CSE 331: Introduction to Algorithm Analysis and Design Fall 2009

Homework 8

Due Friday, November 13, 2009 at the beginning of class

For general homework policies and our suggestions, please see the policy document.
Do not forget to look at Jeff's grading rubric before you write down your solutions.
No collaboration is allowed on problem 1(a).
Starting with homework, I'll put in some "extra" problems, which you should not turn in (they'll not be graded). However, it is recommended that you work on those problems for practice.

---

1. ($20 + 20 = 40$ points)

   (a) (**Your must work on this problem on your own: NO collaboration is allowed**) Exercise 1 in Chapter 4.

   (b) Part (a) of Exercise 2 in Chapter 4.

   Hint: The solutions to both the problems above are short and simple. It might help to think algorithmically about these problems.

2. ($40 + 20 = 60$ points) In class we have seem algorithms that compute the MST in time $O(m \log n)$. This is under the assumption that the graph is given it its adjacency list representation. This of course is not as best as possible[1].

   In this problem you will explore how to implement Prim's MST algorithm if the graph is given in its adjacency matrix form. (The adjacency matrix for a weighted graph is the natural generalization of the unweighted one we have seen in class earlier. In particular, the entry for (i, j) in the matrix is a 0 if (i, j) is not an edge, otherwise it is $c_{(i,j)}$.)

   (a) Show how to implement Prim's algorithm in $O(n^2)$ time if the input graph is given as an adjacency matrix.
   Hint: Look at the implementation of Prim's algorithm in the book and try to think how having an adjacency matrix might help you maintain the critical quantity you need to keep track of (for each vertex). You will have to maintain some auxiliary data structure(s) (but simpler one(s) than the one used in the book's implementation).

   (b) Argue that your algorithm above is the best possible: i.e., *no* algorithm can have a faster asymptotic running time.
   Hint: Recall that the running time of an algorithm has to include the time it takes to write its output (remember problem 1(d) on the mid-term) and the time to read its input.

3. (**DO NOT hand this problem in**) Exercise 8 in Chapter 4.

   Hint: The solution I have in mind uses the property of MST that we *did not* cover in class.

[1]The best known running time for an MST algorithm is $O(m \cdot \alpha(n, m))$, where $\alpha(\cdot, \cdot)$ is the inverse Ackermann function and is a a very slooooooooowly growing function– for all practical input values, the function value is smaller than (say) 5.