Introduction to Algorithms **Problem Set 7**
CS 482, Spring 2008 **Due Wednesday, April 2**

(1) Suppose you and your $k-1$ housemates decide to throw a party. Unfortunately, there seems to be little agreement on who should be invited, and you don't want to invite more people than can fit in your house. Since you're the only one in the house who can get the DVD player to work, the task of resolving this problem has been left to you. In total, your house can hold n people (in addition to you and your housemates). Each housemate i gives you a list P_i of people she would like to have invited to the party. Depending on how much you like housemate i, you pick an integer m_i indicating the minimum number of people on i's list that you'll invite.

Having selected these values m_i already, you'd like to know whether it is possible to invite at most n people to your party such that for each housemate i, at least m_i of the people in P_i are invited. We call this the *Party Invitation* problem.

Prove that Party Invitation is NP-Complete.

[Acknowledgement: This question was composed by Tom Wexler, who taught CS 482 in Spring 2006.]

(2) Some of your friends are co-founders of UnfortuNet, an Internet service provider that had a promising business plan but keeps running up against NP-Complete problems in their day-to-day operations. Lately they've been talking to you about designing a high-speed network joining together a set of routers, V. For each pair of routers u, v, there's a positive integer $c(u, v)$ that denotes the cost of creating a direct link from u to v. (These costs are symmetric, i.e. $c(u, v) = c(v, u)$.) Their goal is to choose an edge set E such that the graph (V, E) is connected, and the combined cost of the edges in E is less than some fixed budget, B.

"That's great," you say, "this is the minimum spanning tree problem! There's a greedy algorithm you can use to compute the cheapest set of edges joining your routers together."

"Not so fast," says one of your friends. "Each of our routers has only 32 interfaces, so we can join it to at most 32 other routers." Thus, the problem your friends really need to solve could be called *Minimum Spanning Tree with Hardware Limitations (MSTwHL)*: given a positive integer budget B, a set of routers V, and positive integer costs $c(u, v)$ for every pair $u \neq v$ in V, satisfying the symmetry property $c(u, v) = c(v, u)$, construct an edge set E such that:

1. Each router $v \in V$ belongs to at most 32 edges in E.
2. The sum of the costs of all edges in E is at most B.

Prove that Minimum Spanning Tree with Hardware Limitations is NP-Complete.

(3) Undaunted by the NP-Completeness of the MSTwHL problem, the engineers at UnfortuNet went ahead and built their high-speed network. In fact, rather than just building a spanning tree they built a big undirected graph $G = (V, E)$ on the set of n routers. Lately they've been experimenting with some new routing software, but unfortunately it's incompatible with the old version of the protocol. Thus, they want to subdivide their network into a "live network" running the old software, and a "test network" running the new version. Here are the constraints they face.

1. There's one particular router, v_1, that **must** belong to the live network.
2. There's another particular router, v_2, that **must** belong to the test network.
3. There's an integer $k > 0$, such that the test network must contain exactly k routers. All others will belong to the live network.
4. The live network L and the test network T must both be connected subgraphs of G. In other words, every two routers in L must be joined in G by a path consisting only of routers in L, and every two routers in T must be joined in G by a path consisting only of routers in T.

The *Test Network Design* problem is the following computational problem: given an undirected graph G, two nodes v_1, v_2, and a desired test network size k, find a live network L and test network T which satisfy (1)-(4).

Prove that Test Network Design is NP-Complete.