600.363 Introduction to Algorithms / 600.463 Algorithms I **Fall 2014**
Homework #7 **Due:** October 28, 2014, 1:30pm

Please start each problem on a new page, and include your name on each problem. You can submit on blackboard, under student assessment.

Remember: you may work in groups of up to three people, but must write up your solution entirely on your own. Collaboration is limited to discussing the problems – you may not look at, compare, reuse, etc. any text from anyone else in the class. Please include your list of collaborators on the first page of your submission. You may use the internet to look up formulas, definitions, etc., but may not simply look up the answers online.

Please include proofs with all of your answers, unless stated otherwise.

---

1 Effect of Squaring (33 points)

For each of the following two statements, decide whether it is true or false. If it is true, give a proof. If it is false, give a counterexample and an explanation.

(a) Suppose we are given an instance of the Minimum Spanning Tree problem on a graph G where all edge weights are positive and unique (so no two edges have exactly the same weight). Let T be a minimum spanning tree for this instance. Now suppose that we replace each edge weight w_e by its square w_e^2, creating a new instance with the same graph but different edge weights.

True or False: T must still be a minimum spanning tree for the new instance.

(b) Now suppose that we are again given a graph G with positive and distinct edge weights. Let s and t be two arbitrary vertices, and let P be a shortest $s-t$ path in G with respect to the given weights. As before, we create a new instance by squaring the weight of every edge.

True or False: P must be a shortest $s-t$ path in the new instance.

2 Boruvka's Algorithm (33 points)

Boruvka's MST algorithm (from 1926) is a bit like a distributed version of Kruskal. We begin by having each vertex mark the shortest edge incident to it. (For instance, if the graph were a 4-cycle with edges of lengths 1, 3, 2, and 4 around the cycle, then two vertices would mark the "1" edge and the other two vertices will mark the "2" edge.) For the sake of simplicity, assume that all edge lengths are distinct so we don't have to worry about how to resolve ties. This creates a forest F of marked edges. (Convince yourself why there won't be any cycles!) In the next step, each tree in F marks the shortest edge incident to it (the shortest edge having one endpoint in the tree and one endpoint not in the tree), creating a new forest F'. This process repeats until we have only one tree.

(a) Show correctness of this algorithm by arguing that the set of edges in the current forest is always contained in the MST.

(b) Show how you can run each iteration of the algorithm in $O(m)$ time with just a few runs of Depth-First-Search and no fancy data structures (heaps, union-find – remember, this algorithm was from 1926!) In other words, given a current set F of marked edges, show how to find the set of edges which consists of the shortest edge incident to each tree in F in time $O(m)$.

(c) Prove an upper bound of $O(m \log n)$ on the running time of this algorithm.

3 Submodular Maximization (33 points)

We saw in class that greedy algorithms are optimal when the problem is a matroid. But what about more complicated functions? Given a universe of elements U, let $f : 2^U \to \mathbb{R}_{\geq 0}$ be a nonnegative valuation function which maps sets to values (here 2^U is the power set of U, i.e. the set of all subsets of U). So, for example, if we have a weight function $w : U \to \mathbb{R}_{\geq 0}$ then the natural valuation function would be $f(S) = \sum_{e \in S} w(e)$. We showed in class that for such a function, finding the maximum value $S \subseteq U$ subject to S being independent in a matroid can be done quickly using the greedy algorithm.

We say that f is *monotone* if $f(S) \leq f(T)$ for all $S, T \subseteq U$ with $S \subseteq T$. We say that a f is *submodular* if

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$$

for all $S, T \subseteq U$.

Submodular functions play an extremely important role in discrete optimization, as they are essentially a discrete analog of concave or convex functions. In this problem we'll work out some basic properties and algorithms.

(a) Let $w : U \to \mathbb{R}_{\geq 0}$ be a weight function on the elements and let the value of a set be the sum of its elements, i.e. let $f(S) = \sum_{e \in S} w(e)$. Prove that f is monotone and submodular.

(b) An equivalent definition of a submodular function is that it exhibits *diminishing returns*. Let $S \subseteq U$ and let $e \in U$, and define $f_S(e) = f(S \cup \{e\}) - f(S)$. So $f_S(e)$ is the marginal benefit of adding e to S. Prove that if f is submodular then $f_S(e) \geq f_T(e)$ for all $e \in U$ and $S \subseteq T \subseteq U$.

The converse is also true (if f has diminishing returns then it is submodular), but the proof is a little bit trickier.

Let's consider the problem of maximizing a submodular function subject to a cardinality constraint, i.e. the problem of finding

$$\max_{S \subseteq U : |S| \leq k} f(S).$$

where $k \leq |U|$ is some input parameter. This is a special case of a matroid constraint (since $\{S : |S| \leq k\}$ form the independent sets of a matroid). It turns out that much of these ideas can be extended to matroid constraints, but cardinality constraints are an easier setting.

The greedy algorithm for this problem works as follows. We start with $S = \emptyset$, and repeat the following k times: find the element $e \in U \setminus S$ which maximizes $f_S(e)$, and add e to S. We can break ties arbitrarily. For the rest of this problem, our goal will be to prove that this is a $(1 - 1/e)$-approximation. More formally, let S be the set returned by the greedy algorithm, and let C be the optimal set, i.e. the set of size at most k which maximizes $f(C)$. We will show that $f(S) \geq (1 - 1/e)f(C)$.

Let a_i be the ith element chosen by the greedy algorithm and let $S_i = \{a_1, \dots, a_k\}$ be the first i elements selected by the greedy algorithm (so in the end the set it returns is S_k).

Let i be an arbitrary iteration (so $1 \leq i \leq k$). We begin by analyzing the marginal value of a_i.

(c) Prove that $\sum_{a \in C \setminus S_{i-1}} f_{S_{i-1}}(a) \geq f(C) - f(S_{i-1})$ (using submodularity and monotonicity).

(d) Prove that $f_{S_{i-1}}(a_i) \geq \frac{1}{k}\left(f(C) - f(S_{i-1})\right)$.

We can now use induction. Let's try to prove that $f(C) - f(S_i) \leq (1 - \frac{1}{k})^i f(C)$. The base case, when $i = 0$, is trivial since $f(C) - f(S_0) = f(C) = (1 - \frac{1}{k})^0 f(C)$.

(e) Prove by induction that $f(C) - f(S_i) \leq (1 - \frac{1}{k})^i f(C)$. Hint: you might want to start with the equality $f(C) - f(S_i) = f(C) - f(S_{i-1}) - f_{S_{i-1}}(a_i)$.

(f) Now prove that $f(C) - f(S_k) \leq \frac{1}{e} f(C)$. Hint: use the fact that $(1 - \frac{1}{x})^x \leq \frac{1}{e}$ for all $x \geq 1$.

This implies the theorem, since we can just rearrange the equation to get the desired inequality $f(S_k) \geq (1 - \frac{1}{e})f(C)$.