QuickSelect

Agenda

- Warm-up
- Partition algorithm
    - Pair activity: partition step-by-step
- Learn QuickSelect algorithm
- QuickSelect implementation (2-day project)
- Exit-ticket

Warm-up

How would you solve the following problem?

Imagine you are trying to find the 3rd smallest score in a class of students without sorting all the scores.

data = [90, 86, 100, **65**, 95, **72**, **70**, 92], k = 3rd

Output: **72**

QuickSelect

It is a selection algorithm. It looks for the k-th smallest element in an unsorted array.

How?

The strategy to implement the Quickselect is another algorithm call **PARTITION**.

Partition Algorithm - Objective

It is a technique used to divide a collection of data (usually an array) into two distinct parts based on a chosen "pivot" element, placing all elements smaller than the pivot on one side and all elements larger than the pivot on the other side. The pivot ends up in its correct position.

Original array:

pivot

65	86	100	**90**	95	72	70	92

Swap pivot, so it is at index 0:

pivot

90	86	100	65	95	72	70	92

Applying the partition algorithm:

pivot

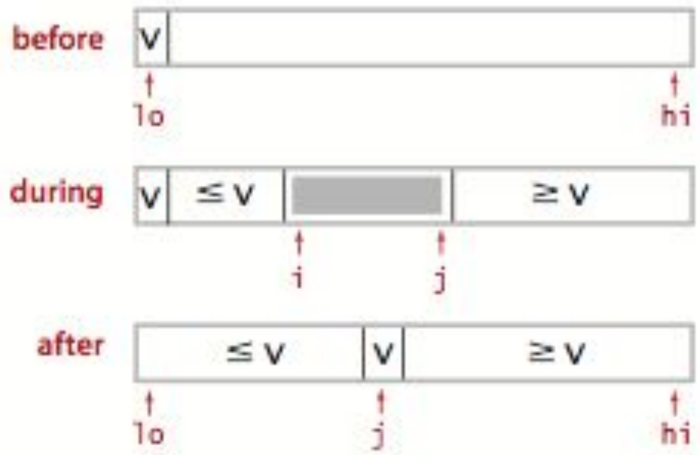
72	86	70	65	**90**	95	100	92

<=90 **>=90**

Partition Algorithm - Strategy

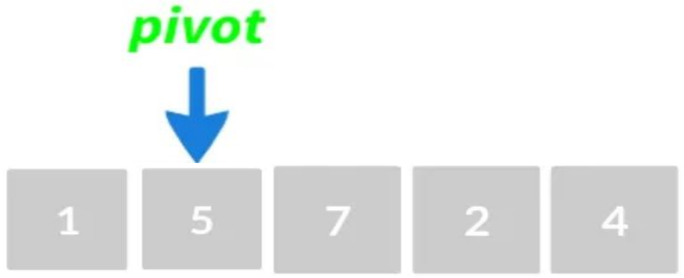
Choose an element from the array, we are going to call it pivot.

Then, create partitions based on the pivot

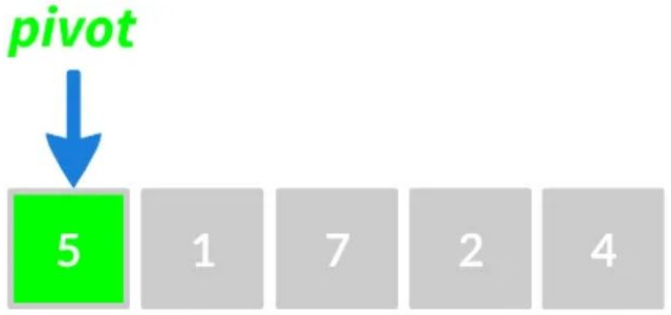


Partition Method: Step-by-Step

1. Create an array of integers and randomly choose an index from the array. The value at that index will be your pivot.



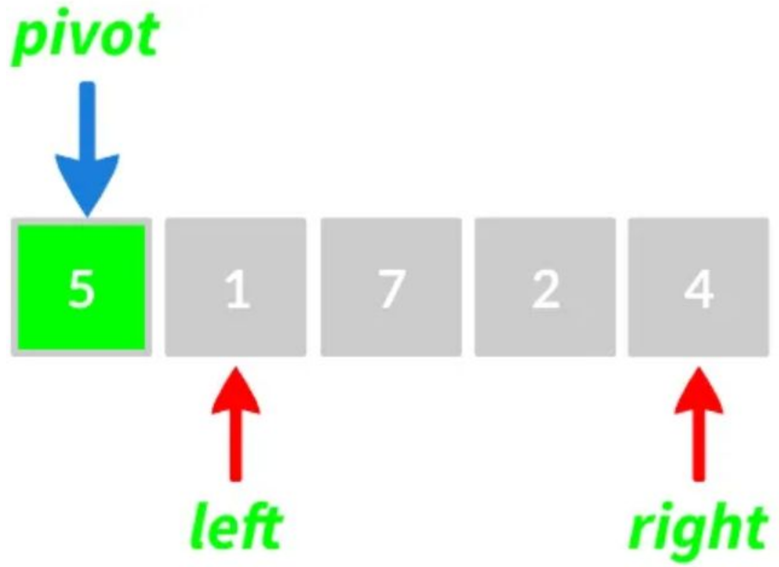
2. If the selected index is not at index zero Swap the pivot with the element at index zero.



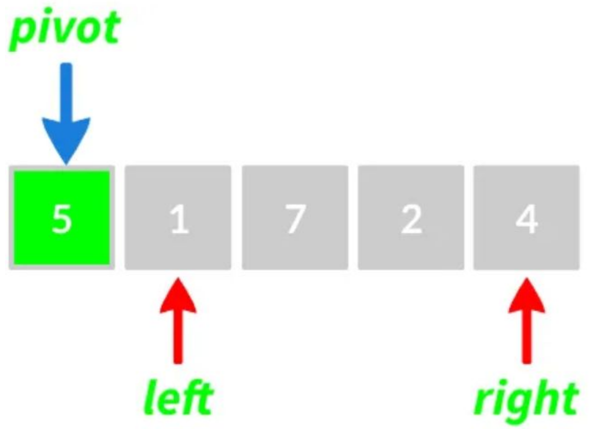
Partition Method: Step by Step

3. Set up "left" and "right" pointers:

“left index” => “leftmost” element
“right index” => "rightmost” element



Partition Method: Step-by-Step



Pair activity:

Follow this rules and find the next steps.

Iterate over the array until left crosses right:

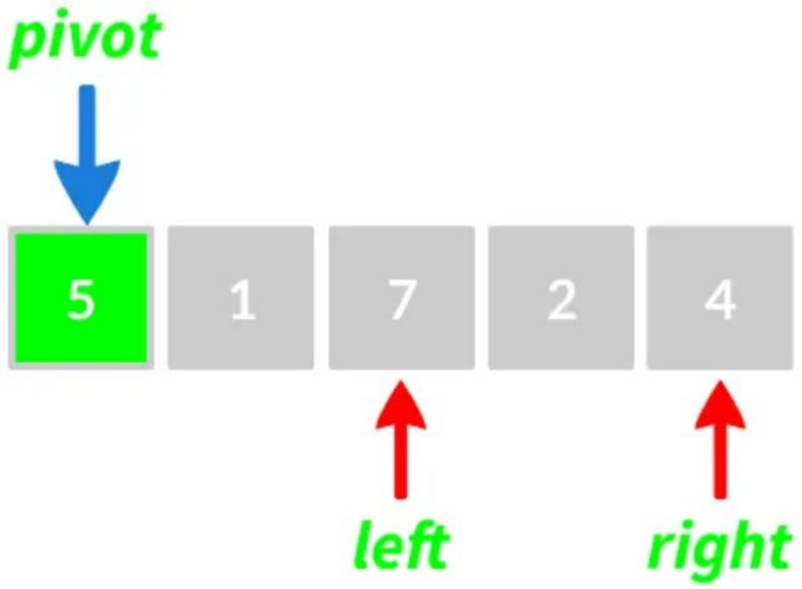
- While arr[left] ≤ pivot, move left forward
- While arr[left] > pivot, swap arr[left] with arr[right] and move right backward

Once left ≥ right, swap the pivot (arr[0]) with arr[left].

Use your notebook or whiteboard, draw the status of the array and indices (left, right) after each iteration.

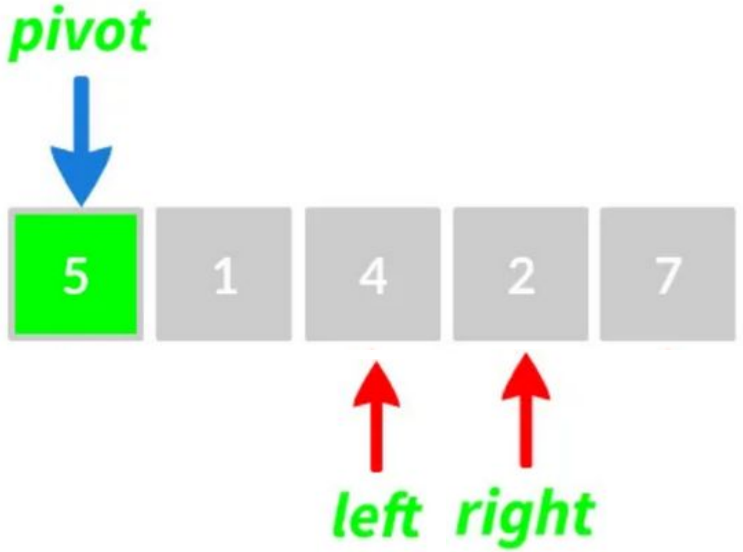
Partition Method: Step by Step

4. Move left cursor to the right



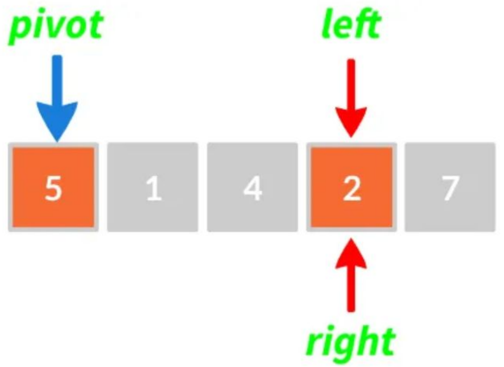
Partition Method: Step by Step

5. Swap between “right” and “left” elements and move right cursor to the left.

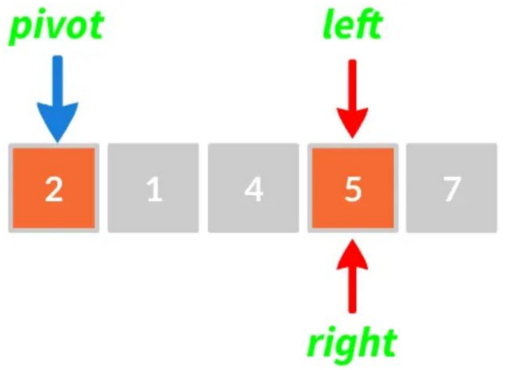


Partition Method: Step by Step

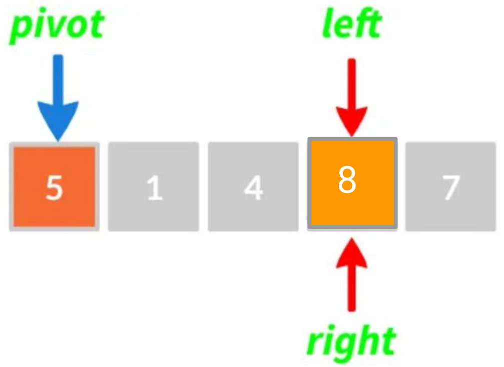
6. Move left cursor to the right.



7. Swap the left element with the pivot



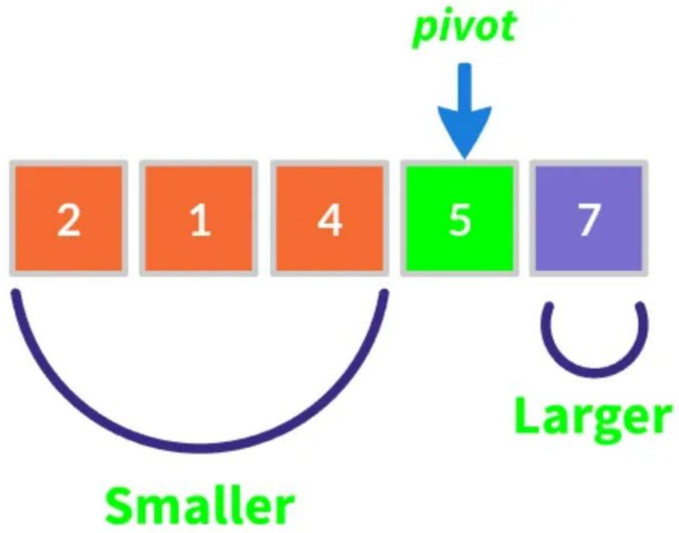
What if the element arr[left] is greater than the pivot?



Swap the element at arr[left - 1] with the pivot

Partition Method: Step-by-Step

8. Return the index of the **final position of the pivot element => 3**



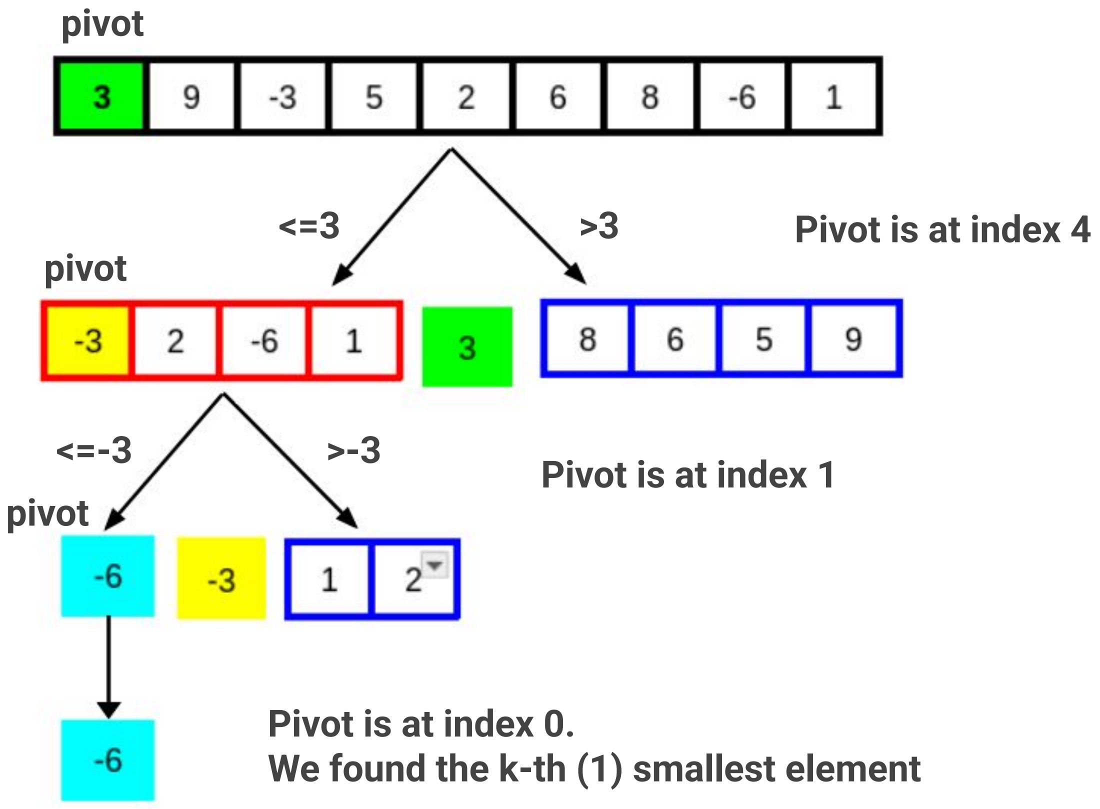
How the partition method will help us implement the Quickselect?

The **partition** method **helps** Quickselect by **organizing the array around a pivot**, placing smaller elements on the left and larger ones on the right.

Since the **pivot ends up in its correct position**, we can quickly **determine if it is the k-th smallest element.**

If not, we recursively search only the relevant half of the array, avoiding unnecessary sorting and reducing the problem size with each step.

Quick Select



Pivot is at index 4

Pivot is at index 1

Pivot is at index 0.
We found the k-th (1) smallest element

Let's find the first smallest element in this array.

k = 1

index = 0

Quick Select - Pseudocode

QuickSelect method to find the k-th smallest element:

```
quickSelect(data, k, start, end)
    pivot_pos  = partition(data, start, end)
    if pivot_pos > k - 1
        Recursive call to your quickSelect(data, k, start=?, end=?)
    Else if pivot_pos < k - 1
        Recursive call to your quickSelect(data, k, start=?, end=?)
    Else
        k-th element found return the value at [k - 1]
```

Coding Time

Lab is posted on our website: https://novillo-cs.github.io/apcsa/labs/

This is a **2-day project (2/26, 2/27)**

Today's homework: Dedicate 30 min to work on this lab at home.

Tomorrow: We will review any questions you might have and continue working on the code.