Augustana Computing Science 310

Programming Assignment #1

Due: October 18, 2017

Your solutions to this assignment must be your own individual work, or you may work with one other person. If working as a team, submit your printed assignments together (and I will mark one), and your individual program on eClass.

Using sequential search to find the largest element in a list takes $n - 1$ key comparisons, where n is the size of the list. To find the 2nd largest element could be done in the same way, after removing the largest element, to take an extra $n - 2$ key comparisons. This would give a total of $2n - 3$ key comparisons. However, finding the 2nd largest element can be accomplished with only $\boldsymbol{n + \lceil \log_2 n \rceil - 2}$ key comparisons in the worst case. Your task for this assignment is to implement a Java program that accomplishes this minimal complexity.

To accomplish this reduction in complexity, you must save information from the first pass (finding the largest element), to use in finding the 2nd largest element. [A form of **dynamic programming**, where you are saving state information to reduce the complexity later on.] The information that you must save or create is a tournament tree with the largest on top (in the root). As you create the tournament tree, also create a hash map of winners together with all elements that lost to each winner [Those who have a Python background will recognize this as a dictionary. In Java, this is a **HashMap**.]

You may assume that all elements will be distinct (no duplicates) and that all elements will be valid integers. Your program must read the testing data from an input file called "data.txt". A sample file is located on eClass. Each line in the testing file contains one set of testing data: the first number indicates how many elements in the test and then all those elements follow on the same line.

For each line of testing, provide the following information:

1. The list of elements
2. Every key comparison that is done, and who won
3. The largest
4. The 2nd largest
5. The list of who lost to each winner.

For example:

```
=====================
9 2 1 7 5 10 6 4
Finding largest...
Comparison: 6 to 4  Winner: 6
Comparison: 5 to 10  Winner: 10
Comparison: 1 to 7  Winner: 7
Comparison: 9 to 2  Winner: 9
Comparison: 10 to 6  Winner: 10
Comparison: 9 to 7  Winner: 9
Comparison: 9 to 10  Winner: 10

Finding 2nd largest...
Comparison: 5 to 6  Winner: 6
Comparison: 6 to 9  Winner: 9

Winner is:  10
2nd place:  9
Results:
6 won against [4]
7 won against [1]
9 won against [2, 7]
10 won against [5, 6, 9]
=====================
```

You must implement your binary tree as an array (**ArrayList**). If you place your root at position 1, then the children are at position 2 and 3. In general, a node at position x has children at $2x$ and $2x + 1$.

Hand In Both:

1. A printed copy of your program (with documentation) and testing results on the "data.txt" file from eClass.
2. Submit a copy of your program on eClass.

I will mark all of the following: Java programming style, documentation (file and method headers, inline as necessary), correctness, and completeness (how far algorithm is completed and whether testing results have been included). <u>Your documentation should indicate how you have achieved your complexity bound</u>. Part of your mark may be determined by the knowledge you exhibit. I may ask you questions about your code, and your answers may be used in determining your mark.