TEL AVIV UNIVERSITY
Department of Computer Science
0368.4281 – Advanced topics in data structures
Fall Semester, 2004/2005

Homework 3, December 13, 2004

Due on Monday December 27. Don't forget to keep a copy of the homework.

1. (a) Show how to extend dynamic trees to support an operation called $evert(v)$. This operation changes the root of the (actual) tree containing v to be v. In other words we change the direction of the edges on the path from v to the previous root of the tree.
(b) Show how to extend dynamic trees to support the operation $lca(u,v)$. This operation assumes that u and v are in the same actual tree and returns the lowest common ancestor of u and v in their tree.
Try to get both operations to run in $O(\log n)$ time.

2. An ordered rooted tree is a rooted tree (think of all the edges as directed towards the root) where there is a total order on the edges from the children $u_1, \ldots, u_k$ of a vertex v into v. (I.e. we think of the edges (u_1, v), (u_2, v), ..., (u_k, v) as an ordered list.) Furthermore, each edge (u, v) has a cost $c(u, v)$ associated with it. Describe a data structure that supports the following operations on an ordered rooted forest.
(a) Link(w,v,c(w,v)): Add an edge e from w to v, thereby making w a child of v in the forest (w is assumed to be a root). The new edge (w, v) is put last in the total order defined on the edges entering v and its cost is set to $c(w, v)$.
(b) Cut(v): Remove the edge from v to its parent from the forest.
(c) Split(v,e): Here $e = (u, v)$ is assumed to be an edge from a child u of v to v. We split v into two nodes v' and v''. Nodes v' and v'' have the same parent as v (and node v is deleted). The edges $(v', p(v))$ and $(v'', p(v))$ have the same cost as the cost of the edge $(v, p(v))$. The edge $(v'', p(v))$ follows $(v', p(v))$ in the total order of the edges incident to $p(v)$ and they related to the other edges incident to $p(v)$ in the same way as $(v, p(v))$ was related to them. The children of v' are those connected to v by edges preceding e (including e) in the total order of the edges between v and its children (They remain ordered in the same way). The children of v'' are those connected to v by edges that follow e (including e) in the total order of the edges between v and its children (They remain ordered in the same way).
(d) Cycle(v,e): Here $e = (u, v)$ is assumed to be an edge from a child u of v to v. Cyclically shift the edges between v and its children so that the edge e is last in the total order imposed on them.
(e) Change-cost(e,r): Change the cost of edge e to r.
(f) Find-min(v): Find the minimum cost edge on the path from v to the root of its tree. If there are ties return the closest edge to the root of minimum weight.

Describe your implementation precisely and clearly, distinguish between the abstract data type and your representation. State how you represent each edge and vertex in the forest. Analyze the performance of your data structure.

3. On a set of n nodes we perform a sequence of operations, each of which is one of the followings.
(a) $insert(u, v)$: Adds an edge between u and v.
(b) $delete - oldest$: Removes the edge that was inserted first among the edges currently in the graph.
(c) $connected(u, v)$: Answers true if there is a path from u to v in the current graph. Otherwise answers false.

Describe a data structure that supports these operations and analyze its performance. (A structure supporting each operation in $O(\log n)$ amortized time would receive maximum score.)

4. For any positive integer n, give a sequence of Fibonacci heaps operations that creates a Fibonacci heap consisting of just one tree that is a linear chain of n nodes (make your sequence as short as you can).

5. Brouvka's algorithm for finding a minimum spanning tree works in iterations each of which performs the followings.
1) Each vertex picks the edge of minimum cost incident to it and colors it blue.
2) Add all blue edges to the minimum spanning tree and contract them.
The algorithm terminates when only one vertex remains.

1. Describe a straightforward implementation of Brouvka's algorithm that runs in $O(m \log n)$ time. (As usual m is the number of edges and n is the number of vertices.)

2. Using the idea of Gabow, Galil, Spencer, and Tarjan to group edges into packets, describe an implementation of Brouvka's algorithm that runs in $O(m \log \log n)$ time.