

Name: Shreenidhi Deepak Pai

Student ID: 1002232249

Approach

Q 1. Every Step of the Quick Sort Algorithm-

Partition: This will be just like partition in a quick sort where we choose an element and create two halves.

All smaller than pivot goes to the left.

Greater elements go to the right.

Recursion / Iteration: After we do the partition, then we will have to find it out that what is rank of the pivot element i.e., how many smaller elements are there than our Pivot. If the rank is same as i then i th element in output. If not, we recurse on the left or right partition according to our rank.

Explanation:

Partition reorder the array in such a way that all elements greater than pivot is moved to right and small goes left.

Quickselect recursively focuses on the sub-array which can contain the i -th smallest element based on rank of a pivot.

So, the next function situated is of `ith_order_statistic` and it calls quicksort that iterates in a way indexing starts with 0 but i think about my user who thinks from center minimum if I say 'st', maximum if print last.

Example:

Given the array [12, 3, 5, 7, 19, 26, 1], if you are looking for the 4th smallest element so it would return.


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  [2] 2311 1 1
● (base) shreenidhi@Shreenidhis-Air DAA-HANDS-ON-8 % python quicksort.py
  The 4th smallest element is 7
○ (base) shreenidhi@Shreenidhis-Air DAA-HANDS-ON-8 %
```

Q 2. The implementation of the data structures using fixed-size arrays for a Stack, Queue, and Singly Linked List is complete:

Stack: Pushed 10, 20, 30, and then popped the top element, which is 30.

Queue: Enqueued 10, 20, 30, and then dequeued the front element, which is 10.

Singly Linked List: Added 10, 20, 30 and traversed the list, which returns [30, 20, 10].



```
Stack pop result: 30
Queue dequeue result: 10
Singly Linked List traversal: [30, 20, 10]
```