CS 577: Introduction to Algorithms — Homework 1

Out: 01/27/17 — **Due: 02/03/17**

Ground Rules

- Review problems will be discussed at the following week's review.
- Self-graded problems will be discussed at the following week's review, and are to be self-graded by you during the review. Only students present at the review session and turning in their graded solutions in person will earn points.
- Graded problems are to be turned in on the due date at the beginning of the lecture.
- Self-graded problems should be done individually.
- Graded problems should be done in pairs.
- Please follow page limits for all self-graded and graded problems.
- Write your name(s) clearly on your submissions, and turn in each problem on **a separate sheet of paper**.

Review Problems

1. You are a visitor at a political convention with n delegates; each delegate is a member of exactly one political party. It is impossible to tell which political party any delegate belongs to; in particular, you will be summarily ejected from the convention if you ask. However, you can determine whether any pair of delegates belong to the same party or not simply by introducing them to each other—members of the same party always greet each other with smiles and friendly handshakes; members of different parties always greet each other with angry stares and insults.

   Suppose that strictly more than half of the delegates belong to the same political party. Describe an algorithm that identifies all members of this majority party using at most $O(n \log n)$ introductions. Prove the correctness of your algorithm and analyze its running time (in terms of the number of introductions). Can you achieve a running time of $O(n)$?

   (Hint: Try to first identify a single member of the majority party.)

2. In this problem, you are given a binary tree with a designated root and your goal is to find a large subtree in this tree that is a complete binary tree. For this problem, a subtree of a binary tree means any connected subgraph; the root of such a subtree is defined to be the unique vertex that is closest to the root of the original tree (in other words, the vertex at the highest level). A subtree is complete if (1) all of its internal nodes other than the root have degree three and the root has degree two (in other words, every non-leaf node has two children); (2) every leaf has exactly the same depth. (See figure on the following page.)

   Describe and analyze a recursive algorithm to compute the largest complete subtree of a given rooted binary tree. Your algorithm should run in time linear in the number of nodes in the tree, and should return the root and the depth of the subtree.

Self-graded Problems

3. **(Page limit: 1 sheet; 2 sides)** You are given two sorted lists A and B with n distinct numbers in each. Develop a divide and conquer algorithm for finding the median[1] of the union of the lists in $O(\log n)$ time. Note that merging the lists would take $\Theta(n)$ time, so that would not be a good idea.

[1] We define the median of a list of length m, where m is even, as the $m/2$th smallest element.

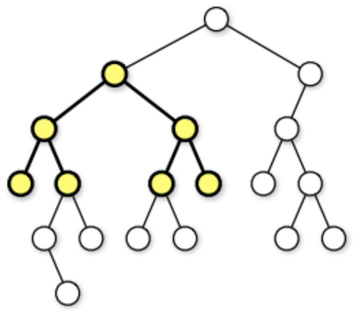

Figure 1: The largest complete subtree of this tree is shown in yellow.

Extend your algorithm to finding the kth smallest element in the union of the two lists, where k is some integer in $\{1, \cdots, 2n\}$.

Graded Problems

4. **(Page limit: 1 sheet; 2 sides)** In a variant of the Towers of Hanoi puzzle, the three towers are labeled 0, 1, and 2, respectively. You are only allowed to move disks from a tower labeled i to the next one labeled $(i + 1) \bmod 3$. You can imagine the towers as being arranged in a circular fashion, in which case, the constraint says that you can only move disks in an anti-clockwise manner. So, for example, in order to move a disk from tower 0 to tower 2, you would have to first move it to tower 1, if both $0 \to 1$ and $1 \to 2$ are legal moves, and this sequence would cost two moves instead of just one.

   (a) Develop a divide and conquer algorithm for moving n disks from tower 0 to tower 2.

   (b) Analyze the asymptotic number of moves your algorithm makes.

5. **(Page limit: 1 sheet; 2 sides)** 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 determine which of the coins is defective using the tester at most k times. You may assume that $k > 1$, that is, $n > 2$. (Is it possible to find the defective coin when $n = 2$?) 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.

   (a) Describe a divide and conquer algorithm for this problem.

   (b) Argue that your algorithm uses the tester at most k times.

   (c) Prove the correctness of your algorithm, in other words that it always correctly identifies the defective coin.