Todays Content

1. Deque problems
2. Peak

1Q: Given arr[N] return length of smallest subarray with sum >= k.

Constraints:

$1 <= N <= 10^5$

$-10^9 <= arr[i] <= 10^9$

$-10^{11} <= K <= 10^{11}$

Ex:

	0	1	2	3	4	5	6	7
arr[] = {	4	-2	3	1	4	2	-3	6 }

k = 10 ans = 4.

Idea1: Generate all subarrays:

for all subarrays, iterate & calculate sum

if sum >= k:

ans = min(ans, subarray length);

TC: $O(N^2 * N) = O(N^3)$ SC: $O(1)$

Idea 2:

	0	1	2	3	4	5	6	7		
arr[] = {	4	-2	3	1	4	2	-3	6 }	k = 10	ans = 4.

\# s e len

{0 4} # sum >= k 5

{1 7} # sum >= k 7

{2 5} # sum >= k 4

```
ans = ∞;
i = 0; i < N; i++) {
    # start subarray at i, iterate till we get sum >= k;
    sum = 0;
    j = i; j < N; j++) {
        sum = sum + arr[j];
        if (sum >= k) { # [i..j]          {i.. j j+1
            ans = min(ans, j-i+1);
        } break;
    }
}
return ans;
```

TC: O(N²) SC: O(1)

3. Binary Search.

Target : Min subarray len with sum >= k

Search Space: l = 1 h = N

Discard? m = (l + h)/2;

Say there exists a subarray of m length with sum >= k

$\boxed{m}$ m+1 m+2 . . .

T T F T T F

Just because m is possible we cannot claim, everything on right is True, few might not be possible

Say there doesn't exist a subarray of m length with sum >= k

$\boxed{m}$

T F T T F

We cannot directly discard left side, because few subarray length's might be valid on left side.

4. 2 pointer: Not possible.

1. If on a technique BS is not possible, 2 pointer too is not possible.

2. If subarray {i...j} is possible:

{i.. j} j+1 j+2 .. N-1 : We cannot claim outside are valid

{i .... j-2 j-1 j} : We cannot claim inside are valid

Idea3: Subarrays sums related problems start with pf[] sums

	0	1	2	3	4	5	6	7	8	9	10	11	
arr[] = {	3	2	1	2	3	1	-3	2	7	6	-10	6 }	k = 10

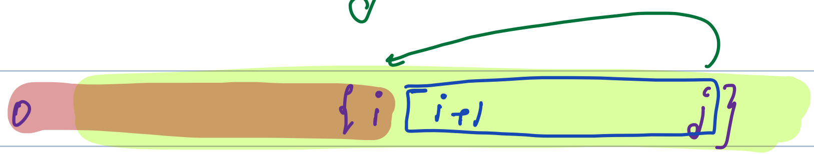
0

	-1	0	1	2	3	4	5	6	7	8	9	10	11
pf[] = {	0	3	5	6	8	11	12	9	11	18	24	14	20 }

When dealing with subarray sums, there will always be a 0 at start.

#obs: if #(i, j) = if pf[j] - pf[i] >= k

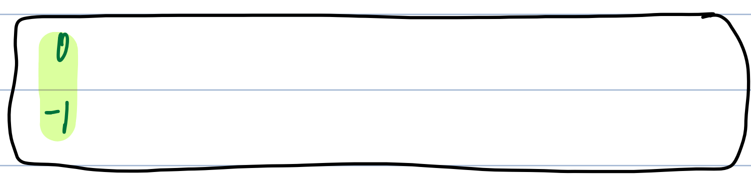
subarray sum [i+1 - j] >= k len = j-i;



#DryRun:

Steps1:

	-1	0	1	2	3	4	5	6	7	8	9	10	11
pf[] = {	0	3	5	6	8	11	12	9	11	18	24	14	20 }



Steps 2:

	-1	0	1	2	3	4	5	6	7	8	9	10	11
pf[] = {	0	3	5	6	8	11	12	9	11	18	24	14	20 }

0	3
-1	0

Step3:

	-1	0	1	2	3	4	5	6	7	8	9	10	11
pf[] = {	0	3	5	6	8	11	12	9	11	18	24	14	20 }

val :	0	3	5
ind :	-1	0	1

Step4:

	-1	0	1	2	3	4	5	6	7	8	9	10	11
pf[] = {	0	3	5	6	8	11	12	9	11	18	24	14	20 }

val :	0	3	5	6
ind :	-1	0	1	2

Step5:

	-1	0	1	2	3	4	5	6	7	8	9	10	11
pfs[] = {	0	3	5	6	8	11	12	9	11	18	24	14	20 }

val :	0	3	5	6	8
ind :	-1	0	1	2	3

Step6:

	-1	0	1	2	3	4	5	6	7	8	9	10	11
pfs[] = {	0	3	5	6	8	11	12	9	11	18	24	14	20 }

11

4 d = 5

val :	0	3	5	6	8
ind :	-1	0	1	2	3

len = ~~x~~ 5

Step7:

	-1	0	1	2	3	4	5	6	7	8	9	10	11
pfs[] = {	0	3	5	6	8	11	12	9	11	18	24	14	20 }

val :	3	5	6	8	11	12
ind :	0	1	2	3	4	5

Step 8:

-1	0	1	2	3	4	5	6	7	8	9	10	11

pf[] = { 0 3 5 6 8 11 12 9 11 18 24 14 20 }

9

6

val :	3	5	6	8	11	12
ind :	0	1	2	3	4	5

* *

Step 9:

-1	0	1	2	3	4	5	6	7	8	9	10	11

pf[] = { 0 3 5 6 8 11 12 9 11 18 24 14 20 }

val :	3	5	6	8	9	11
ind :	0	1	2	3	6	7

Step 10:

-1	0	1	2	3	4	5	6	7	8	9	10	11

pf[] = { 0 3 5 6 8 11 12 9 11 18 24 14 20 }

18

8

val :	3	5	6	8	9	11
ind :	0	1	2	3	6	7

len = . 8 7 6 5

Step 11:

	-1	0	1	2	3	4	5	6	7	8	9	10	11
pf[] = {	0	3	5	6	8	11	12	9	11	18	24	14	20 }

24
9

val :	$\boxed{9}$	$\boxed{11}$	18
ind :	$\boxed{6}$	$\boxed{7}$	8

len = . 3 2

Step 12:

	-1	0	1	2	3	4	5	6	7	8	9	10	11
pf[] = {	0	3	5	6	8	11	12	9	11	18	24	14	20 }

14
10

val :	~~18~~	~~24~~
ind :	~~8~~	~~9~~

len = .

Step 13:

	-1	0	1	2	3	4	5	6	7	8	9	10	11
pf[] = {	0	3	5	6	8	11	12	9	11	18	24	14	20 }

val :	14	20
ind :	10	11

Con: Keep data in Inc order in Deque <ele: index>

For iterate Pf[R]: TC: O(N+N) = O(N) SC: O(N)

Iterate at start of Deque:

if Pf[R] - start of Deque >= k:

update ans;

remove start;

else

break

Iterate at back of Deque:

if Pf[R] <= end of Deque:

remove end

else

insert Pf[R] at end.

```
int length(vector<int> &arr){

    long sum=0;
    vector<long> pf(0, arr.size());

    long sum=0;
    for(int i=0; i< arr.size(); i++){
        sum = sum + arr[i];
        pf[i] = sum;
    }


    deque<pair<long, int>> dq;
    dq.push_front({0,-1});
    int ans=0;
    for( int i=0; i< arr.size(); i++){
        # for pf[i] first check, if we can have ans;
        while( dq.size()>0 && pf[i]- dq.front().first>=k) {
            ans = min(ans, i- dq.front().second);
            dq.pop_front();
        }

        # Before Inserting pf[i] check if we can delete elements
        while(dq.size()>0 && pf[i]< dq.front()){
            dq.pop_back();
        }

        dq.push_back({pf[i],i});
    }

} return ans;
```