COM2031 Advanced Algorithms, Autumn Semester 2019

Lab 4: k-th smallest element

Purpose of the lab

Your task is to design and implement a recursive algorithm to compute the k-th smallest value in a set of integers.

You should work out the design of the algorithm before you start to implement it. In particular consider the following:

- What is the base case?
- How do you divide the problem into smaller problems?
- What is the "combine" part of the algorithm?
- What is the time complexity of your algorithm? (use the Master theorem)

Once you have worked these out then you can start coding. But if you start coding before you have worked these out then the task will be more difficult.

Lab Task 1: Find k-th smallest element in an array

Your task is to design a D&C algorithm for the following problem:

Let S be an unsorted set (array) of n integers.
Let k be an index between 1 and n : 1 <= k <= n.
The kth-smallest element of S is defined as the k-th element in Sort(S), the sorted version of S.

Your task is to write a Divide and Conquer algorithm for

```
public static int FindK(final int k, final int[] S)
```

which outputs the kth-smallest element of S and runs in average time O(n).

Hint: for Divide, use a pivot p (like in Quicksort) to split S into two parts S_left and S_right, where S_left is the set of elements less than p and S_right is the set of elements greater than p.

Consider the size of S_left. Let's call it *sl*

- If *sl* > *k*-1 then the k-th element must be in S_left
- If *sl* = *k-1* then the k-th element must be the pivot p (exactly k-1 elements less than p)
- If *sl* < *k-1* then the k-the element must be in S_right (BUT it won't be the k-th element – which element will it be?)

Therefore a recursive call of FindK is required only on S_left or S_right but not both (and possibly with a different value of k.)

What is the worst running time of the algorithm?

Lab Task 2: Find the median value in an array

Use inspiration from FindK to provide an O(n) algorithm to find the *median* of a set of integers S, defined as follows:

- If the size of S is odd (2n+1) then it is the middle value (n+1) when they are sorted
- If the size of S is even (2n) then it is the mean of the two middle values, i.e. at n and (n+1) when they are sorted.

In the case where the size of the set S is event, adapt your FindK algorithm with the pivot approach to a Divide and Conquer algorithm `FindKPair`

```
public static int[] FindKPair(final int k, final int[] S)
```

which finds both the k-th smallest and (k+1)th smallest values in S. As with FindK above it will be the Divide part of the algorithm that needs careful consideration of all the possibilities.