Lecture No. 5, 10/1, Topics

COS 521
Fall 2002
Tarjan

Persistence:

Fat node method: add new pointers, data with version stamps.

Navigation: search among version stamps for most recent version.

Partial persistence: version list: can number versions sequentially from 0; versions are linearly ordered.

To select correct version, use a binary search tree, or the Van Emde Boas data structure for insertion and accessing the predecessor. Time for a navigation step is $O(\log m)$ or $O(\log\log m)$, respectively, where $m = \#$ versions. Space is O(1) per update. To get this bound for Van Emde Boas use hashing (randomization).

P. Van Emde Boas, "Preserving order in a forest in less than logarithmic time and linear space," *Info. Proc. Lett.* 6 (1977), 80-82

Full persistence: version tree: linearize via Euler Tour (list continuing preorder, postorder visits). Assign numbers to versions using Dietz-Sleator algorithm for numbering list elements with insertion. Can then use the same navigation techniques as with partial persistence, with the same bounds.

P. Dietz and D. Sleator, "Two algorithms for maintaining order in a list," *Proc. 19th Annual ACM Symp. on Theory of Computing* (1987), 365-372.

Node copying method to obtain partial persistence:

Keep a *fixed* number of extra pointer slots per node, create a new node copy when these become full.

Only the most recent copy of each node is active.

Adding new pointers proliferates back through predecessors of copied nodes.

To get constant amortized bound per ephemeral update: need constant bounded in-degree, must add predecessor pointers for current version, and need a number of extra pointer slots equal to maximum number of predecessors.

Node splitting method to obtain full persistence: like node copying for partial persistence, except that when a node becomes full it is split into two nodes, each with half its extra slots free.

To get constant amortized bound, need constant bounded in-degree, must symmetrize by adding predecessor pointers for all versions, and need a number of extra pointer slots equal to twice the number of pointers per ephemeral node in the symmetrized version.

J. Driscoll, N. Sarnak, D. Sleator, and R. Tarjan, "Making data structures persistent," *J. Computer and System Sciences* 38 (1989), 86-124.