Analysis of Algorithms	**Problem Set 1**
CS 6820 Fall 2014	**Due 5pm, Friday, September 12, 2014**

This problem set has 5 problems with parts of varying difficulty. Harder parts are marked with a *, and these are extra credit. See the course web page for rules on extra credit work. A full solution for each problem includes proving that your answer is correct. If your group cannot solve a problem, but can do some parts, or have partial results, write down how far you got, and why are you stuck.

Students may work on homework in groups of up to 2-3 people. Each group may turn in a single solution set that applies to all members of the group. However, students are asked to understand each of their group's solutions well enough to give an impromptu whiteboard presentation of the solution. You may use any fact we proved in class without proving the proof or reference, and may read the relevant chapters of the Kleinberg-Tardos or Kozen books, provided you state them clearly. However, you may **not use other published papers, or the Web to find your answer**.

Solutions can be submitted on CMS in pdf format (only). Please type your solution or write extremely neatly to make it easy to read. If your solution is complex, say more than about half a page, please include a 3-line summary to help us understand the argument.

Please ask any clarifying questions using Piazza, where we will post all answers.

(1) Assume $G = (V, E)$ is a connected undirected graph. We defined a *graph cut* as the set of edges $C \subset E$ connecting the two parts of a partition (A, B) of the vertex set V with both A and B non-empty. We considered the *matroid associated with the graph*, whose independent sets are the subsets of edges that do not contain cycles, and whose basis are the spanning trees in G. Recall that a *matroid cycle* in a matroid $\mathcal{M}$ is a minimal set of edges C that are dependent, in the sense that for each proper subset $A \subset C$, $A \neq C$ A is independent. Finally, we define a *matroid cut* in a matroid $\mathcal{M}$ as a set of edges D such that there is no basis B in the matroid that avoids D (that is, it has $B \cap D = \emptyset$), and D is minimal in the sense that for each proper subset $A \subset D$, $A \neq D$, there is a basis B that avoids A. For each of the following statements, either prove them or provide a counterexample.

(a) In a connected graph $G = (V, E)$, any simple graph cycle C (a cycle going though every node at most once) is also a matroid cycle in the matroid associated with graph G.

(b) In a connected graph $G = (V, E)$ if C is a matroid cycle of the matroid associated with graph G, then the edges of C also form a simple cycle in the graph.

(c) In a connected graph $G = (V, E)$, any graph cut C defined by a partition (A, B) (with both A and B nonempty) C is also a matroid cut in the matroid associated with graph G.

(d) In a connected graph $G = (V, E)$ if D is a matroid cut of the matroid associated with graph G, then there is a partition (A, B) of the vertex set such that D is the graph cut associated with this partition.

(e) In a connected graph $G = (V, E)$ if C is a simple cycle and D is the edge set of a cut then the cardinality of the intersection of $C \cap D$ must be even.

(f*) Let C be a matroid cycle, and D a matroid cut in a matroid not necessarily associated with a graph. Then the cardinality of the intersection of $C \cap D$ must be even.

(g*) Let C be a matroid cycle, and D a matroid cut in a matroid not necessarily associated with a graph. Then the cardinality of the intersection of $C \cap D$ cannot be 1.

(2) You are consulting for a company which needs to build a spanning tree of cables to connect up their various offices. They have n offices, and so need $n-1$ edges in the spanning tree. Two contractors X and Y made offers to build edges of the tree at various costs. Consider a graph with edges labeled both with a cost, and a label X or Y depending which company made the offer to build the edge. Ideally they would want to build the minimum cost spanning tree, but it turns out that the company has contracts X and Y to build x and y of the edges respectively. They are not even sure if there is a spanning tree with x edges labeled x and y edges labeled Y.

(a) Give a polynomial time algorithm that either finds a spanning tree with exactly x edges labeled X and exactly y edges labeled Y or correctly concludes that no such tree exists.

(b*) Suppose each edge e has a cost $c_e \geq 0$. Give a polynomial time algorithm that finds the minimum cost spanning tree with exactly x edges labeled X and exactly y edges labeled Y, if such a tree exists.

Hint: you may want to consider modifying the cost of edges by one of the two companies by adding a constant c to each. For example, say edges offered by company X keep their cost c_e, while edges e offered by company Y will have cost $c+c_e$ for some constant for some constant c. How does this effect the solution? and how does this effect the min-cost spanning tree.

(3) For each part of this problem give a proof, or a counter-example with explanation.

You may use the following stronger variants of our main spanning tree lemma from class without proof.

Lemma 1 *Consider an edge e that is crossing a cut (A, B). If edge e has the unique minimum cost among all edges crossing the cut, then all minimum cost spanning trees contain e. If e is one of the minimum cost edges crossing the cut, then there exists a minimum cost spanning tree that contains e.*

(a) Consider the minimum spanning tree problem in an undirected graph $G = (V, E)$, with a cost $c_e \geq 0$ on each edge. Assume all edge-costs are different. Suppose you are given a spanning tree T with the guarantee that for every $e \in T$, e belongs to *some* minimum-cost spanning tree in G. Can we conclude that T itself must be a minimum-cost spanning tree in G?

(b) Consider the same question for the minimum-cost arborescence problem in a directed graph $G = (V, E)$ with root r. Suppose you are given an arborescence $A \subseteq E$ rooted at r with the guarantee that for every $e \in A$, e belongs to *some* minimum-cost arborescence in G rooted at r. Can we conclude that A itself must be a minimum-cost arborescence in G?

(c) Consider the special case of the question in part (b) when G is an acyclic graph, that is, it contains no directed cycles. Can we conclude in this special case that A itself must be a minimum-cost arborescence in G?

(d) How would your answer to (a), (b), and (c) change if we do not assume that all edge-costs are different?

(4) Consider a directed graph $G = (V, E)$ with a root r and nonnegative costs c_e on the edges. Suppose you are implementing a variant of the minimum arborescence algorithm that will be discussed in class on Friday September 5th, but the graph is huge, and you want to decrease the number of iterations.

(a) The algorithm discussed in class identifies 0-cost edges, and then contracts cycles in the graph of 0-cost edges. Argue briefly that we can instead contract whole strongly connected components.

(b) For a node v with no entering edge of 0-cost we let y_v be the minimum cost of an entering edge, and we modified the costs of all edges entering v to $c'_e = c_e - y_v$. Suppose we instated use $c'_e = \max(0, c_e - 2y_v)$, hoping that this will turn many more edges 0-cost. Let T' the minimum cost arborescence with costs c', and let T be the minimum cost arborescence with costs c. Show that $\sum_{e \in T'} c_e \leq 2 \sum_{e \in T} c_e$.

(c) Combining ideas of (a) and (b) consider the following algorithm for finding an arborescence. Define modified costs as in (b), contract all connected components of the subgraph of 0-cost edges for costs c'. Recursively use the algorithm to find an arborescence in the contracted graph and expand it to an arborescence of G using the edges with o c' cost contracted. Show that the resulting algorithm is a 2-approximation algorithm, that is, it finds an an arborescence of cost at most twice the minimum possible cost.

(5*) Let's go back to the original motivation for the minimum spanning tree problem: we are given a connected, undirected graph $G = (V, E)$ with positive edge lengths $\{\ell_e\}$, and we

want to find a spanning subgraph of it. Now, suppose we are willing to settle for a subgraph $H = (V, F)$ that is "denser" than a tree, and we are interested in guaranteeing that for each pair of vertices $u, v \in V$, the length of the shortest u-v path in H is not much longer than the length of the shortest u-v path in G. By the *length* of a path P here, we mean the sum of ℓ_e over all edges e in P.

Here's a variant of Kruskal's algorithm designed to produce such a subgraph.

- First, we sort all the edges in order of increasing length. (You may assume all edge lengths are distinct.)
- We then construct a subgraph $H = (V, F)$ by considering each edge in order.
- When we come to edge $e = (u, v)$, we add e to the subgraph H if there is currently no u-v path in H. (This is what Kruskal's algorithm would do as well.) On the other hand, if there is a u-v path in H, we let d_{uv} denote the total length of the shortest such path; again, length is with respect to the values $\{\ell_e\}$. We add e to H if $3\ell_e < d_{uv}$.

In other words, we add an edge even when u and v are already in the same connected component, provided that the addition of the edge reduces their shortest-path distance by a sufficient amount.

Let $H = (V, F)$ be the subgraph of G returned by the algorithm.

(a) Prove that for every pair of nodes $u, v \in V$, the length of the shortest u-v path in H is at most 3 times the length of the shortest u-v path in G.

(b) Despite its ability to approximately preserve shortest-path distances, the subgraph H produced by the algorithm cannot be too dense. Let $f(n)$ denote the maximum number of edges that can possibly be produced as the output of this algorithm, over all n-node input graphs with edge lengths. Prove that

$$\lim_{n \to \infty} \frac{f(n)}{n^2} = 0.$$

Hint: can you show that the minimum degree in this graph cannot be bigger than $\sqrt{n}$?