Lecture No. 1, 9/17, Topics

COS 521
Fall 2002
Tarjan

Binary Trees
Binary Search Trees, internal node version and leaf version
Rotations in a binary search tree
Red-black Trees: O(logn) insert/delete time, with O(1) rotations
Optimum Binary Search Trees

Internal: $O(n^3)$ by dynamic programming
$O(n^2)$ by dynamic programming plus "monotonicity" (Knuth)

Open problem #1: faster?

Leaf: O(nlogn) by Hu-Tucker algorithm: alphabetic Huffman coding

Open problem #2: faster?

Self-adjusting binary search trees

Splay trees: splay after each access.

Open problem #3: dynamic optimality of splay trees? (within a constant factor of best possible on any access sequence?)

References

Binary trees, binary search trees, red-black trees, self-adjusting trees:

R. Sedgewick, *Algorithms in C, Parts 1-4,* Third Edition, Addison-Wesley, 1998. (algorithms)

R. Tarjan, *Data Structures and Network Algorithms,* Society for Industrial and Applied Mathematics, 1983. (algorithms and analysis)

Optimum binary search trees:

T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms,* Second Edition, MIT Press, 2001. (internal $O(n^3)$)

D. Knuth, "Optimum binary search trees," *Acta Informatica* 1 (1971), 14-25. (internal $O(n^2)$)

T. Hu and A. Tucker, “Optimal computer search trees and variable length alphabetic codes,” *SIAM J on Applied Mathematics* 21 (1971), 514-532.

Huffman Codes:

T. Cormen, et al. [ibid]

Self-adjusting search trees:

R. Tarjan [ibid]