# CS 577: Introduction to Algorithms — Homework 1

**Out: 02/07/12** — **Due: 02/14/12**

## Ground Rules

- **Grading.** 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.
- **Collaboration.** You may work and submit solutions in pairs.
- **Lateness.** Homework is due promptly at the start of class. Late homework will receive zero credit.
- **Extra credit questions.** Extra credit questions will not directly contribute to your score. However, they will be taken into account in the final grading and may improve your overall grade if your total score is close to the boundary between two grades. Furthermore they are fun to solve and improve your understanding of the course.
- 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. **(6 points)** Solve the following recurrences. (If you cannot find an exact answer, give the best upper and lower bounds that you can.) In each case you may take $T(1) = 1$.
   (a) $T(n) = 3T(n/2) + n^3$
   (b) $T(n) = 2T(n-3) + 1$
   (c) $T(n) = T(n/2) + \log n$
2. **(4 points.)** You are given an unsorted list of $n$ integers, all in the range $[1 \cdots m]$. Give an algorithm for sorting this list in time $O(n+m)$. (Note that for small $m$ this is asymptotically better than $O(n \log n)$.)
3. **(5 points.)** Problem 5.3 in the textbook (Pg. 246–247).

   **(Extra credit.)** Give an $O(n)$ time algorithm for Problem 3. (Hint: In $O(n)$ time can you throw away at least half the cards such that if there is a "majority element" in the original collection, that element continues to form a majority in the remaining collection?)

Note: Since a few people in class do not have the textbook yet, we have reproduced the questions from the book on the back of this page.

## Kleinberg and Tardos 5.3

Suppose you're consulting for a bank that's concerned about fraud detection, and they come to you with the following problem. They have a collection of $n$ bank cards that they've confiscated, suspecting them of being used in fraud. Each bank card is a small plastic object, containing a magnetic stripe with some encrypted data, and it corresponds to a unique account in the bank. Each account can have many bank cards corresponding to it, and we'll say that two bank cards are *equivalent* if they correspond to the same account.

It's very difficult to read the account number off a bank card directly, but the bank has a high-tech "equivalence tester" that takes two bank cards and, after performing some computations, determines whether they are equivalent.

Their question is the following: among the collection of $n$ cards, is there a set of more than $n/2$ of them that are all equivalent to one another? Assume that the only feasible operations you can do with the cards are to pick two of them and plug them in to the equivalence tester. Show how to decide the answer to their question with only $O(n \log n)$ invocations of the equivalence tester.