Problem Set 4—Due Friday, May 18 3:15PM

(20) Problem 1. Problem 22.4-5 on page 615.

(25) Problem 2. Suppose we are given an m arc, n vertex directed graph and we want to be able to quickly answer questions of the form: what is the shortest path (in terms of number of edges) from i to j? for an arbitrary pair of vertices i, j (we return either the sequence of vertices to visit, or we return NONE if there is no such path). We expect to be answering many such queries, so we will first spend some time processing the graph and storing information so we can then quickly answer any such query. As a baseline, we could plan to construct an $n \times n$ table P where each entry $P[i, j]$ is a pointer to a linked list of the vertices on the shortest path. Note that once the table is built, it take to $\Theta(k)$ time to answer a query for a given i, j that has a best path with k edges.

A How could we construct such a table $P[i, j]$ efficiently? What is the time and space to build it?

B Suppose we still want to we able to answer queries in time $\Theta(k)$, but we want to reduce the space used by the baseline solution. Describe a solution that uses $O(n^2)$ space and give the pre-processing time.

C Now suppose we have an undirected graph and we just want to be able to answer queries of the form: is there a path from u to v? Describe an efficient algorithm to preprocess the graph to quickly answer such queries. A good answer would have fast preprocessing time, use little space, and answer such queries in $O(1)$ time.

(15) Problem 3. Suppose you are given a C program and you can determine for each function, the set of functions which it calls (thus,e.g., you are told Function1 calls 2 and 3, 3 calls 4 and 5, ...). Describe how you can efficiently determine if your C program is recursive (a recursive program has the possibility that one of its functions can call itself, possibly indirectly. Thus if A calls B which calls C which calls A, we have recursion).

Problem 4 (20) Suppose we want to find a Minimum Spanning Tree subject to some additional constraints. For the two settings below, describe how to efficiently find such an MST and give the run time of your solution.

A We have a specific arc (u, v) that must be included (so we want the cheapest spanning tree that includes this arc).

B We have two arcs (u, v) and (x, y) and exactly one of these two arcs must be included in our tree (so we can't use both, but we must use one of them).

Problem 5 (25) Suppose you have a directed graph G with edge weights (distances) . You want to find the shortest path from an initial vertex s to all other vertices, but you have the additional constraint that you are allowed to use at most three arcs on the path.

Describe an efficient algorithm to find the shortest path (note: not just its cost but the actual sequence of vertices). Give the running time of your solution. Hint: start by finding the shortest path to all vertices using one arc.