Step. 1 Problem

- k smallest elements in same order using O(1) extra space

Input: an array of n-elements and a number k.
Output: k smallest elements from the array but they must be in the same order as they are in given array
Constraint: we are allowed to use only O(1) extra space.

Examples:

Input	Output
[4, 2, 6, 1, 5], k = 3	[4, 2, 1]
[2, 2, 6, 1, 5], k = 2	[2, 1]
[], k = 0	[]

Step. 2 Methods

Methods	Time complexity	Extra space	Same order?
Sort first (merge, quick, heap, etc.) then return elements [4, 2, 6, 1, 5], k = 3 => [1, 2, 4, 5, 6]	O(nlogn)	O(1) or O(n) for merge sort	No
Sort **copied** array then traverse in origin array [4, 2, 6, 1, 5], k = 3 => [4, 2, 6, 1, 5] => [4, 2, 1] [1, 2, 4, 5, 6]	O(nlogn)	O(n)	Yes
Quick select **k smallest** then traverse in origin array [4, 2, 6, 1, 5], k = 3 => [4, 2, 6, 1, 5] => [4, 2, 1] [1, 2, 4]	Avg: O(kn)	O(n)	Yes
★ Insertion sort* then return elements [4, 2, 6, 1, 5], k = 3 => [4, 2, 1]	$O(n^2)$	O(1)	Yes

Step. 3 Core idea

0. Given an array and k

1. Aim: Move k min elemts
to the first k positions

2. Get max of first k elemts
and traverse from k + 1 elemt

3. if num < max :
    delete max
    move elemts one position left
    set arr[k - 1] = num
  elif num >= max :
    continue

4. Return output

[7, 2, 6, 1, 5, 3, 9] K = 3

[7, 2, 6, 1, 5, 3, 9]

[7, 2, 6, 1, 5, 3, 9]

Index = 0

[2, 6, 1, 5, 3, 9]

Step. 4 Example

K = 5

[7, 2, 6, 1, 5, 3, 9]

↓ Select first k elements

[7, 2, 6, 1, 5, 3, 9]

↓ Get value and index of the maximum element

[7, 2, 6, 1, 5, 3, 9]

Compare: 7 > 3

↓ Delete the maximum value and shift elements one place left, arr[4] = 3

[2, 6, 1, 5, 3, 9]

Compare: 6 < 9

↓ Continue

[2, 1, 5, 3, 4, 9]

↓ Output

[2, 1, 5, 3, 4]

Step. 5 Algorithm

```
getKsmallest(arr, k)
```

Input: An array arr and an integer k which <= len(arr)
Output: k smallest elements in the same order as they are in arr

```
IF arr EQUALS [] or k EQUALS len(arr):
    RETURN arr
SET index TO k
WHILE index < len(arr):
    SET elemMax TO max(arr[:k])
    SET currentValue TO arr[index]
    IF currentValue < elemMax:
        arr.remove(elemMax)
        SET arr[k-1] TO currentValue
        index -= 1
    index += 1
RETURN arr[:k]
```

Step. 6 Complexity

- Time complexity.

$$T_{\text{find}}(n) = \begin{cases} 0 & \text{if } n = 0, 1 \text{ or } k = 0, n \\ T_{\text{traversal}}(n-k) \cdot T_{\text{shift}}(n) & \text{if } n > 0 \end{cases}$$

$$T_{\text{traversal}}(n-k) \in \Theta(n), T_{\text{shift}}(n) \in \Theta(n)$$

$$\text{Solving, } T_{\text{find}}(n) \in \Theta(n^2)$$

- Space complexity.

$$S_{\text{find}}(n) \in \Theta(1)$$

Step. 7 Implementation

```
def getKsmallest(arr, k):
    if arr == [] or k == len(arr):
        return arr
    index = k
    while index < len(arr):
        elemMax = max(arr[:k])
        currentValue = arr[index]
        if currentValue < elemMax:
            arr.remove(elemMax)
            arr[k-1] = currentValue
            index -= 1
        index += 1
    return arr[:k]
```

arr = [1, 1, 2, 2, 3, 1, 1], k = 5

↓

Output: [1, 1, 2, 1, 1]

arr = [1, 1, 2, 2, 3, 1, 1], k = 4

↓

Output: [1, 1, 1, 1]

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

↓

Output: [1, 3, 4, 2, 0]

Step. 8 Extensions

Methods	Time complexity	Extra space	Same order?
Sort first (merge, quick, heap, etc.) then return elements [4, 2, 6, 1, 5], k = 3 => [1, 2, 4, 5, 6]	O(nlogn)	O(1) or O(n) for merge sort	No
Sort **copied** array then traverse in origin array [4, 2, 6, 1, 5], k = 3 => [4, 2, 6, 1, 5] => [4, 2, 1] [1, 2, 4, 5, 6]	O(nlogn)	O(n)	Yes
Quick select **k smallest** then traverse in origin array [4, 2, 6, 1, 5], k = 3 => [4, 2, 6, 1, 5] => [4, 2, 1] [1, 2, 4]	Avg: O(kn)	O(n)	Yes
★ Insertion sort* then return elements [4, 2, 6, 1, 5], k = 3 => [4, 2, 1]	$O(n^2)$	O(1)	Yes

Thank you! Any suggestion or thought is very welcomed!