Homework Doubts

1. Pick B elements ✓
2. Min swaps to bring all elements less ✓
   than B
3. Choclate distribution ✓ : After Sorting
4. TC doubts :

Pick B elements

Given ar[N] elements & B, return max sum, which can be obtained by picking B elements from corners.

Constraints

$1 \leq N \leq 10^5$

$1 \leq ar[i] \leq 10^9$

$1 \leq B \leq N$

Ex: ar[10] :

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

ans = 20

B = 4

Case-I:

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

Case-II :

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

Case-III:

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

Case-IV:

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

Case-V:

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

Sum	Left	Right
10	4	0
16	3	1
19	2	2
20	1	3
20	0	4

Ex: ar[10] :

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

B = 5

left	Right
5	0

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

sum = 15

P_1 P_2

sum = sum − ar[P_1] + ar[P_2] = sum − 4 + 7 = sum + 3 = 18

$P_1 = P_1 - 1$ $P_2 = P_2 - 1$

left	Right
4	1

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

sum = 18

P_1 P_2

sum = sum − ar[P_1] + ar[P_2] = sum − (−1) + 5 = sum + 6 = 24

$P_1 = P_1 - 1$, $P_2 = P_2 - 1$

left	Right
3	2

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

sum = 24

P_1 P_2

sum = sum − ar[P_1] + ar[P_2] = sum − (3) + 6 = sum + 3 = 27

$P_1 = P_1 - 1$ $P_2 = P_2 - 1$

left	Right
2	3

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

sum = 27

P_1 P_2

sum = sum − ar[P_1] + ar[P_2] = sum − 6 + 7 = sum + 1 = 28

$P_1 = P_1 - 1$, $P_2 = P_2 - 1$

left	Right
1	4

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

sum = 28

P_1 P_2

sum = sum − ar[P_1] + ar[P_2] = sum − 3 + 2 = sum − 1 = 27

$P_1 = P_1 - 1, = -1,$ $P_2 = P_1 - 1$

left	Right
0	5

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

sum = 27

```
long PickElements(int ar[], int N, int B){

    long sum=0;
    long ans= Integer.MINVALUE
    // Case-I: Pick all B elements from leftside: [a_0 a_1 .. a_{B-1}]
                                                          ↑
                                                          P1
        i=0; i<B; i++){  ──────→ TC: O(B)
            sum = sum+ar[i]
        }                              Total iterations
        if(sum > ans){                 TC: O(B) ──→ O(N)
            ans = sum
        }                              SC: O(1)
    // Remaining Cases:
        int P1 = B-1, P2 = N-1;

        while(P1>=0){  ──────→ TC: O(B)
            sum = sum - ar[P1] + ar[P2]
            P1 = P1-1, P2 = P2-1;
            if( sum > ans){
                ans = sum
            }
        }
        return ans;
}
```

Ex: arr[ ] :

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

B = 5

Pick + leave part = sum

left	Right
5	0

0	1	2	3	4	5	6	7	8	9	10
3	6	3	-1	4	8	2	7	6	5	7
				$\uparrow P_1$						$\uparrow P_2$

sum = 15   35 = 50

Pick + left = sum

left	Right
4	1

0	1	2	3	4	5	6	7	8	9	10
3	6	3	-1	4	8	2	7	6	5	7
			$\uparrow P_1$						$\uparrow P_2$	

sum = 18   32 = 50

left	Right
3	2

0	1	2	3	4	5	6	7	8	9	10
3	6	3	-1	4	8	2	7	6	5	7
		$\uparrow P_1$						$\uparrow P_2$		

sum = 24   26 = 50

left	Right
2	3

0	1	2	3	4	5	6	7	8	9	10
3	6	3	-1	4	8	2	7	6	5	7
	$\uparrow P_1$						$\uparrow P_2$			

sum = 27   23 = 50

left	Right
1	4

0	1	2	3	4	5	6	7	8	9	10
3	6	3	-1	4	8	2	7	6	5	7
$\uparrow P$						$\uparrow P_2$				

$\boxed{\text{sum = 28}}$ 22 = 50

left	Right
0	5

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

sum = 27   23 = 50

Idea 2: Pick elem + leftout ele = Sum of arr ele   TC: O(N)

$\underbrace{\text{Pick elem}}_{\text{max}}$ = Sum of arr[ ] - $\underbrace{\text{leftout ele}}_{\text{min}}$

Final ans = Sum of arr[ ] - {min Subarray Sum of len, N-B}

2Q) Min Swaps to bring all elements <= B together.

Continous:

	0	1	2	3	4	5	6	7	8	9	10
arr[11] :	6	8	3	10	11	2	14	1	5	9	12

B = 7

#ele <= B : 5

Swaps

0	1	2	3	4	5	6	7	8	9	10
6	8	3	10	11	2	14	1	5	9	12

3 swaps

0	1	2	3	4	5	6	7	8	9	10
6	8	3	10	11	2	14	1	5	9	12

3 swaps

0	1	2	3	4	5	6	7	8	9	10
6	8	3	10	11	2	14	1	5	9	12

3 swaps

0	1	2	3	4	5	6	7	8	9	10
6	8	3	10	11	2	14	1	5	9	12

3 swaps

0	1	2	3	4	5	6	7	8	9	10
6	8	3	10	11	2	14	1	5	9	12

2 swaps

0	1	2	3	4	5	6	7	8	9	10
6	8	3	10	11	2	14	1	5	9	12

2 swaps

0	1	2	3	4	5	6	7	8	9	10
6	8	3	10	11	2	14	1	5	9	12

3 swaps

Idea: Calculate no: of ele less than <= B in every window.

Ex: arr[9] = {3 10 14 6 18 4 2 7 11}

B=8

Step1: Element < 8 = 5

Replace all ele <=8, with : 1, ele > 8 with: 0

	0	1	2	3	4	5	6	7	8
arr[9] =	1	0	0	1	0	1	1	1	0

sum = 2

sum = 2

sum = 3

sum = 4

sum = 3

max sum = 4

Total ele <= B : 5

min swaps = 5-4 = 1.

Chcolate distribution: Skip it. → { This is different from assignment }

Given N student marks, assign choclate to all N students in such a way that, Calculate min of choclates to assign to all N students.

Note1: Each student get atleast 1 choclate

Note2: if $ar[i] > ar[i-1]$

Choclates assigned to i^{th} student should be more $i-1^{th}$ student

Note3: if $ar[i] > ar[i+1]$

Choclates assigned to i^{th} student should be more $i+1^{th}$ student

Ex1: ar[]:

	0	1	2	3	4	
ar[]	1	5	2	1	6	
cho	1	1	1	1	1	
cho	1	5	2	1	6	: 15
cho	1	3	2	1	2	:

Total Chocolate

Ex2:

	0	1	2	
ar[]	3	100	60	
choc	1	1	1	: 7
choc	1	3	2	: 6 choclate
choc	1	2	1	: 4 choclate

Ex3:

	0	1	2	3	
ar[]	3	100	60	80	
choc	1	2	1	2	: 6 choclates

Ex3:

	0	1	2	3	
ar[]	3	100	60	50	
choc	1	3	2	1	: 7 choclates.

Idea:

Only1: if ar[i] > ar[i-1]

Choclates assigned to i^{th} Student should be more $i-1^{th}$ Student

Only2: if ar[i] > ar[i+1]

Choclates assigned to i^{th} Student should be more $i+1^{th}$ Student

	0	1	2	3	4
ar[ ] :	3	6	2	8	10
left :	1	2	1	2	3
Right :	1	2	1	1	1
Both :	1	2	1	2	3

	0	1	2	3	4
ar[ ] :	3	6	9	11	7
left :	1	2	3	4	1
Right :	1	1	1	2	1
Both :	1	2	3	4	1

	0	1	2	3	4	5	6	7	8	9
ar[ ] :	2	6	3	1	10	1	2	20	5	2
left :	1	2	1	1	2	1	2	3	1	1
right :	1	3	2	1	2	1	1	3	2	1
Both :	1	3	2	1	2	1	2	3	2	1

Idea: Given an arr[]:

→ Calculate left[]

→ Calculate Right[]

& Take max().