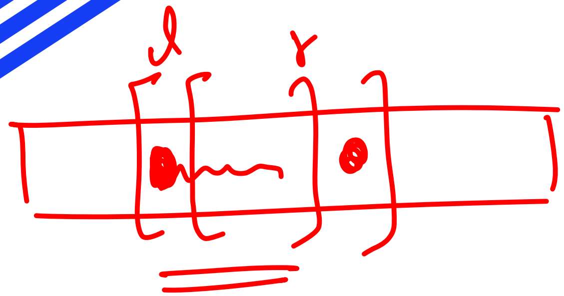



$(l, r)$

$\rightarrow (l+1, r+1)$

sum -= arr[l]

# Two Pointers

sum += arr[r+1]

fixed size sliding window technique

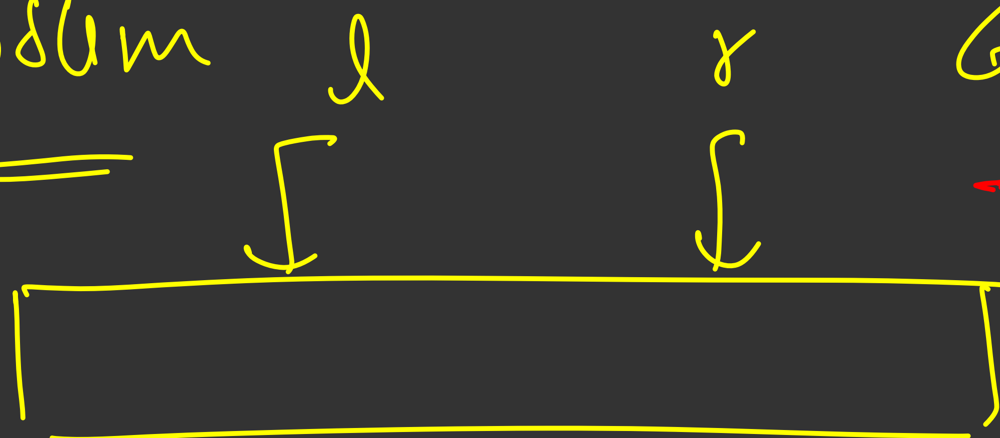
variable size sliding window

- Priyansh Agarwal

Two pointers

αα

Ad hoc problem




Good segment

# Two Pointers

- Widely used in Competitive Programming
- Optimization Technique
- Most Two Pointer problems can be solved using Binary Search
- Useful for a lot of array based problems
- Super useful for interviews too

Optimization over a lot of B.S ideas

$O(n\log n) \longrightarrow O(n)$

Given 2 sorted arrays, for each element in 1st array find number of elements smaller than that in the 2nd array

$O(n)$ $\qquad$ $O(m)$

$n+m = 2\cdot 10^5$

$n\log m = 20\cdot 10^5$

| | | | |
|---|---|---|---|
| 1 | 4 | 5 | 9 |
| 0 | 2 | 2 | 3 |

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 10 |

$O(n\log m)$

First Approach: Binary Search for each elements

$\downarrow$

$O(n+m)$

$1 \le n, m \le 10^5$

$0 \le a[i], b[i] \le 10^9$

A

B

$a_i \le a_{i+1}$

$(0 \le i \le n-2)$

$b_i \le b_{i+1}$

$(0 \le i \le m \; 2)$

No of elements smaller than $a_i = 5$

No. of " " than $a_{i+1}$

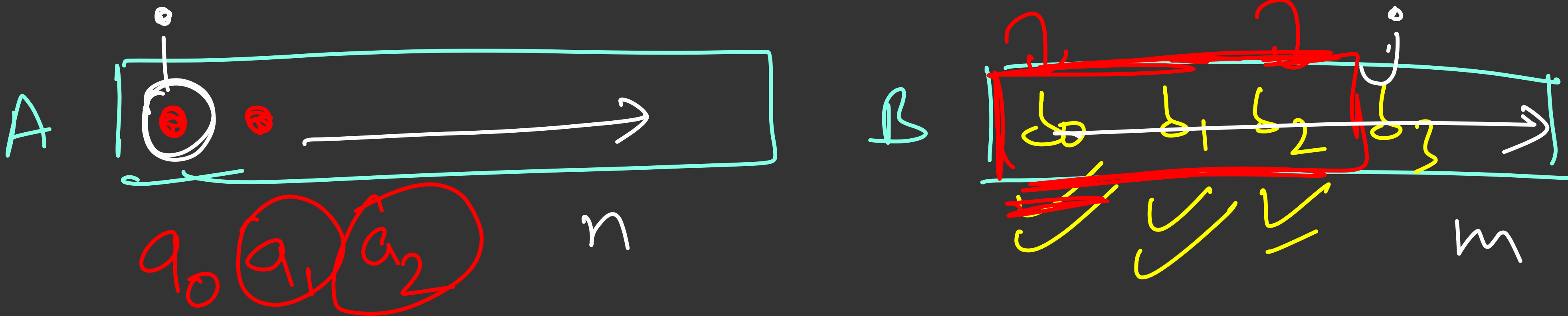
1. $< 5$
2. $\geq 5$

$$b_0, \quad b_1, \quad b_2, \quad b_3, \quad b_4, \quad < \quad a_i$$

$$\| \quad \| \quad \| \quad \| \quad \| \quad < \quad a_{i+1}$$

$$a_{i+1} \geq a_i$$

$$\boxed{b_5 \geq a_i}$$




$a_i \quad a_{i+1}$

$b_0 < a_0$ $\qquad$ $b_2 < a_1$

$b_1 < a_0$ $\qquad$ $b_3 \geq a_1$

$b_2 \geq a_0$

2 $\quad$ 3

$a_2 \geq a_1$ $\qquad$ $b_j < a_i$

$O(n+m)$

## Given 2 sorted arrays, for each element in 1st array find number of elements smaller than that in the 2nd array

| | | | |
|---|---|---|---|
| 1 | 4 | 5 | 9 |

| | | | |
|---|---|---|---|
| ~~3~~ 2 | ~~2~~ 3 | 6 | 10 |

Second Approach: 2 pointers

## Solution using 2 pointers

amortized time complexity

If 5 elements are smaller than a[i], how many elements will be lesser than a[i + 1]?

$\boxed{\geq 5}$

Clearly, we should check for elements bigger than first 5 elements now as a[i + 1] >= a[i]

Having 2 pointers and both only move right. Time complexity?

O(n+m)

```
vector<int> a(n), b(m);
vector<int> ans(n);
int i = 0, j = 0;
while(i < n){
    while(j < m && b[j] < a[i]){
        j++;
    }
    ans[i] = j;
    i++;
}
```

# Good Segments Technique (Increasing)

$1 \le n \le 10^5$

$1 \le a(i) \le 10^9$

- Problem 1: Given an array of positive integers find the length of longest subarray with sum <= K

$1 \le k \le 10^9$

$\underline{\underline{k}}$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 10 | 2 | 3 | 4 | 1 | 1 | 2 | 1 | 5 |

$\underline{\underline{k = 9}}$

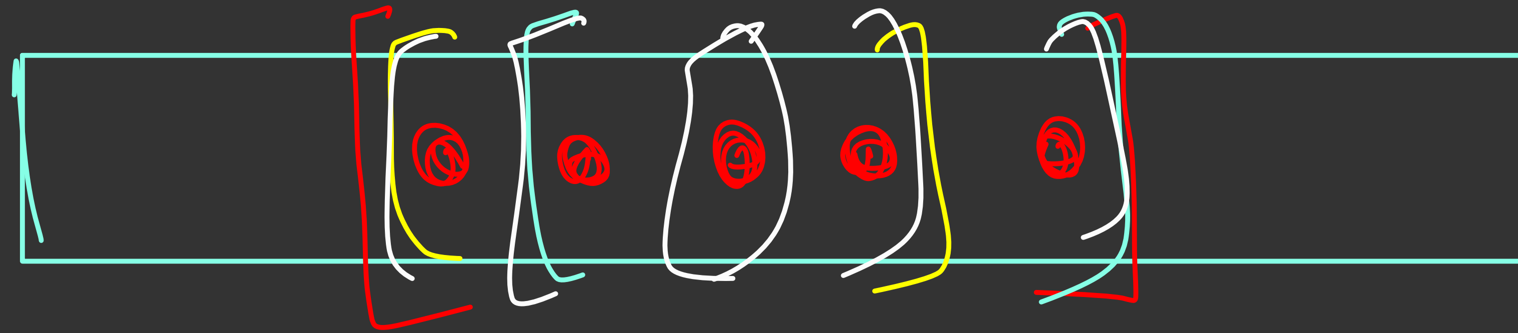
Requirement : sum of subarray must be $\leq k$

we know that elements are +ve

if there exists a subarray of length $\underline{\underline{5}}$ whose sum $\leq k$

Can I say that there will also exist a subarray of length $\underline{\underline{4}}$ whose sum $\leq k$

for length $\underline{\underline{3}}$, $\underline{\underline{2}}$, $\underline{\underline{1}}$

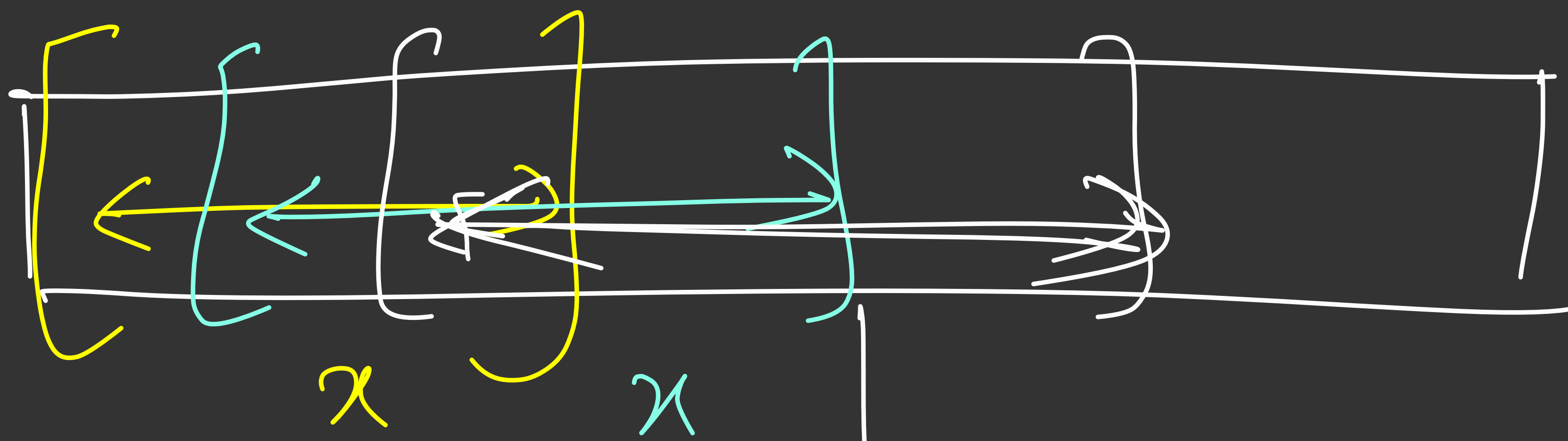


$\leq k$

$$\boxed{f(x)} = \begin{cases} \text{True if there exists a subarray of length } x \text{ whose sum} \leq k \\ \text{false o/w} \end{cases}$$

T T T T T $\boxed{T}$ F F F F F F F F

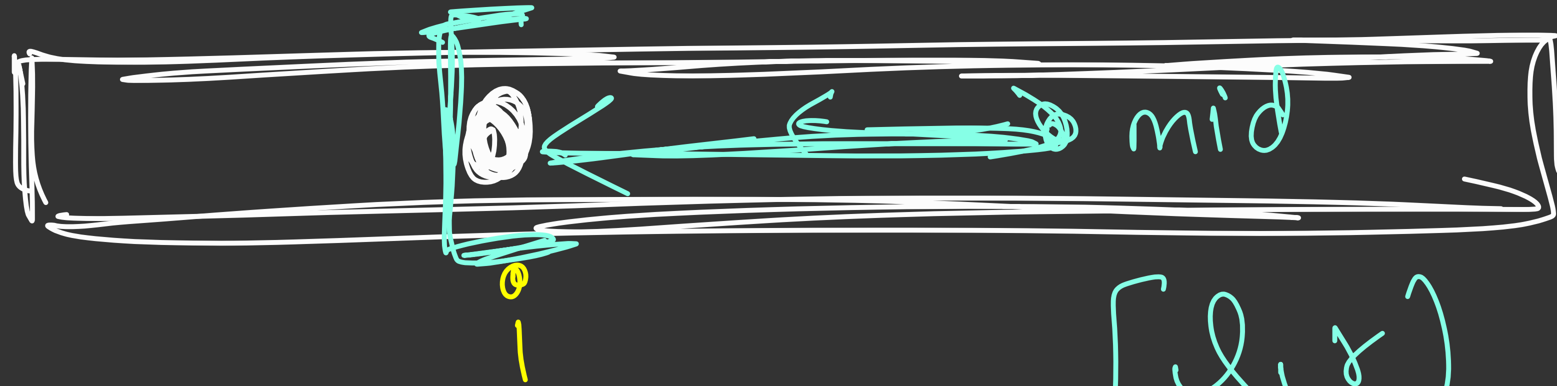
To implement $f(x)$




$f(x) = \text{True}$

$O(n)$

$O(\log n \cdot O(n))$
$= O(n \log n)$




$[l, r)$

find out the length of the longest subarray which starts at $i$

sum $\leq k$

logn

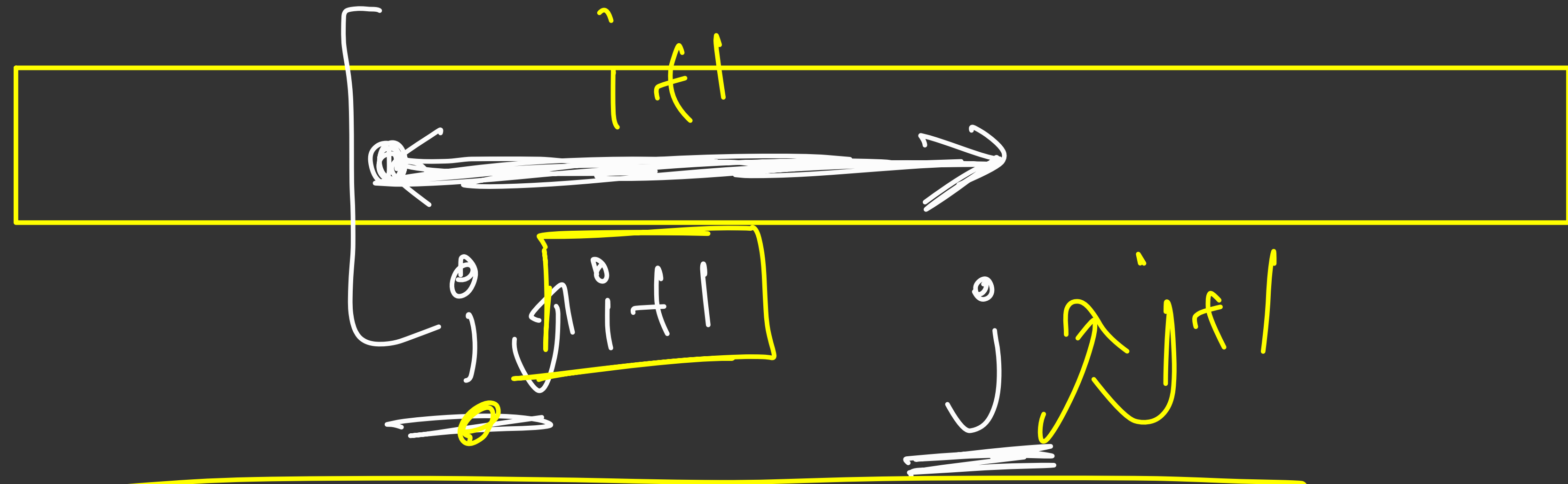
if I know the length of longest subarray which starts at every index

$\longrightarrow$ global maximum of this will be answer

If I know the largest subarray which ends at every index

→ global maximum of this

will DP answer ?



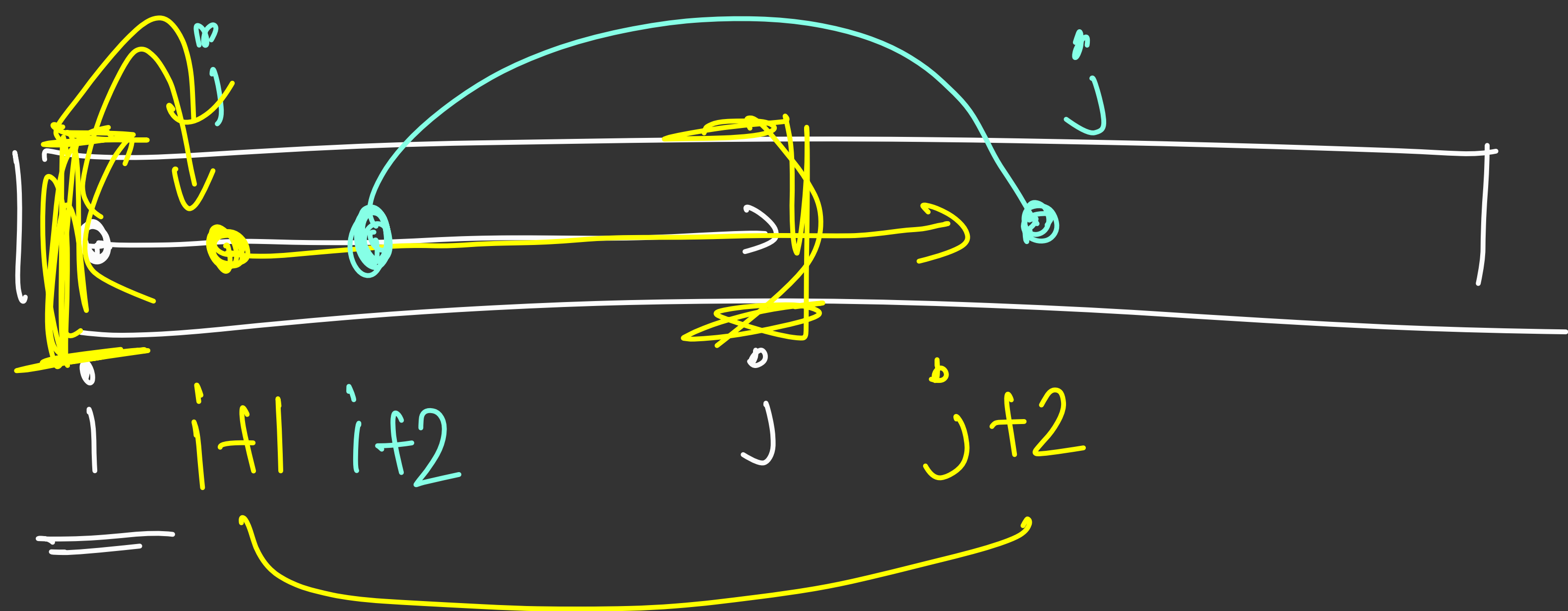

$$\boxed{a[i] + a[i+1] ---- a[j]} \le k$$

but

$$a[i] + a[i+1] ---- a[j] + a[j+1] > k$$

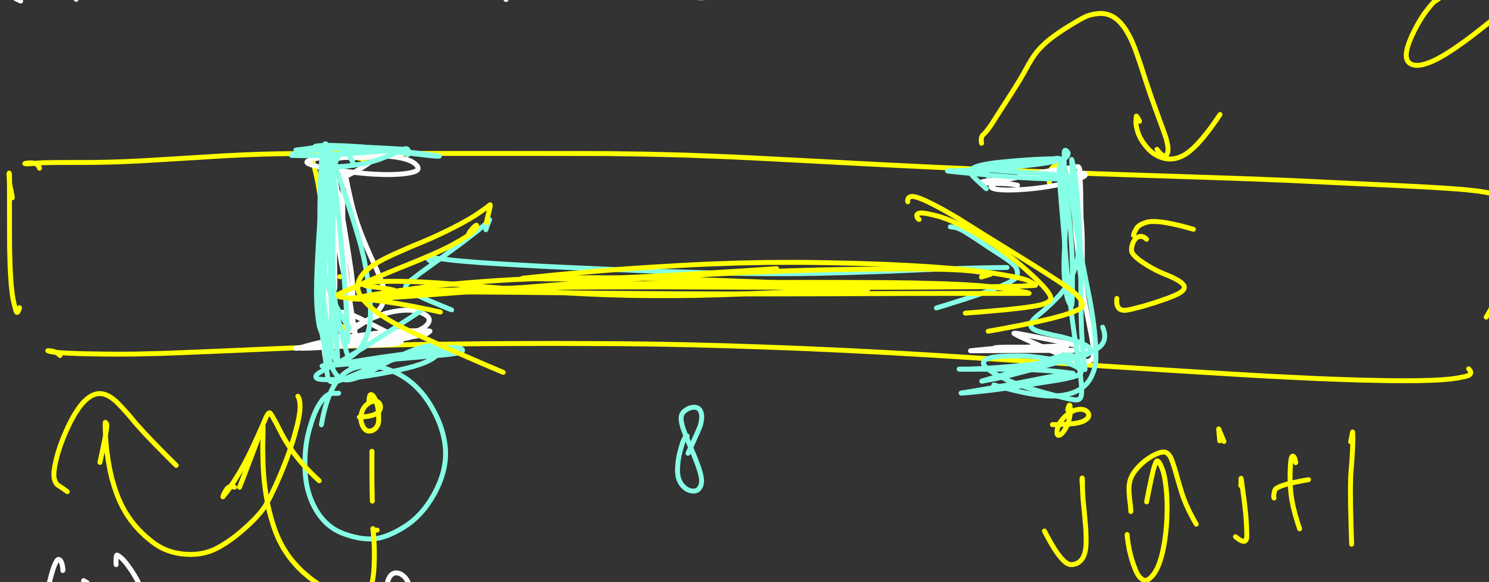
$$\cancel{a[i]+}\ \boxed{a[i+1] ---- a[j]} \le k$$

for every index (i) find out the farthest index (j) s.t sum from a[i] to a[j] $\leq$ k




and take the global max

for every index (j) find out the farthest index (i) to the left such that sum $\leq$ k




$k=10$

$a(i) + a(i+1) \ldots \ldots a(j) \leq k$

but

$$\boxed{a[i-1] + a[i] + a[i+1] \cdots a[j] > k}$$

$$a[i] + a[i+1] \cdots a[j] + a[j+1]$$

but

$$\boxed{a[i-1] + a[i] + a[i+1] \cdots a[j] + a[j+1]}$$

# Good Segments Technique Problem 1

```
vector<int> a(n);
int k;
int ans = 0;
int i = 0, j = 0;
while(j < n){
    // include the jth element in your segment
    sum += a[j]
    while(i <= j && sum > k){ // move left pointer 1 step left
        // do somethign while removing a[i]
        sum -= a[i];
        i++;
    }
    // if current segment is valid, update your answer
    if(sum <= k)
        ans = max(ans, j - i + 1);
    j++; // move right pointer 1 step right
}
```

[1 1 1 2 4 1]

i j

k=5

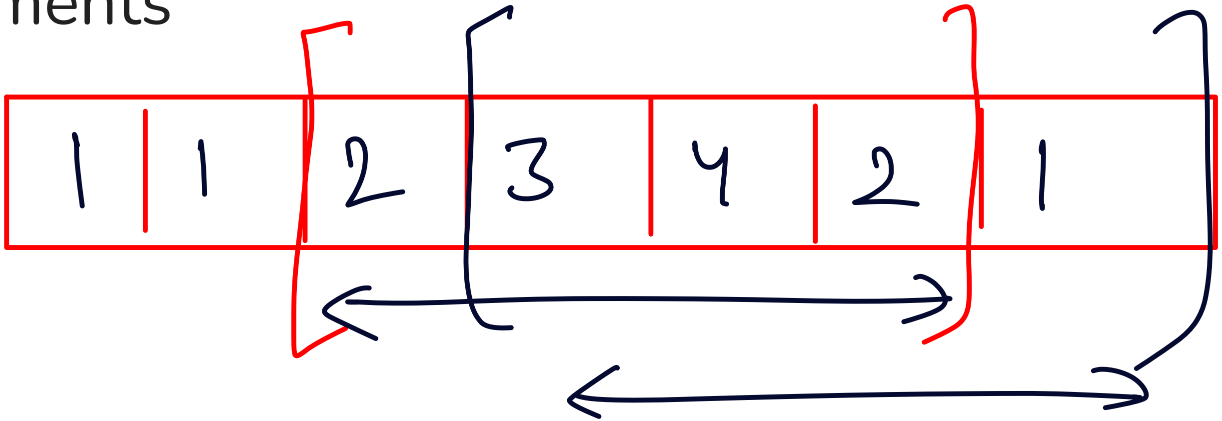
sum=4

ans=4

# Good Segments Technique (Increasing)

k = 3

- Problem 2: Given an array find the length of longest subarray with not more than K distinct elements



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 | 3 | 5 | 6 | 7 | 1 |

$2 \to 1$

$3 \to 2$

$4 \to 1$

$5 \to 1$

$6 \to 1$

4, 5

m.erase(1);


no. of distinct elements from i to j

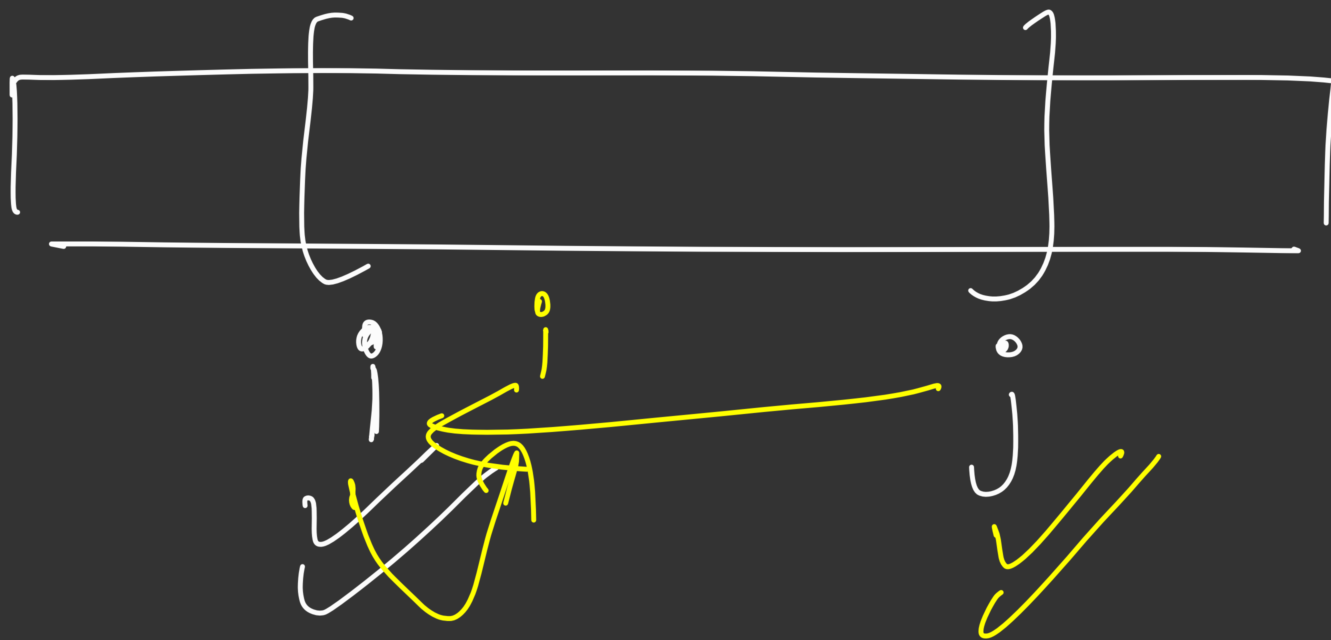
$\leq k$

no. of " " from i-1 to j

$> k$

# Good Segments Technique Problem 2

```
vector<int> a(n);
int k;
int ans = 0;
int i = 0, j = 0;
map<int, int> freq;
while(j < n){
    // include the jth element in your segment
    freq[a[j]]++;
    while(i <= j && freq.size() > k){ // move left pointer 1 step left
        // do somethign while removing a[i]
        freq[a[i]]--;
        if(freq[a[i]] == 0)
            freq.erase(a[i]);
        i++;
    }
    // if current segment is valid, update your answer
    if(freq.size() <= k)
        ans = max(ans, j - i + 1);
    j++; // move right pointer 1 step right
}
```




## good segment

(i to j is a good segment)

(i+1 to j is also a " " )

try to move j forward

(i to j is a good segment)

(i-1 to j is a bad segment)

(i-1 to j+1 is always " ")

try to keep i as much as possible towards left but increase it until the segment is bad fix starting point

(fixing i)

i to j is the best

when you move i forward you try to move j as much as possible towards right until segment is good

fix ending point

(fixing j)

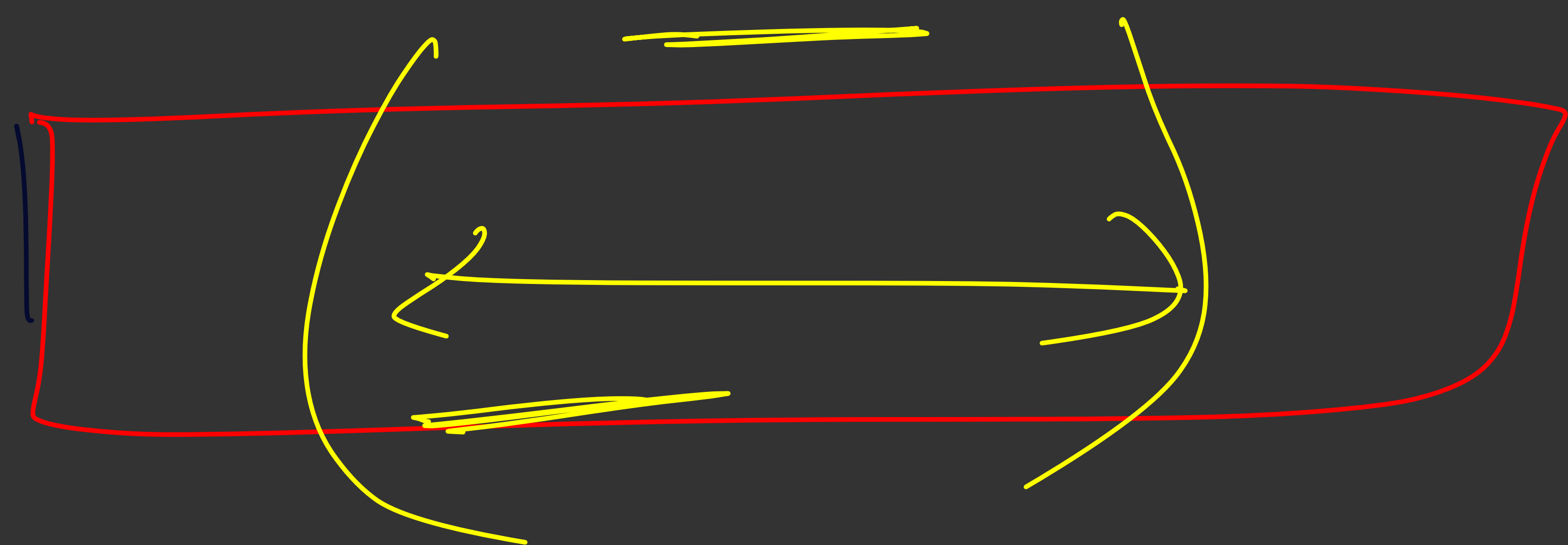
i to j is good

i-1 to j is sad

when you move your j forward you try to keep i pointer as much left as possible but you increase i until segment is sad

If a subarray of size X is good are all the subarray enclosed within this subarray also good ?

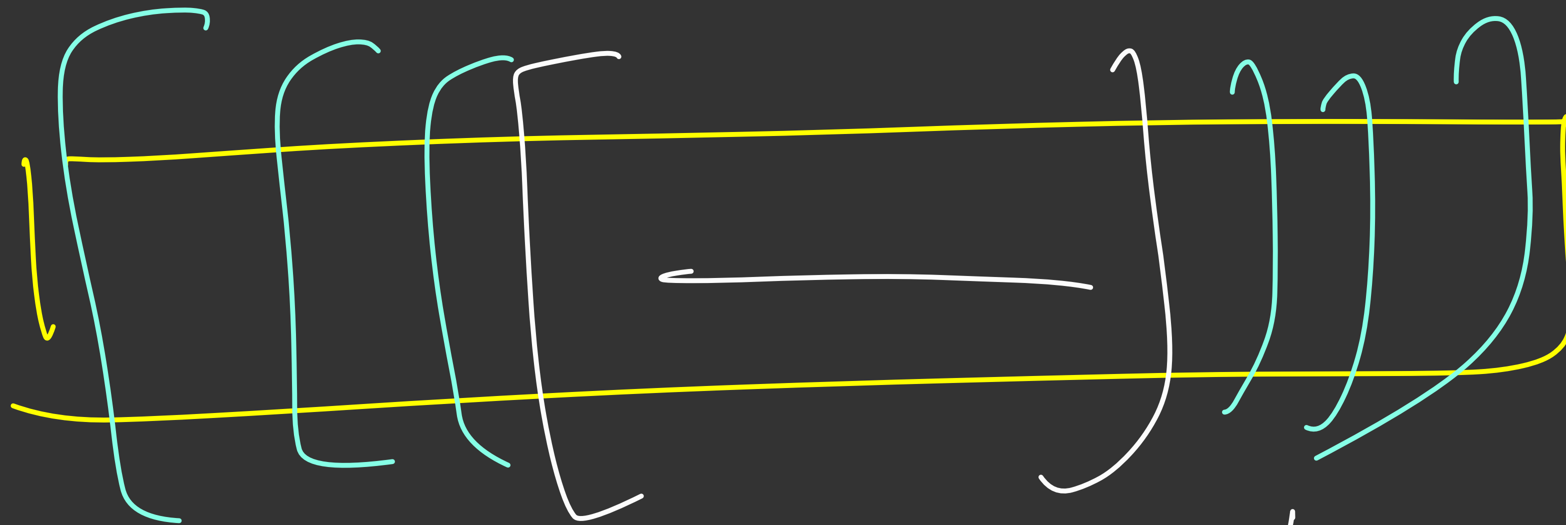
All subarrays enclosed within a good subarray are also good



## Good Segments Technique (Decreasing)

- Problem 3: Given an array of positive integers find the length of smallest subarray with sum of elements >= K

All subarrays enclosing a good subarray are also good



atleast k