→ Eg: [1 4 8 10], [2 3 6 7 9], k = 6

We have to choose 6th element from sorted array

[1 2 3 4 6 (7) 8 9 10]

So, we divide the array such that we have 6 element on leftside

	6	4
arr1	0	
arr2	5	

↳ need 1 more

Now if we decide to pick 0 element from arr1, then we will have to pick all element from arr2, but we notice that arr2 have only 5 element. ⟶ so no matter what left side array count cannot be 6.

In that case, we cannot be picking 0 element from arr1, there has to be atleast 6−5 = 1 pick from arr1, ∴ Low = max(k−n, 0)

Now Similarly, if we decide to pick 0 element from arr2, our arr1 doesnot have 6, element, it only has 4

	6	4
arr1	4	
arr2	0	

need 2 more

So to prevent that, we cannot have 0 pick from arr2, we need to have 6-4 = 2, pick from arr2

high = min(k, m) ↳ this will make sure that we donot pick all elements from arr1 & pick 0 elements from arr2

Eg: m = 4 n = 5

b	s
1 —— 4	
5	2

low = max(0, k-n)

= max(0, 6-5) = 1

high = min(k, m)

= min(6, 4) = 4

So based on this, arr1 → will have atleast 1 element & max of 4, element

* So In that case

when arr1 = 1, arr2 = 6-1 = 5, we can pick 5 element from arr2 [# all element from arr 2]

* When we pick 4 element from arr1 (# all element from arr1) the we will pick at least 2 element from arr 2

6 - 4 = 2