Please hand in each problem on separate sheets with your name and netID on each. If a problem requires multiple sheets, please staple the sheets for that problem together.


**Question 1**
Show that for any fixed constant \( k \geq 2 \), determining whether or not an undirected graph \( G \) has a spanning tree \( T \) with exactly \( k \) leaves is NP-Complete.

**Question 2**
Suppose we have an undirected graph \( G = (V, E) \) with edges costs \( c_e \geq 0 \) for each edge \( e \in E \), and a set of nodes \( S \subseteq V \). Done. Recall that a Steiner tree for \( S \) is a tree \( T \) in \( G \) that spans all nodes in \( S \). As you’ve seen in class, in general, finding the minimum cost Steiner tree is NP-Complete.

(a) Give a polynomial time algorithm to find the minimum cost Steiner tree in the case that \( |S| = 3 \).

(b) Give a 2-approximation for the problem of finding a minimum cost Steiner tree (for any set \( S \)).

**Question 3**
Imagine you’re planning an end-of-the semester party, and are trying to decide which of your \( n \) friends to invite. This time, the limit is not the size of your house, since you’ll be outside. Instead, the problem is that some of your friends just don’t get along, and inviting any two who don’t get along is a recipe for disaster. Luckily, it turns out each of your friends has a problem with at most three of your other friends.

In deciding who to invite, you assign each friend \( i \) an awesomeness factor \( a_i \). (If you’re not the sort of person who assigns numerical values to your friends, that’s cool, just pretend.) Your goal is to invite a set of friends \( S \) so as to maximize \( A = \sum_{i \in S} a_i \), the sum total awesomeness of all friends you do invite, without inviting any two people who don’t get along.

Give a 1/3 approximation algorithm for this problem.

**Bonus Question (Optional*)**

*(tricky, not worth much and no partial credit; basically just for fun if you’re bored done with the rest of the problems)*

In class we saw a 2-approximation for minimizing the makespan in a load balancing problem. That is, we gave an algorithm to schedule \( n \) jobs with lengths \( t_1, t_2, \ldots, t_n \) on \( m \) machines such that the latest completion time is at most twice the minimum possible such value. Another natural objective would be to try to minimize the average completion time of all jobs. (For one machine, we know how to do this by simply scheduling the shortest jobs first, but things get a bit trickier with \( m \) machines.)
Now suppose we wanted to satisfy both objectives; we’d like to minimize the makespan and the average completion time.

(a) Give an example with 4 jobs and 2 machines for which no solution minimizes both objectives simultaneously.

(b) Prove that in general, there always exists a solution that is a 2-approximation to both objectives.