COS 423 Problem Set 3(revised) Due Wednesday March 23, 11AM
Spring 2011
Tarjan
No collaboration on 1 and 2; collaboration allowed on 3 and 4.
Definition of "safety" corrected in Problem 3. (The definition is correct in the Lecture 9 slides.)

1. (type-3 rank-pairing heaps) A type-3 rank-pairing heap is just like a type-2 rank-pairing heap except that the allowed node types are different. Specially, in a type-3 rp-heap, a node can be 1,1; 1,2; 1,3; or 0,j for any $j > 2$. The purpose of this problem is to implement and analyze type-3 rp-heaps. You may want to consult the latest version of Lecture 7 (rank-pairing heaps) for ideas.

(a) Give an implementation of the decrease-key operation on a type-3 rp-heap. Specifically, modify the implementation of decrease-key on type-2 rp-heaps so that it preserves the type-3 rank invariant.

(b) Prove that the maximum node rank in a type-3 rp-heap containing n nodes is at most $c\lg n$ for some constant c. Make c as small as you can.

(c) Prove that the following amortized time bounds hold for the type-3 pairing heap operations starting with no heaps: each operation except delete-min and delete takes O(1) amortized time; each delete-min and delete operation takes O($\lg n$) amortized time, where n is the number of items currently in the heap.

2. (negative cycles and breadth-first scanning) The purpose of this problem is to further explore the effect of negative cycles on the behavior of the breadth-first scanning algorithm for the single-source shortest path problem. Consider the breadth-first scanning algorithm (without subtree disassembly): see Lecture 8 (revised), slide 28. Give an example of a graph with a negative cycle and a run of breadth-first scanning on this graph such that at some point in the computation the parent pointers contain a cycle, but at some later point they define a tree rooted at s, the start vertex.

3. (heuristic search) The purpose of this problem is to verify some of the properties of heuristic search as described in Lecture 9, slides 22-24. We are given a directed graph with an arc length $c(v, w)$ for each arc (v, w). Our goal is to find a shortest path from s to t. We are also given an easy-to-compute vertex function e such that $e(v)$ is an estimate of the distance from v to t for each vertex v. We call the function $e(v)$ *safe* if $e(t) = 0$ and $e(v) \le c(v, w) + e(w)$ for every arc (v, w). The one-way (forward) heuristic search algorithm finds a shortest path from s to t by running the scanning algorithm and choosing as the next vertex v to be scanned the v in L such that $d(v) + e(v)$ is minimum. (Ties are broken by using a fixed vertex order.)

(a) Prove that if e is safe, $e(v)$ is at most the length of a shortest path from v to t, for every vertex v.

(b) Prove that if e is safe, then for any vertex v, when v is chosen to be scanned, $d(v)$ is the length of a shortest path from s to v. Conclude that the algorithm will scan each vertex at most once, and it can stop when t is chosen to be scanned.

(c) Suppose e and f are two safe distance estimates such that $e(v) \leq f(v)$ for every vertex v. We want to compare the efficiency of the algorithm using e as an estimate to its efficiency using f as an estimate. Let $S(e)$ and $S(f)$, respectively, be the set of vertices scanned when using e or f, respectively, until t is chosen to be scanned. Prove that $S(f)$ is contained in $S(e)$. That is, every vertex scanned when the algorithm uses f is also scanned when the algorithm uses e. Thus a bigger distance estimate results in no additional vertex scans, as long as the estimate is safe.

(d) Consider a graph with arc lengths and a distance estimate e such that $e(t) = 0$ and $e(v)$ is at most the length of a shortest path from v to t for every vertex v. Do part (i) or part (ii) below. For extra credit, do both parts.
(i) Prove that when t is first scanned, $d(t)$ is the correct shortest distance to t.
(ii) Give an example of such a graph and such a distance estimate on which some vertex is scanned at least twice before t is scanned.

4. Devise a class of graphs with no negative cycles on which the scanning algorithm with a bad choice of scanning order runs in time exponential in the number of vertices. Prove that your construction works.