CS 577: Introduction to Algorithms 9/13/2012

Homework 1

Instructors: Shuchi Chawla and Dieter van Melkebeek Due: 9/27/2012

This assignment covers divide-and-conquer (chapter 5 in the textbook) and algorithmic graph primitives (chapter 3 in the textbook).

Guidelines

- Assignments consist of ungraded problems, graded problems, and extra-credit problems.

  You should work on all the non-extra-credit problems, but you only need to spell out and turn in your solutions for the graded ones. We will provide model solutions to the graded as well as the ungraded problems.

  You are encouraged to work on the extra-credit problems and to turn in your solutions for them. They will not directly contribute to your score but they will be marked and taken into account for your final grade.

- You will be graded on the correctness as well as the clarity of your solutions. You are expected to prove any claims that you make. In particular, when you are asked to design an algorithm, you should argue its correctness as well as its running time. You may use without proof any results covered in class, as long as you state and cite them properly.

- You may work and submit solutions in pairs. Please refrain from consulting sources other than the text and lecture notes.

- Please start a new sheet of paper for each problem, and clearly write your name(s) on each sheet. This is because different problems may be graded by different TAs.

- Homework is due promptly at the start of class. Late homework will receive no 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!

Ungraded problems

1. Problem 5.2 in the textbook (p. 246).

2. Given a sorted array of distinct integers $A[1, \cdots, n]$, you want to find out whether there exists an index i for which $A[i] = i$. Give a divide-and-conquer algorithm that runs in time $O(\log n)$.

3. You are given an unsorted array of n integers, all in the range $[1 \cdots m]$. Give an algorithm for sorting this array in time $O(n + m)$. Note that for small m this is linear time. Why doesn't the $\Omega(n \log n)$ lower bound apply in this case?

4. In class we described an algorithm that multiplies two n-bit binary integers x and y in time $O(n^{\log_2 3})$. Call this subroutine `fastmultiply`(x, y).

   In this problem we employ this subroutine to efficiently convert any decimal integer x with n digits into binary. The following is an outline for an algorithm.

```
function dec2bin(x)
  if n = 1: return binary[x]
  else:
    split x into two decimal numbers x_L, x_R with n/2 digits each
    return ???
```

   Here `binary`$[\cdot]$ is an array that contains the binary representation of all one-digit integers. That is, `binary`$[0] = 0$, `binary`$[1] = 1$, up to `binary`$[9] = 1001$.

   Fill in the missing details in the algorithm. Assume that a lookup in `binary` takes $O(1)$ time. Determine the running time of the algorithm.

Graded problems

5. (5 points) Problem 5.3 in the textbook (p. 246–247).

6. (5 points) Problem 5.7 in the textbook (p. 248–249).

7. (5 points) Problem 3.10 in the textbook (p. 110–111).

Extra-credit problems

8. Give an $O(n)$ time algorithm for problem 5.

   A correct solution for this problem will also count towards the credit for problem 5.

9. The following problem is very challenging. We won't hand out model solutions for it so you can submit a solution till the end of the course.

   You are given n coins, at least one of which is bad. All the good coins weigh the same, and all the bad coins weigh the same. The bad coins are lighter than the good coins.

   Find the exact number of bad coins by making $O(\log^2 n)$ weighings on a balance. Each weighing tells you whether the total weight of the coins you put on the left side of the balance is smaller than, equal to, or larger than the total weight of the coins you put on the right side.