Order Statistics Algorithm

Problem Description

Implementing a randomized algorithm to find the kth order statistic in an unsorted array of integers. The kth order statistic represents the element that would be at the kth position if the array were sorted in ascending order.

Suggested solution: Floyd-Rivest algorithm

The Floyd-Rivest algorithm is a randomized selection algorithm used to find the k-th smallest element in an unsorted array efficiently. It is an extension of the quickselect algorithm and offers an improved worst-case time complexity (we discussed in class that improving performance is one benefit of randomized algorithms). The algorithm works by recursively partitioning the input array into subarrays based on a randomly selected pivot element. It then narrows down the search range by focusing on the relevant subarray that contains the desired element. This process continues until the k-th smallest element is found. What sets the Floyd-Rivest algorithm apart is its ability to achieve a good average-case performance while also providing a worst-case time complexity of O(n), where n is the size of the input array. This is achieved by carefully selecting the pivot element and performing a series of partitioning steps. By combining randomization and efficient partitioning techniques, the Floyd-Rivest algorithm offers a reliable and fast solution for finding the k-th smallest element. It is commonly used in applications where an efficient selection of elements is required, such as order statistics, approximate median finding, and more.

Requirements

- Write a program that takes three parameters: an integer `k`, an integer `size` (length of the array), and an integer array `arr`. The program should return only one value, the kth order statistic.
- The program should handle cases where `k` is out of bounds (less than 0 or greater than `size - 1`) and return -1 appropriately.
- Use an **efficient randomized algorithm**.
- Test your implementation with different arrays and values of k to ensure correctness.
- Use comments to explain your code.

Instructions

- Submission format: single Python file named as your roll number. Example: `M20AIE200.py`
- The program should take the input from the command line arguments.
  Example: `python M20AIE200.py 2 7 1 4 6 2 8 5 7`
  Where `k = 2`, `size = 7` and `arr = {1, 4, 6, 2, 8, 5, 7}`
- Note that the array `arr` may have repeated elements and the variable `k` takes values from 0 to size - 1. That is when `k=0`, the program would return the smallest element in the array.
- Ensure the efficiency. Using a naive method to solve the problem will not be accepted.

- In the following program, the three variables are already assigned values from command line arguments and they are passed to the function `kthOrderStatistic`.
  Complete the program by only implementing the function `kthOrderStatistic`.

```
#---------------------------------
import random
import sys

def kthOrderStatistic(k, size, arr):
    return

if len(sys.argv) < 4:
    print("Usage: python roll_no.py K size arr[1] arr[2] ... arr[size]")
    sys.exit(1)

k = int(sys.argv[1])
size = int(sys.argv[2])
args = sys.argv[3:]
arr = [int(arg) for arg in args]

result = kthOrderStatistic(k, size, arr)
print(result)
#---------------------------------
```

- Examples:
  $ `python M20AIE200.py 2 7 1 4 6 2 8 5 6`
  `4`
  $ `python M20AIE200.py 5 7 1 4 6 2 8 5 6`
  `6`
  $ `python M20AIE200.py 2 10 5 6 8 5 8 4 7 1 5 9`
  `5`
- To ensure the correctness and efficiency of your implementation, compare the time required for your implementaion against the time required by a naive method (sorting the array then returning the kth element). Repeat the comparison while increasing array size **(This exercise carries no marks and does not require any submission)**.
- Plagiarism of any form will result in 0 marks being awarded and result in disciplinary action being taken.