Name: ID:

ICS 163 — Graph Algorithms — Winter 1994 — Midterm

1. Let G be the graph drawn below, with the indicated edge capacities.

   a. Treating the capacities as lengths, find a minimum spanning tree in G.

   b. Show a sequence of augmenting paths found by the shortest augmenting path algorithm for maximum flow from s to t, and the capacities of each path.

   c. Show a sequence of augmenting paths found by the maximum capacity augmenting path algorithm, and the capacities of each path.

   d. Find a minimum cut separating s from t in G.

Name: ID:

2. Describe a graph for which the greedy algorithm does not compute the maximum matching. (The greedy algorithm would consider each edge in some order, including the edge in the matching if possible, or throwing out the edge if one of its endpoints is already matched. You may choose which order the edges are considered in.)

Name: ID:

3. Let G be a bipartite graph with property P_k:

$$P_k: \text{ every vertex in } G \text{ is incident to exactly } 2^k \text{ edges}$$

For example, a four-dimensional hypercube is a bipartite graph in which every vertex is incident to four edges, so it has property P_2. A complete bipartite graph with four vertices on each side also has property P_2. But in a three-dimensional cube every vertex is incident to three edges, so it does not have property P_k for any k.

   a. Show how to use an Euler tour to construct a subgraph G' of G with property P_{k-1}.

   b. Part (a) proves that G must always have a subgraph with property P_{k-1}. Use this fact to prove that G must always have a perfect matching.

   c. How long does it take to compute a perfect matching using this method?

   d. Describe a non-bipartite graph, having property P_k for some k, but not having any perfect matching.

Name: ID:

4. **(Baseball Elimination)**

Suppose there are n teams in major league baseball, After the season is most of the way through, m games remain to be played. (Denote by g_{ij} a game scheduled between certain teams t_i and t_j.) Assume that each game ends in a win or a loss (no ties). We wish to know whether it is still possible for the Angels to end up as the best team in the league.

The result will depend not only on how the Angels play (in the optimal case, they will win all their games) but also on how well the other teams play. If team t_i is a_i games ahead of the Angels, that team must lose at least a_i games if the Angels are going to come out ahead. Conversely if team t_i does lose a_i games and the Angels win all their games, the Angels will come out ahead of t_i. But obviously if team t_i loses g_{ij}, team t_j wins, and t_j may instead need to lose some different game.

Show how to formulate this as a maximum flow problem. The object is to create a graph and specify its edge capacities, in such a way that the maximum flow will be at least some value F if and only if it is possible to specify who wins and loses each game g_{ij} in a way that makes each team t_i lose at least a_i games. (Hint: Make a graph that has vertices not only for each team but also for each scheduled game not involving the Angels. Set up your capacities so that each game receives one unit of flow, representing one loss, which it can pass along to either team playing the game.)

You should describe this construction in general terms, including a list of all vertices and edges in the graph, the edge capacities, and the value of F. You may draw an example if you think that would make your description easier to understand, but you should not actually compute any maximum flows, nor do you need to prove that your construction works; I just want you to say how you would set up the problem.