CSI3108-01

Programming HW#2

Max 50 points

Due on Sept. 21(Fri) 2018, by 5pm

[Part 1] Write a Java program to implement the Deterministic Selection Algorithm given below for finding the kth smallest element in an array of n elements. (30 pts)

Det_Select(k, n)
1. Divide S into n/a sets of a each;
2. Find the median in each set;
3. Recursively find the median x of the "baby" medians;
4. Partition S with x as pivot;   // like quicksort partitioning
5. Let i = rank(x);
   if ($k == i$)   return x;
   if ($k < i$)   return Det_Select(k, Lower part of array);
   else   return Det_Select(k-i, Upper part of array);

[Input]
The first line has the number of test cases. Then each test case consists of two lines. In the first line, there are three integers, k, n, and a. where $1 \leq k \leq n$. In the second line, there are n positive integers; each contiguous integers are separated by a blank. Assume that the maximum value of n is 1,000.

[Output]
For each case, first print the case number as '#x', where x is a case number, starting from 1 and then on the same line print the kth smallest element and the median of the baby medians obtained for the 1st iteration of the algorithm.

[Sample Input and Output]
Input

```
20        // total test cases
7 34 5    // k=7, n=34, and a=5, Test case #1
14 57 24 6 37 32 2 43 30 25 23 52 12 63 3 5 44 17 34 64 10 27 48 8 19 60 21 1 55 41 29 11 58 39  // 34 integers
…
```

Output

```
#110 30   // the 7th smallest element=10, the median of baby medians during the 1st call = 30
…         // Do NOT print comments
```

[Part 2] Implement the Randomized Selection algorithm discussed in the class with Java and then run the program with random inputs of enough sizes to compare the actual CPU times of both the randomized and deterministic selection algorithms. But you should determine a proper value of a for the deterministic algorithm through various experiments for the input; for example, a = 5, 7, 11, …. Write a report on the performance comparisons. (20 pts)