CS369B: Problem Set #1

Due in class on Thursday, January 24, 2008

Instructions:

(0) **Warning:** Budget *a lot* of time for this problem set.

(1) Students taking the course for a letter grade should attempt 4 of the following 5 problems; those taking the course pass-fail should attempt 2 of them.

(2) Some of these problems are quite difficult. I highly encourage you to start on them early and discuss them extensively with your fellow students. If you don't solve a problem to completion, write up what you've got: partial proofs, lemmas, high-level ideas, counterexamples, and so on. This is not an IQ test; we're just looking for evidence that you've thought long and hard about the material.

(3) You may refer to your course notes, and to the textbooks and research papers listed on the course Web page *only*. You cannot refer to textbooks, handouts, or research papers that are not listed on the course home page.

(4) Collaboration on this homework is *strongly encouraged*. However, your write-up must be your own, and you must list the names of your collaborators on the front page.

(5) No late assignments will be accepted.

Problem 1

Fibonacci heaps.

(a) Unlike normal heaps or binomial heaps, Fibonacci heaps can include trees that are very deep. Prove this by giving a sequence of operations for a Fibonacci heap with n elements such that the depth of one of its trees is $\Omega(n)$.

[Hint: induction on n.]

(b) Suppose we modify Fibonacci heaps so that we only rip out a node if k of its children have already been ripped out. Intuitively, this should speed up (the constant factors of) the amortized time bound for Decrease Key, since there will be fewer cascading cuts, and slow down Extract Min, since the trees can get more scraggly. Try to quantify how the amortized bound for Extract Min degrades as a function of k.

Problem 2

Matroids. A *matroid* is given by a ground set E of elements and a collection $\mathcal{I}$ of subsets of E, called *independent sets*, satisfying three properties:

(i) $\emptyset \in \mathcal{I}$;

(ii) $\mathcal{I}$ is closed under subsets: if $S \subseteq T$ and $T \in \mathcal{I}$, $S \in \mathcal{I}$;

(iii) *the Exchange Property*: if $S, T \in \mathcal{I}$ and $|S| > |T|$, then T can be extended by some element of $S \setminus T$: there is an element $e \in S \setminus T$ such that $T \cup \{e\} \in \mathcal{I}$.

A *basis* of a matroid is a maximal independent set. A *cycle* is a minimal dependent set (i.e., all of its proper subsets are in $\mathcal{I}$). A *cut* is a minimal subset that intersects every basis.

(a) [Do not hand in.] Convince yourself that all bases of a matroid have the same cardinality. This is called the *rank* of the matroid.

(b) [Do not hand in.] Let $\mathcal{I}$ denote the acyclic subgraphs of an undirected connected graph. Convince yourself that this is a matroid, the bases are the spanning trees of the graph, and cuts and cycles of the matroid correspond to cuts and cycles of the graph.

(c) Let M be an arbitrary matroid and assign distinct real-valued costs to its elements. Prove the *Cut Property:* for every cut S of M, the cheapest element of S belongs to every minimum-cost basis of M.

(d) Let M be an arbitrary matroid and assign distinct real-valued costs to its elements. Prove the *Cycle Property:* for every cycle S of M, the costliest element of S belongs to no minimum-cost basis of M.

(e) [Do not hand in.] Let M be an arbitrary matroid and assign distinct real-valued costs to its elements. Define the *greedy algorithm* for computing a minimum-cost basis of M as in Kruskal's MST algorithm: start with $S = \emptyset$; go through the elements of M in order from cheapest to costliest, in each iteration adding the current element to the set S if and only if doing so preserves independence. Convince yourself that the correctness proof for Kruskal's algorithm remains valid in this general matroid context.

(f) Consider a ground set E and a collection $\mathcal{J}$ of subsets of E that satisfies the first two defining properties of a matroid but not the third. Prove that there exists an assignment of costs to the elements E such that the greedy algorithm fails to output the minimum-cost subset in $\mathcal{J}$.

Problem 3

An $O(m \log\log n)$-Time Implementation of Borůvka's Algorithm.

(a) Suppose as a preprocessing step, we sort the edges in each node's adjacency list by cost. (This takes $O(m \log n)$ time.) Show how to implement the rest of Borůvka's algorithm to run in time $O(m+n \log n)$.

[Hint: It might help to implement contractions only implicitly. Have each node of the original graph keep track of which edges in its adjacency list are useless, in that they point to a different node in the same connected component of the tree-so-far. Keep the potentially useful ones sorted by cost. Can you achieve a bound of $O(n)$ per phase, plus $O(m)$ additional time overall to maintain the adjacency lists?]

(b) Call an array *partially sorted with parameter k* if every element is less than k positions away from its rightful position in the sorted version of the array. ($k = 1$ is fully sorted, $k = n$ is unsorted.) Show how to k-partially sort an array of n numbers in $O(n \log \frac{n}{k})$ time.

[Hint: linear-time median.]

(c) Strengthening the result in (a), show that $O(m \log\log n)$ preprocessing time for partial sorting is enough to obtain a bound of $O(m + n \log n)$ for the rest of the work done by Borůvka's algorithm.

(d) Explain why performing $\log\log n$ Borůvka phases prior to your algorithm in (c) yields an $O(m \log\log n)$ MST algorithm (including all preprocessing steps).

Problem 4

MSTs and Shortest-Path Trees: The Best of Both Worlds. Our first two topics are minimum-spanning trees and shortest-path trees. But what if we want a single tree possessing the good properties of both? Consider an undirected graph $G = (V, E)$ with distinct and nonnegative edge lengths and a source vertex s.

(a) Show that the shortest-path tree rooted at s can be an extremely bad approximation of the MST (in terms of the sum of the lengths of the edges in the tree). How big a gap can you obtain? (You should give a family of examples such that, as the number of nodes n goes to infinity, the total edge cost of the shortest-path tree is $f(n)$ times the MST cost, where $f(n)$ is as large a function as possible.)

(b) Conversely, show that the MST T can be a bad approximation of the shortest-path tree, in that there can be vertices v such that the s-v path in T has length much larger than that of a shortest such path in G. How big a gap can you obtain?

(c) Prove that there always exists a tree that simultaneously $O(1)$-approximates the cost of an MST and also $O(1)$-approximates shortest-path distances from s to all other vertices. What kind of upper bounds on the trade-offs between the two approximation factors can you obtain?

[Hint: start with the MST, do a traversal starting from the source. If a path in the current tree is too long relative to a shortest path in G, shortcut it. You need to bound the cost of all the edges that you add.]

(d) Can you obtain any interesting lower bounds on the best-possible trade-offs between the two approximation factors?

Problem 5

Subgraphs that preserve all distances. Consider a graph $G = (V, E)$. This problem is a generalization of the previous in that we want a subgraph of G that approximately preserves *all* shortest-path distances in G, not just those involving a distinguished source vertex. For a subgraph H of G, let $d_H(u, v)$ denote the length (in hops) of a shortest u-v path in H. Note that $d_H(u, v) \geq d_G(u, v)$ for all u, v, H. Define α_H by $\max_{u,v}(d_H(u, v)/d_G(u, v))$ – i.e., the largest factor by which a shortest path in H is bigger than one (with the same endpoints) in G. If H is not connected, then we define $\alpha_H = +\infty$.

(a) Show that if we restrict H to be a spanning tree of G, then for some graphs G we are stuck with $\alpha_H = \Omega(n)$.

(b) We therefore seek to minimize both the number of edges of H and α_H. Suppose we fix a target α^* for the latter and try to minimize the former. Consider the following Kruskal-like heuristic: go through the edges of G in arbitrary order, and include an edge (u, v) into the graph H if and only if there is not yet a u-v path in H with α^* or fewer hops. Prove that this algorithm terminates with a subgraph H satisfying $\alpha_H \leq \alpha^*$.

(c) Prove an upper bound, in terms of α^* and n, on the number of edges that the subgraph H will include.

[Hint: use the fact that a graph with *girth* (i.e., length of the shortest cycle) equal to g has no more than $n^{g/(g-2)}$ edges. You can assume this without proof.]

(d) Suppose the edges of G have nonnegative (not necessarily unit) weights. Give an analog of the algorithm in (b) so that the bounds in part (b) (in terms of α_H) and (c) (in terms of the number of edges) continue to hold.

Can you find any meaningful upper bound the total edge *cost*, rather than merely the number of edges, of the subgraph H constructed by this algorithm? You could, perhaps, parametrize such an upper bound in terms of the MST cost, which is a lower bound on the cost of every subgraph with finite α-value.