Department of Computer Science **University of San Francisco**

Computer Science 673
Fall 2016
Homework 4: Sorting, Selecting, Trees
Due Friday, September 16th

All problem / exercise numbers are from the **3rd edition** of Introduction to Algorithms (the 1st and 2nd editions are different!) Note the difference between problems and exercises!

1. Problem 8-4

   Water Jugs

   Suppose you are given n red and n blue water jugs, all of different shapes and sizes. All red jugs hold different amounts of water, as do the blue ones. Moreover, for every red jug, there is a blue jug that holds the same amount of water, and vice versa.

   It is your task to find a grouping of the jugs into pairs of red and blue jugs that hold the same amount of water. To do so, you may perform the following operation: pick a pair of jugs in which one is red and one is blue, fill the red jug with water, and then pour the red jug into the blue jug. This operation will tell you whether the red or the blue jug can hold more water, or if they are of the same volume. Assume that such a comparison takes one time unit. Your goal is to find an algorithm that makes a minimum number of comparisons to determine the grouping. Remember that you may not directly compare two red jugs or two blue jugs.

   a (3 points) Describe a deterministic algorithm that uses $\Theta(n^2)$ comparisons to group the jugs into pairs.

   b (2 points) Prove a lower bound of $\Omega(n \lg n)$ for the number of comparisons an algorithm solving this problem must make.

   c (5 points). Give a randomized algorithm whose expected number of comparisons is $O(n \lg n)$. What is the worst-case number of comparisons for your algorithm?

      For part c, you do *not* need to prove that the expected number of comparisons is $O(n \lg n)$.

2. (5 points) Let $X[1 \ldots n]$ and $Y[1 \ldots n]$ be two arrays, each containing n numbers, already in sorted order. Give $O(\lg n)$-time algorithm to find the median of all $2n$ elements in arrays X and Y.

3. Special Cases of Select:

   (a) (4 points) Give an algorithm to find the second smallest element in a list of n elements in at most $n + \lceil \lg n \rceil - 2$ comparisons. *Note:* Your algorithm can use no more than $n + \lceil \lg n \rceil - 2$ total comparisons in the worst case. *Hint:* Find the smallest element, too.

(b) (4 points) Give an algorithm to find the 3rd smallest element in a list, using the smallest number of worst-case comparisons. Exactly how many comparisons does your algorithm take in the worst case? **Hint:** Use your solution to part a — you will need to find the smallest and second-smallest element as well!