CPSC-320 Intermediate Algorithm Design and Analysis Winter 2000

Homework # 3

Due in class on Wednesday, October 25.

Reminder: while you are encouraged to work with others on the homework problems, you MUST write up your solutions on your own.

In this homework, you need to describe both algorithms and explanations of why they are correct. Describe your algorithms clearly, using either full sentences in English, pseudocode, or a combination. When explaining correctness, you should use full sentences - either English or mathematical notation or both is fine. Answers do not have to be long, but should be clear and nicely presented. A part of the mark for the problems will be based on the quality of the presentation.

1. An undirected graph $G = (V, E)$ is *bipartite* if V can be partitioned into two sets V_1 and V_2 so that every edge in E joins a vertex of V_1 to a vertex of V_2. (Thus there is no edge between any pair of vertices in V_1 or between any pair of vertices in V_2).

   a) Give an example of a bipartite graph and an example of a graph that is not bipartite.

   b) Describe a linear time algorithm to test if a graph is bipartite.

   c) Give a short and clear explanation of why your algorithm is correct.

2. Let A and B be two sets, each with n elements from some ordered set, such that A resides in computer P and B in Q. P and Q can communicate by sending messages, and they can perform any kind of local computation.

   a) Design an algorithm to find the nth smallest element of the union of A and B. You can assume that all the elements are distinct. Your goal is to minimize the number of messages, where a message can contain one element or one integer. (Note that computations such as sorting of the elements in A on computer P or the elements of B on computer Q do not add to the message count.)

   b) What is the number of messages sent in the worst case? Explain your answer.

3. Consider the following algorithm for the single source shortest paths problem when the graph may have negative weights (but you can assume that the graph has no negative cycles). The algorithm simply finds the smallest edge weight, say $-w$, adds w to every edge weight, and then uses Dijkstra's algorithm for solving the shortest path problem on the modified graph. The output is the shortest path tree output by Dijkstra's algorithm, (with the edge weights as in the original input).

   Do you think this algorithm correctly computes the shortest path tree? Why or why not?

4. An *arborescence* of a directed graph is a rooted tree such that there is a directed path from the root to every other vertex in the graph.

   a) Give a polynomial time algorithm to test whether G contains an arborescence.

   b) Give a bound on the running time of your algorithm, using O-notation. Explain your answer.