Exercise sheet 8.

Data structures SoSe 2020

László Kozma, Katharina Klost

Due 12:00, June 19th, 2020

Exercise 1 Ancestor queries in trees *2+3 Points*

Consider a rooted tree T, with nodes of arbitrary degree. Recall the *preorder* and *postorder* traversal of a tree.

(a) Let x, y be two nodes of T. Show that x is the ancestor of y *if and only if* x precedes y in the preorder traversal but not in the postorder traversal.

(b) Using this observation, design a data structure that efficiently supports the following queries, starting from a tree consisting of a root r:
insert(x) creates a new node, linking it as the rightmost child of x, and returns a pointer to this node,
delete(x) removes node x if it is a leaf, otherwise reporting error, and
ancestor(x, y) returns *true* if x is the ancestor of y and *false* otherwise.

What is the running time of operations?

Hint: You can use the list labeling data structure from class as a black box.

More hint: Store a traversal of the tree from which you can get the ordering of nodes both by preorder and by postorder.

Exercise 2 List labeling *3 Points*

Suppose we implement the algorithm for list labeling from class, but we set the overflow densities to the same constant value at every node. (Recall that the density of x is the fraction of leaves in the subtree rooted at x that are nonempty. The overflow density is the threshold above which the subtree is considered "too dense".)

Sketch a small example that shows that this strategy can be very inefficient.

Exercise 3 Maintaining a partial order *3 Points*

Suppose we store a directed acyclic graph with n vertices (initially there are no edges).

Maintain an integer *label* ℓ for each vertex, so that $\ell(x) < \ell(y)$, whenever y is *reachable* from x (by following directed edges). If neither of x and y is reachable from the other, then the labels may be in arbitrary relation.

The operation *insert*(x, y) adds the directed edge $x \to y$ to the graph, updating the labels, and reporting an error if a cycle has been created.

Show how to implement m insert operations (assume $m > n$) in time $O(m^2)$ (easy).

Bonus (+3p): Improve the running time to $O(mn)$ or better.

Exercise 4 Programming exercise *1 Point*

The programming exercise is due June 29th (30 points).

For this time, please write a brief summary (one paragraph) of your plan and/or progress so far.

Total: 12 points. Have fun with the solutions!