CSE 102 — Fall 2020 – Advanced Homework 6

Lower Bounds: Decision Trees and Adversary Arguments
(7 points+)

Attempt **ONE** of the following problems.

1. (easy: 2 points) Design an algorithm that uses $\lceil \frac{3n}{2} \rceil - 2$ comparisons to find the maximum and minimum of an array. [Hint: For an even array, pair up the elements and do $\frac{n}{2}$ comparisons, then find the largest of the winners, and separately, find the smallest of the losers. If n is odd, the last key may have to be considered among the winners and the losers.]

2. (medium: 3 points) Present an algorithm that uses $n + \lceil \log n \rceil - 2$ comparisons in the worst case to compute the second largest element of an array of distinct integers. [Hint: use a tournament method. Draw a diagram to illustrate your answers. Present your arguments clearly depending upon whether n is a power of 2 or not.]

   All questions below are hard problems. Assuming that you have looked up full solution from the internet/book sources cited below and no other assistance, you can earn up to 7 points. However, continue to use the AdvancedHWCitationGuidelines including the statements describing any help received from (or give to) any human beings or through discord. Extra Credit will be awarded if you are able to create a solution without help from any source.

3. Any binary tree with k leaves has an *average height* of at least $\log_2 k$.

   [Section 12.2 of BB[1]]

   Write the proof in detail using words, diagrams, induction and algebra that demonstrates that you have excellent understanding.

4. Any algorithm that uses comparison to find the second largest element in an array of n integers must do at least $n + \lceil \log n \rceil - 2$ comparisons in the worst case.

   [Section 5.3.3 of BV[2]]

   Write the proof in detail drawing tables to describe the adversary argument and an example (different than the one presented in the book) that brings out all the features of the adversary argument.

5. Any algorithm to find the median of n integers, for odd n, using comparisons, must do at least $\frac{3n}{2} - \frac{3}{2}$ comparisons in the worst case.

   [Section 5.5 of BV]

   Write the proof in detail drawing diagrams and tables to describe the adversary argument and an example (different than the one presented in the book) that brings out all the features of the adversary argument.

[1] Brassard and Bratley, Fundamentals of Algorithmics
[2] Basse and Van Gelder, Computer Algorithms

6. [This problem has been modified to provide greater clarity.]

   Let $T[1, ...n]$ be a sorted array of integers possibly including negative integers. *Integers need not be distinct*. We need to find an index i such that $1 \leq i \leq n$ and $T[i] = i$, provided an index exists. Only following types of comparisons are allowed: is $T[i] < j$.

   (a) Prove by a simple argument that a player can answer this question by asking $2n$ questions.

   (b) Find a better lower bound (or establish that none exists) using an adversary argument. Adversary strategy must provide the following: (i) adversary response to any question: is $T[i] < j$? where i is not necessarily equal to j, (ii) description of a strategy that the adversary will use to find consistent answers to prolong the game as long as possible. Observe that the lower bound will be a linear function of n in the worst case and (iii) describe how your strategy will allow the adversary to provide consistent answers.

   (c) Observe that if the sorted array consists of *distinct* integers, then there does exist an $O(\log n)$ algorithm similar to binary search algorithm. Explain clearly why and which part of your strategy will breakdown if integers were to be distinct.

   (d) Illustrate your strategy using an example (or a few examples) with a variety of questions that the player can ask and that your strategy will still work. Use the same set of questions on an example, where integers must be distinct, to describe at which step will the strategy break down and the player will be able to conclude correct answer without asking $O(n)$ questions.

7. Describe an algorithm to sort five elements using seven comparisons. Do not write code; use either tree diagrams with intermediate results or another drawing mechanism supported by arguments to describe and justify your algorithm.

   [Hint: Although it appears to be an easy problem, it is quite hard. No credit will be given just for code or simply the correct solution without justification or diagrams.]

8. Given an algorithm to find the median of five keys with only six comparisons in the worst case. Do not write code; either tree diagrams with intermediate results or another drawing mechanism supported by arguments to describe your algorithm.

   [Hint: Although it appears to be an easy problem, it is quite hard. No credit will be given just for code or simply the correct solution without justification or diagrams.]