6505 HW9: Matching and Flows

1. Consider the following *greedy* algorithm for finding a maximum matching: Start with an arbitrary edge as the initial matching. Find another edge that does not have a vertex in common with the current matching. If one exists, add it to the current matching. Repeat till no more edges can be added.

   (a) What is the running time of this algorithm on a graph with n vertices and m edges? [5pt]

   (b) Give an example of a graph where the algorithm does not find the maximum matching. [5pt]

   (c) Show that the matching found by the algorithm always has at least half as many edges as a maximum matching. [15pt]

   (d) Now consider a similar algorithm for finding a *maximum weight* matching in an edge-weighted graph: Greedily add the heaviest edge possible to the current matching; stop when no further edge can be added. Show that this algorithm finds a matching whose weight is at least half the optimum. [bonus]

2. Let G be a directed graph with two special vertices s and t. Any two directed paths from s to t are called *vertex-disjoint* if they do not share any vertices other than s and t. Prove that the maximum number of directed vertex-disjoint paths from s to t is equal to the minimum number of vertices whose removal ensures that there are no directed paths from s to t. [25]

3. Let G be a directed graph with capacities on its edges and two special vertices s and t. The capacity of a directed path from s to t is the smallest of the capacities of edges on the path. Give an efficient algorithm to find a path from s to t of maximum possible capacity. [25pt]

4. Let P range over the set of $s-t$ paths for two vertices s, t of a given undirected graph. Let C range over cuts that separate s and t. Then show that

$$\max_{P} \min_{e \in P} c_e = \min_{C} \max_{e \in C} c_e.$$

   Here c_e is the capacity of edge e. [25pt]