Problem Set 3—Due Friday, February 23, 3PM

(30) Problem 1. The *yes-no, s-t long path* problem is: given a directed graph G=(V,E) with edge weights $d(i,j)$, two vertices s,t and a target distance k; is there a simple-path from s to t of length at least k? A *simple-path* is a path that never visits a vertex twice (thus the path has no cycles).

a) Show that the yes-no s-t long path problem is in NP (note, this problem is NP-C but you are NOT being asked to prove that).

b) Show that you can use a program that solves the yes/no s-t long path problem to find the *length* of the longest simple path from s to t? You should find the longest possible length using a polynomial number of calls to the yes/no s-t long path routine, plus polynomial additional work. Thus you will show that the optimization problem (find the largest length) is polynomial time reducible to the yes-no problem.

c) Show that you can use a program which solves the yes/no s-t long path problem to actually find a simple-path from s to t of length at least k (when one exists). You should find the path using a polynomial number of calls to the yes/no s-t long path routine, plus polynomial additional work. Thus you are showing that the problem of finding an s-t long path of a minimum size is polynomially reducible to yes/no s-t long path.

(30) Problem 2. Consider the *similar Path* problem (SP):
INPUT: (As in problem 1) a directed graph G=(V,E) with edge weights $d(i,j)$, and two vertices s,t.
GOAL: find two edge-disjoint paths from s to t such that the lengths (sum of edge weights) of the two paths are the same (this comes up in network routing when we want to avoid jitter: we send data along two paths and want them to arrive at (about) the same time so we can combine them more easily). We want to show that this problem is NP-complete.

a) Show that this problem is in NP.

b) Consider the following proposed proof to show that SP is NP-hard (which with a) will show it is NP-C). We use directed hamilton s,t path (HP s,t) defined as follows:
INPUT: a directed graph G, $s,t \in V$
GOAL: a simple-path from s to t that visits all vertices.

We reduce HP s,t to SP as follows: given an input for HP s,t : a directed graph G, s,t (note: no edge weights), we solve this input by converting to a new graph G' which is the same except it has a new direct edge from s to t of weight $n-1$, and all the original edges in G are given weight 1. We give this to our SP solver and get two equal paths or "no solution". Thus whenever there is an s,t hamilton path in G, we will get two edge disjoint s,t paths of length $n-1$ in G': the original hamilton path in G, and the direct path (s,t). Further, if there is no s,t hamilton path in G, we can't find a second path to match the length $n-1$ direct arc path (s,t).

This proof unfortunately doesn't quite work. Where does it fail (be as precise as possible)?

c) Fix the proof of part b) by modifying the construction of G' a little.

(15) Problem 3. Suppose we are given a *Traveling Sales Man* (TSP) problem where cities are points in the plane and distances are actual Euclidean distances (Call this the Euclidean TSP).

Show that the *proof* of theorem 8.18 in KT on page 479 (34.14 p. 1013 in CLRS) in the text does **not** prove that the Euclidean TSP is NP-hard (this problem is in fact still NP-hard, but you are not asked to prove this).

(15) Problem 4 Estimate /calculate the time to do one iteration of the loop below on a fast PC (or a CSIF PC) when compiled using high optimization (-O4 in C). This time you are allowed to use some timing experiments to help. Assume n is at least $1,000$. Describe how you got your answer (including any timing results you got).

```
for (i=2; i <= n ; i++)
        if (n%i == 0) // bottleneck **
          return 0;
  return 1;
```