CS 577: Introduction to Algorithms — Homework 1

Out: 01/27/16 **Due: 02/03/16**

Ground Rules

- This homework is for both sections of CS 577.
- The homework consists of two questions. You are required to turn in detailed solutions for both. We will pick one question at random to grade.
- You will be graded on the correctness as well as clarity of your solutions. You are required to prove any claims that you make. In particular, when you are asked to design an algorithm, you must argue its correctness as well as running time. You may use without proof any theorems proven in class or in the textbook, as long as you state and cite them properly. Please refrain from consulting sources other than the text and lecture notes.
- You are required to work and submit solutions in pairs, unless you have prior permission from the instructors.
- Homework is due promptly at the start of class. Late homework will receive zero credit.
- Start working on your homework early. Plan your work in such a way that you have the opportunity to put some problems on the back burner for a while and revisit them later. Good luck!

Problems

1. The Tres Chique Cinema is so small that there is only one row of n seats. Furthermore, any time a patron is brought to his or her assigned seat, any patrons already seated next to them must rise and sit again, as must any next to them, and so forth. For example, if there are patrons in seats 1, 2, 4, and 6, and someone sits in 3, the patrons in 2 and 4 must rise and sit again, which in turn requires the patron in 1 to rise and sit again; so seating the new arrival in this case causes three additional reseatings.

   As the owner of Tres Chique, you would like to minimize the total number of seatings and reseatings necessary to seat an entire row of n patrons. Determine the order in which patrons should be brought to their seats so as to require only $O(n \log n)$ seatings and reseatings. Write a recurrence for the total number of (re)seatings and solve it. You may assume that $n = 2^k - 1$ for some integer $k \geq 0$.

   (Hint: Think about which patron you should seat last, and use recursion!)

2. You are given $n = 2^k$ coins, all of which look identical. However, one of the coins is defective – it weighs either slightly more or slightly less than the rest (you don't know which). You also have at your disposal an electronic equivalence tester, which has two compartments. You may place any set of objects in each compartment; the tester tells you whether or not the two sets weigh the same. Note that the tester does not tell you which side is heavier and which one lighter – only whether they weigh the same or not.

   Your goal is to use the tester to determine which of the coins is defective. You may assume that $k > 1$. (Is it possible to find the defective coin when $k = 1$?)

   Give an algorithm for finding the defective coin that uses the tester at most k times. Prove the correctness of your algorithm. Note that you are allowed to use the tester only k times, not $O(k)$ or k plus a few more times. You will receive partial credit for slightly worse solutions.