CS 624: Analysis of Algorithms
Assignment 3
Due: Monday, Oct. 16 2023

1. Show how to find the second smallest element in an array of size n using \( n + \lceil \log n \rceil - 2 \) comparisons in the worst case. This is the exact number, not a big-O estimate. Hint: Also find the smallest element. Another hint – use a tournament like algorithm. The challenge is to understand exactly how it is done and why the number of comparisons is what it is.

2. Sorting the i largest numbers: Given a set of n numbers, we wish to find the i largest in sorted order using a comparison-based algorithm. Describe the worst case run time of each of the three algorithms below, both in terms of n and i.
   (a) Sort the numbers, and list the i largest.
   (b) Build a max-priority queue from the numbers, and call EXTRACT-MAX i times.
   (c) Use an order-statistic algorithm to find the \( i^{th} \) largest number, partition around that number, and sort the i largest numbers.

3. Exercise 1.1 in Lecture note 7 (binary search trees).

4. Exercise 1.2 in Lecture note 7 (binary search trees).

5. Exercise 1.3 in Lecture note 7 (binary search trees).

6. Exercise 1.4 in Lecture note 7 (binary search trees).

7. Exercise 1.5 in Lecture note 7 (binary search trees).

8. Exercise 1.6 in Lecture note 7 (binary search trees).

9. Exercise 2.2 in Lecture note 7 (binary search trees).

10. Exercise 4.1 in the Lecture 7 handout - you have to prove all these properties are correct (for many of the proofs - use contradiction) Please be careful. This is an exercise about binary search trees, not about algorithms used to construct those trees. So all you can use in doing this problem is the definition of a binary search tree. If you write something like, "this can’t happen because the algorithm would have placed this element somewhere else", then your reasoning can’t possibly be correct.