Rough work

Q1. Pick from both sides! </> Solved

Using hints except Complete Solution is Penalty free now

Use Hint

Problem Description

You are given an integer array **A** of size **N**.
You have to perform **B** operations. In one operation, you can remove either the leftmost or the rightmost element of the array **A**.
Find and return the **maximum possible sum** of the **B elements** that were removed after the **B** operations.
NOTE: Suppose **B = 3**, and array A contains 10 elements, then you can:

- Remove 3 elements from front and 0 elements from the back, OR
- Remove 2 elements from front and 1 element from the back, OR
- Remove 1 element from front and 2 elements from the back, OR
- Remove 0 elements from front and 3 elements from the back.

Problem Constraints

$1 <= N <= 10^5$
$1 <= B <= N$
$-10^3 <= A[i] <= 10^3$

Input Format

First argument is an integer array **A**.
Second argument is an integer **B**.

Output Format

Return an integer denoting the maximum possible sum of elements you removed.

Example Input

Input 1:

```
A = [5, -2, 3 , 1, 2]
B = 3
```

Input 2:

```
A = [ 2, 3, -1, 4, 2, 1 ]
B = 4
```

Example Output

Output 1:

```
8
```

Output 2:

```
9
```

Example Explanation

Explanation 1:

```
Remove element 5 from front and element (1,
2) from back so we get 5 + 1 + 2 = 8
```

Explanation 2:

```
Remove the first element and the last 3
elements. So we get 2 + 4 + 2 + 1 = 9
```

Approach (from Hints)

Hint 1

In this array we have to remove some elements from left to right, so the calculation of cumulative sum i.e. **prefix sum** can be helpful.
Similarly, we have to remove some elements from right to left, so the calculation of cumulative sum i.e. **suffix sum** can be helpful.

Try to cover all the cases one by one, using prefix and suffix sum calculation.

Note: Take care of **corner cases** where only elements from left OR only elements from right are removed.

Solution Approach

Approach using Prefix and Suffix Sums:
Maintain two arrays prefix[i] and suffix[i] where prefix[i] denotes sum of elements from index [0,i] and suffix[i] denotes sum of elements from index [i,N-1].
Now iterate from left and one by one pick elements from left for example: if you pick ‘a’ elements from left and remaining ‘k-a’ elements from right.
So the sum in this case will be prefix[a-1] + suffix[n-(k-a)]
Maintain the maximum among all and return it.
Time Complexity: O(N)
Space Complexity: O(N)
where N is number of elements in array A
Bonus: Try solving it in O(1) space.

$\Rightarrow$ A =

2	3	-1	4	2	1
0	1	2	3	4	5

$\rightarrow$ len = 6

pSum =

2	5	4	8	10	12
0	1	2	3	4	5

sSum =

11	9	6	7	3	1

say B = 5

i = 0

pS[0] + sSum[N-B-i]

if we select 'a' elements from lyt, we must choose 'B-a' element from right

→ since index 'i' starts from '0' $\Rightarrow$ a = (i+2) from lyt

$\Rightarrow$ B-a = (B-i-2) from right

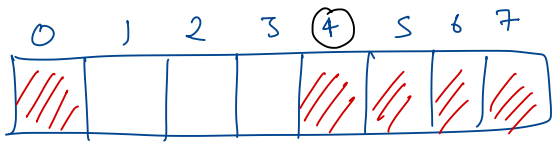
starting index of 'B-a' elements from right = N - (B-a)

= N - (B-i-2)

= (N+1 - B + i)

constant K

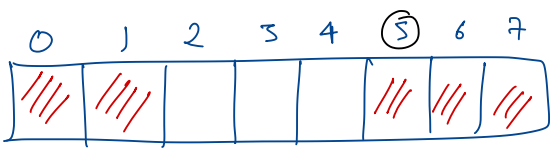
so when $i = 0$ or $a = 1$ $N = 8, B = 5$



starting index of right-half element

$= (8 + 1 - 5 + 0) = 4$ ✓

when $i = 1$ or $a = 2$



starting index of left-most element in the right-batched element

$= N + 1 - B - i = 8 + 1 - 5 + 1$

$= 5$ ✓

⋮

when $a = B$ or $i = a - 1 \Rightarrow i = B - 1$

leftmost index $= N + 1 - B + i$

$= N + \cancel{1} - \cancel{B} + \cancel{B} - \cancel{1}$

$= N \longrightarrow$ out of bounds

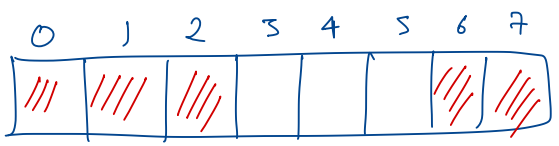
⇒ We can only check the first 'a-1' elements from left

⇒ index 'j' will from elements 2 → a-2

OR index 0 → a-2

psum & ssum

At any given moment,



i = 2

sum = psum[2]

i.e psum[i]

sum = arr[6] + arr[7]

= ssum[6] → (N - B + 1 + i)

→ 8 - 5 + 1 + 2

= 6 ✓

After each iteration,

we have to carry forward the highest leftSum + rightSum

Edge Case Check

Even before iterating the loop, we can check the sum of the first B left elements, and last B elements.

whichever is the highest, gets set as the answer which can change depending on the iteration

left = pSum[B-1]

right = sSum[N-B]