Programming Assignment 1

CSCD320 Algorithms

Eastern Washington University, Spokane, Washington

Please follow these rules strictly:

1. Verbal discussions with classmates are encouraged, but each student must independently write his/her own work, without referring to anybody else's solution.
2. No one should give his/her code to anyone else.
3. The deadline is sharp. Late submissions will **NOT** be accepted (it is set on the Canvas system). Send in whatever you have by the deadline.
4. Every source code file must have the author's name on the top.
5. All source code must be written in Java and commented reasonably well.
6. You are NOT allowed to use library-provided methods if you are asked to implement them.
7. Sharing any content of this assignment in any way with anyone who is not in this class of this quarter is NOT permitted.

The implementation of a simple but efficient randomized algorithm for finding the ith order statistic, using divide and conquer.

Definition 1. *The* ith order statistic *of an array A of n numbers is (1) the ith smallest number in A, if $i \leq n$; or (2) not defined, otherwise.*

Note that the given array A is arbitrary and may not be sorted. Also, array A may or may not have duplicates in its member elements. Below are a few examples:

- The first order statistic of $A[5, 4, 2, 7]$ is 2, because 2 the smallest number in the array A. Obviously the first order statistic is just the minimum of all the array elements.
- The nth order statistic of an array $A[0..n-1]$ is just the maximum of all the array elements.
- Both the third and the fourth order statistics of $A[5, 2, 4, 7, 4, 1]$ are the number 4, because 4 the third as well as the fourth smallest number in the array A.
- The seventh order statistic of $A[5, 2, 4, 7, 4, 1]$ does not exist, because array A has only 6 numbers.

Given the array A and the value of $i \leq n$, a **trivial solution** for finding the ith order statistic of A is: First, sort the array A. Then, we can directly pick the ith smallest number from the sorted array A. However, the sorting itself spends $\Omega(n \log n)$ time, making the whole algorithm's time complexity to be $\Omega(n \log n)$. **The goal of this project** is to implement an efficient algorithm that uses randomness and the divide-and-conquer strategy for finding the ith order statistic, whose average time cost is only $\Theta(n)$. The average-case linear time cost analysis of this algorithm is beyond the scope of this course, but the algorithm's behavior is implementable using the skills you already have.

The algorithm to be implemented

We start with the deterministic `partition` procedure that is used in the deterministic `QuickSort` algorithm that you have learned from the data structure class[1].

```
/* Partition A[left...right] using A[right] as the pivot.
   Return: the index of the array location that saves the pivot after partition.
*/
int partition(A,left,right)
{
   pivot = A[right];
   index = left;
   for(i = left ... right-1)
      if(A[i] <= pivot)
         swap(A[index],A[i]);
         index ++;
   swap(A[index],A[right])
   return index;
}
```

Now, we want to make the deterministic `partition` to be randomized by randomly picking an array element from $A[left..right]$ as the pivot. (Note that the Java library does provide random number generation function and look it up for its details on how to call it.)

```
/* Partition A[left..right] using a randomly picked element from
   A[left..right] as the pivot.

   Return: the index of the array location that saves the pivot after partition.
*/
int randomized_partition(A,left,right)
{
   i = random(left, right); //A number randomly picked from the range [left..right].
   swap(A[i],A[right]);
   return partition(A,left,right);
}
```

[1]Revisit your data structure material if you cannot recall it well.

Then, the randomized algorithm for finding the ith order statistic of any general working area $A[p..r]$ is as follows, where $i \leq r - p + 1$.

```
int RandomizedSelect(A,p,r,i)
{
   if(p==r)
      return A[p];

   q = randomized_partition(A,p,r); //A[q] is the pivot after partition
   k = q - p + 1;
   if(i==k)
      return A[q];
   else if (i<k)
      return RandomizedSelect(A,p,q-1,i)
   else
      return RandomizedSelect(A,q+1,r,i-k)
}
```

Therefore, in order to find the ith order statistic of the whole array, the initial function call will be `RandomizedSelect(A,0,n-1,i)`, where n is the array size and $i \leq n$. Note that we use 0-based indexing for arrays.

Before you do any coding, **your first job** is to go through the given algorithm and understand it is doing its job correctly, without worrying about its time cost.

Specification of your program

1. Your program should be named as: `OS_Finding.java`
2. The input and output of your program.

   There are two inputs to your program. One is a text file, where each line is an **integer** number. All the numbers in the text file may not be sorted. The data set stored in the input text file may or may not have duplicates. The file name should be supplied by the user to your program as a command line parameter. The second input is a **natural number**, representing the value of i, which should also be supplied by the user to your program as a command line parameter. Your program is going to look for the ith order statistic from the data set stored in the input text file.

   For example, if we supply $i = 1$ and the given file named `data.txt` has the following content:

   ```
   4
   8
   4
   3
   6
   9
   2
   ```

Then, the command line you type would be:

```
$java OS_Finding data.txt 1
```

and the output print on the screen should be:

```
$2
```

For the same input text file, if $i = 5$, then the command line you type would be:

```
$java OS_Finding data.txt 5
```

and the output print on the screen should be:

```
$6
```

For the same input text file, if $i = 9$, then the command line you type would be:

```
$java OS_Finding data.txt 9
```

and the output print on the screen should be:

```
$null
```

(Please note that the '$' symbol in the above example runs represents the terminal's command line prompt symbol and is not part of what your program prints on the screen.)

Submission

- All your work files must be saved in one folder, named: **firstname_lastname_EWUID_cscd320_prog1**
  (1) We use the underline '_' not the dash '-'.
  (2) All letters are in the lower case including your name's initial letters.
  (3) If you have middle name(s), you don't have to put them into the submission's filename.
  (4) If your name contains the dash symbol '-', you can keep them.
  (5) If you use IDE tool for programming, do not submit all auxiliary files from the IDE (for example: project file, package management file, etc). You only save and submit the `OS_Finding.java` file in the foler, and make sure that single java file works in command line.
- You then compress the above whole folder into a .zip file.
- Submit .zip file onto the Canvas system by the deadline.

Note

- You must implement all functions of the program by yourself except the part for random number generation. You can use a Java library function to generate random numbers.
- Your assignment will receive zero if you copy/borrow (even with changes) existing code from elsewhere.