K-Closest points to Origin → QuickSelect + BinarySearch

Eg: [[-2 10] [-4 -8] [10 7] [-4 -7]]

Initially: Hypothetically assume, we donot know what our pivot Index is, We have to find One pivot Idx

This pivot Index should be found in such a way that, all the elements to the left of the pivot Index, should be smaller than the Selected Pivot Index.

Assume Initially

pivot Idx = n

Left = 0 , right = n-1 , K = 3

We check if pivot Idx = K, if equal then all element to left are smaller than pivotIdx, hence we have found K smallest | Closest point.

[[-2 10] [-4 -8] [10 7] [-4 -7]]

↑ left ↑ right ↑ pivot Idx

we perform binary Search on array & find a new pivot Idx,

We keep doing this Until pivot Idx = K

[-2 10 , -4 -8 , 10 7, -4 -7]

↑ lft   mid   ↑ right

Now keep pushing element greater than mid to its right

mid

[-4 -7 -4 -8 10 7 , -2 10]

↑ lft   ↑ right

[-4 -7 -4 -8 10 7 -2 10]

↑ lft   ↑ right

mid

[-4 -7 10 7 -4 -8 -2 10]

↑ lft ↑ right

[-4 -7 10 7 -4 -8 -2 10]

↑ right   ↑ lft

If lft < pivot :

then return lft + 1, because all the value to lft must be smaller than the new pivot that we return.

But here lft > pivot | Mid

so we just return lft, Because Everything to left is sorted.

new_pivot_Idx = 1 != k.

so we repeate Binary search

But this time we perform Binary Search on adjusted left & right pointer

[-4 -7 10 7 -4 -8 -2 10]

↳ new pivot

If we carefully observe all the element to the left of new pivot, will for sure be smaller.

But there is no such gurantee for elements to the right of it.

→ Now we check, if our k is to left of new pivot point then our right = new_pivot − 1

→ If our k is to right of new_pivot then our left will be equal to new_pivot, because there is no gurantee that our elements on right are completely greater than current value

so left = new-pivot

Second iteration

mid

[-4 -7 10 7 -4 -8 -2 10]

↑ left ↑ right

[-4 -7   -2 10   -4 -8   10 7]

mid

↑ left   ↑ right

[-4 -7   -4 -8   -2 10   10 7]

mid

left
right

[-4 -7   -4 -8   -2 10   10 7]

mid

↑ right   ↑ left

since our left is not less than our mid, we return left are new_pivot

since our new_pivot < k
we perform binary search of right side

[-4 -7   -4 -8   -2 10   10 7]

mid

↑ left   ↑ right

[-4 -7   -4 -8   -2 10   10 7]

mid

left   right

[-4 -7   -4 -8   10 7   -2 10]

mid

↑ left   ↑ right

[-4 -7   -4 -8   10 7   -2 10]

↑ right   ↑ left

Our new_pivot = 2

2 < k (3)

we perform binary search on right part

left = new_pivot

4th iteration

mid

[-4 -7   -4 -8   10 7   -2 10]

left   right

mid

[-4 -7, -4 -8, -2 10, 10 7]

left right

mid

[-4 -7, -4 -8, -2 10, 10 7]

right   left

Our new_pivot = k :

hence we return Every Element to the left of it

[[-4, -7], [-4, -8], [-2, 10]]

Now, If we observe carefully, we see that all the element to the left of pivot are smaller than pivot