R R'

for K = 3

R: farthest ending point for start = L.

Claim

$L \rightarrow R$

$l < L'$ } we can use two pointers.

then $R \leq R'$ }

Binary search $(n \log n)$ → Two pointers (n)

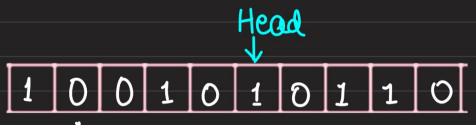
Head and Tail

K = 3

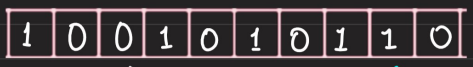
head = -1, tail = 0 } ⇒ subarray is empty.

```
while (tail < n)
{
   //push the head as far as possible
   // process current window
   // tail++, update accordingly
}
```

tail → ans for this tail = 6



tail → ans = 5



tail

Heal

↳ ans = 7

for every tail, we can get the answer

↓

take max over all the answers.

⋮

so on.

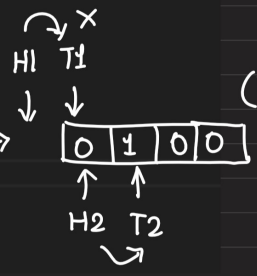
Finally ans = 7.

```cpp
lli n,k;cin>>n>>k;
vector<lli> arr(n);
for(int i=0;i<n;i++) cin>>arr[i];

int cnt0s = 0;
int head = -1,tail = 0;
int ans = 0;
while(tail<n)
{
    while(head+1<n && (arr[head+1]==1 || (arr[head+1]==0 && cnt0s<k)))
    {
        head++; //head shifted to next element
        if(arr[head]==0){
            cnt0s++;
        }
    }
    //Update
    ans = max(ans,head-tail+1);

    //Move tail ahead
    if(head>=tail){
        if(arr[tail]==0) cnt0s--;
        tail++;
    }else{
        tail++;
        head = tail-1;
    }
}
cout<<ans<<'\n';
```

Why this →



(K = 0) } edge case

Printing head, tail for each iteration is the best way to debug.

Time Complexity :

1) push the head as far as possible → ——— O(H)

2) process current window → ——— O(A)

3) tail ++, update accordin → ——— O(T)

T T+1

Time complexity ≡ O(N * (H + A + T))

Here,

$\sum_{\forall N} H = N \Rightarrow O(H) = 1$ (armotized) , $O(A) = O(T) = 1$

∴ TC ≡ O(N)

Types of Problems :

- Count # of subarrays.
- Max len / Min len subarray.

Q2. N, arr | 1 | 2 | 3 | 2 | 3 | 4 | 1 | 5 | 1 | 8 | K = 2

$N \leq 10^5$
$Arr[i] \leq 10^9$

Find #Subarrays with #distinct elements $\leq K$.

eg: 1 1 1 distinct

| 1 | 2 | 1 |

2
2
2

#

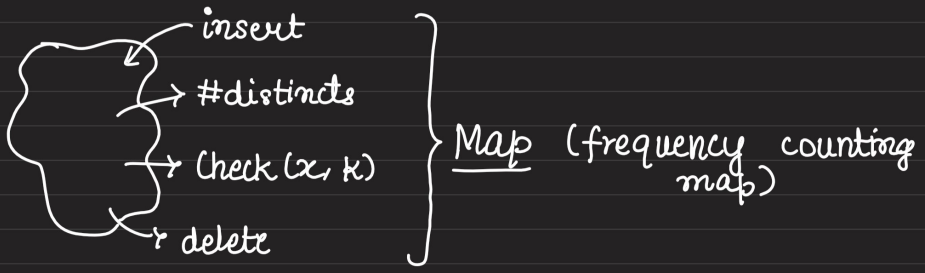
| 1 | 2 | 3 | 2 | 3 | 4 | 1 | 5 | 1 | 8 |

L R K = 2

tail Head

Tail (L') Head (R')

$L < L' \Rightarrow R' \leq R$ } Two pointers.



3 Sum

A: 

Choose i, j, k st $A[i] + A[j] + A[k] = T$ (target)

- $0 \le i < j < k < N$
- $N \le 5 \times 10^3$

Soln: If $N \le 500$

```
for (i → (0, n-1))
    for (j → (i+1, n-1))
        for (k → (j+1, n-1)) {
            if (A[i] + A[j] + A[k] == T)
                cnt++;
        }
```

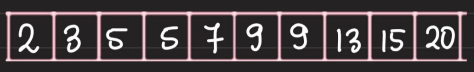
Naive approach

Variants

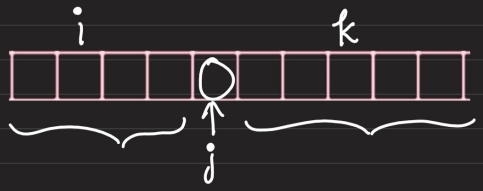
- → Find & print one of the soln
- → Find if a solution
- ★ → Find #solutions.

Step (1) : Sort (A, A+n)

Now

A: 

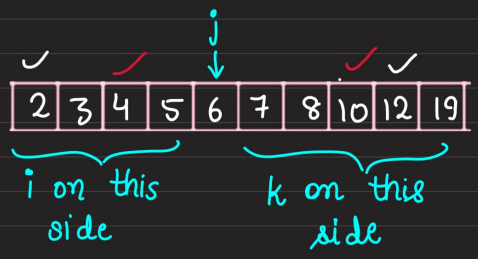
T = 21
(5, 7, 9)

A: 

Find #ways to choose i, k st.
$A[i] + A[k] = T - A[j]$

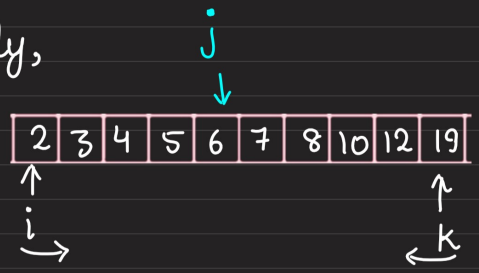
In O(N), we can reduced 3 sum → 2 sum.

eg:



T = 20

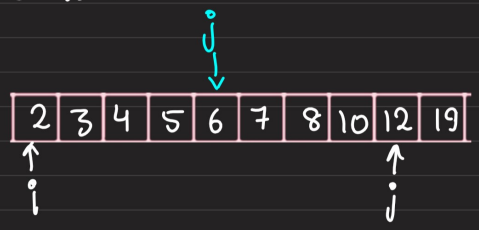
Initially,



A[i] + A[j] + A[k] = 27
beyond target.

If i increses sum↑,
If k decreses sum↓

∴ Move k.

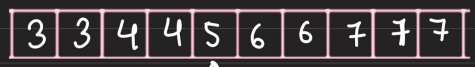


sum = 20 ★
psbl = 1.

for (j → (0, n-1))
↳ (———) (N)

O(N²)

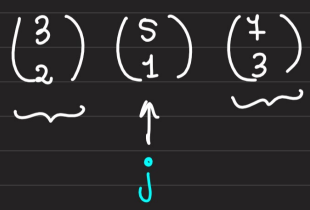
Counting Answers :



T = 15

let's take a simpler case

Ans = 6



Ans = 2 * 3 * 1
= 6.