Due: Thursday 9/6 before 10:30

1. In class we discussed the relation between maximum s-t flow and minimum s-t cut problems.

   (a) Give an algorithm to solve the minimum s-t cut problem in time $O(n^3)$. You may use any of the max-flow algorithms we studied in class.

   (b) Give an algorithm for the maximum s-t flow problem that uses calls to a minimum s-t cut algorithm, but doesn't rely on any of the maximum flow algorithms we studied. How many mincut calls do you need?

2. You are given a capacitated graph $G = (V, E)$ and $s, t \in V$. You want to increase the maximum s-t flow value as much as possible, but are allowed to increase only one edge's capacity.

   (a) How do you find such an edge? (You may assume you have a max-flow or min-cut algorithm and can call it.) What is the running time of your algorithm?

   (b) Is it always possible to find such an edge? Justify your answer.

3. Let $G = (V, E)$ be a bipartite graph, that is, let $V = A \cup B$, where $A \cap B = \emptyset$ and every edge of G has one endpoint in A and another in B. Additionally, let every edge e have a nonnegative weight w_e. A *matching* is a subset $M \subseteq E$ of edges such that for every two edges $e, f \in M$, e and f have no common endpoints.

   Suppose you have access to an efficient maximum s-t flow algorithm. Show how to use it to find a maximum-weight matching in G.

4. Suppose G is an weighted acyclic directed graph, and let s and t be two vertices of G. Give an efficient algorithm for the *maximum-weight* s-t cut. Does your algorithm work if the graph contains directed cycles?