CS 577: Introduction to Algorithms — Homework 2

Out: 09/26/06 — **Due: 10/10/06**

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 and running time.
- Collaboration. You are allowed to discuss questions with other people in the class. However, **you must solve and write your answers yourself without any help**. You must also give explicit citations to any sources besides the textbook and class notes, including discussions with classmates. Solutions taken from external sources such as the WWW, even if cited, will receive no credit unless there is significant "value added". In cases of doubt, you may be asked to explain your answer to the instructor and this will determine your grade.
- Lateness. Please see the class webpage for details on the lateness policy.
- 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. **(Minimum Vertex Cover in Trees.)** Let G be a graph with vertex set V and edge set E. A vertex cover of G is a subset of V, V', such that *every* edge in E is incident on some vertex in V'. The first three figures on the next page show graphs with vertex covers – each vertex in the vertex cover is circled. You should verify that each of the figures gives a valid vertex cover.

   In the minimum vertex cover problem, our goal is to find a vertex cover of a given graph with the fewest number of nodes. Figures (i) and (ii) on the next page display two different vertex covers for the same graph; the one on the left (i) has more nodes than the one on the right (ii); note that the latter is the minimum vertex cover for the graph.

   (a) Find the minimum vertex cover for the trees in figures (iv), (v), (vi) and (vii) on the next page.

   (b) Give a linear-time greedy algorithm that, given a tree, finds the minimum vertex cover in that tree. (A slightly worse running time, like $O(n \log n)$ or $O(n^2)$ will receive partial credit.)

2. Problem 4.14 in the textbook (p. 195).

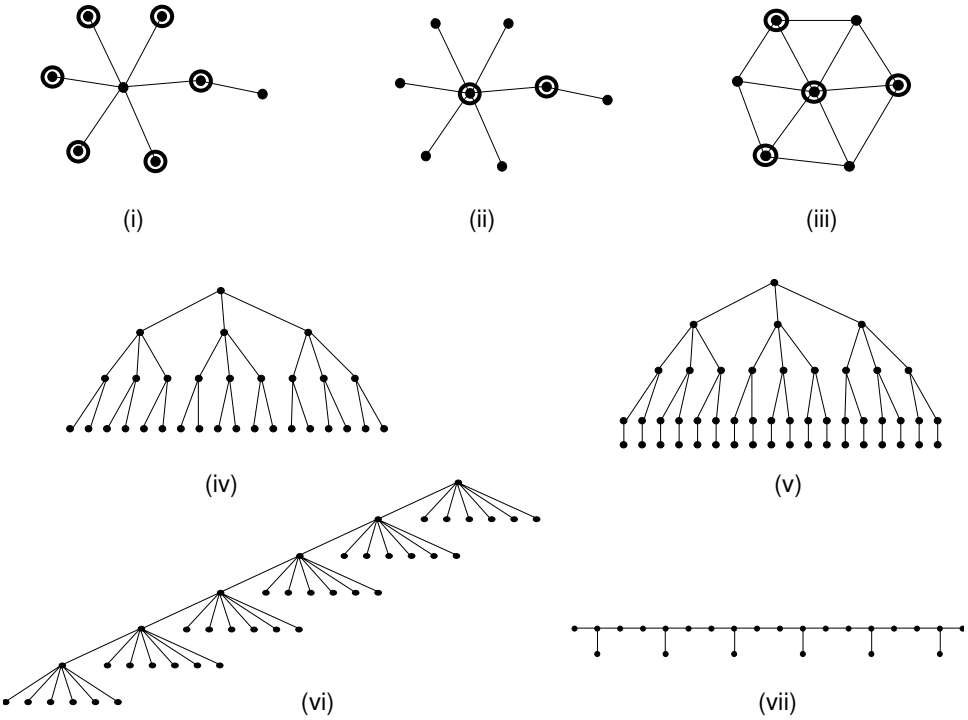
3. **(Median of two sorted arrays.)** Let A and B be two sorted arrays of n elements each. We can easily find the median element in A — it is just the element in the middle — and similarly we can easily find the median element in B. (Let us define the median of $2k$ elements as the element that is greater than $k-1$ elements and less than k elements.) However, suppose that we want to find the median element overall — i.e., the nth smallest in the *union* of A and B. How quickly can we do that? You may assume that there are no duplicate elements.

   Your job is to give tight upper and lower bounds for this problem. Specifically, find a function $f(n)$ satisfying the following:

   (a) Give an algorithm whose running time (measured in the number of comparisons) is $O(f(n))$, and

   (b) Give a lower bound showing that any comparison-based algorithm must make $\Omega(f(n))$ comparisons in the worst case.

   Can you get rid of the O and Ω to make your bounds *exactly* tight in terms of the number of comparisons needed for this problem?



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

   Extra Credit: Give an $O(n)$ time algorithm for the same problem. (**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?)

5. Problem 5.4 in the textbook (p. 247–248).

6. Problem 5.5 in the textbook (p. 248).