Lecture No. 4, 9/26, Topics

Randomization in search structures

Average-case analysis of deterministic Quicksort

= analysis of randomized Quicksort

= average total insertion time for binary trees

= average total insertion time for treaps

leads to analysis of treaps

Treap: randomized search tree

O(log*n*) access/insert/delete time

O(1) rotations per insert/delete

R. Seidel and C. Aragon, "Randomized search trees," *Algorithmica* 16(1996), 464-497.

Skip Lists

W. Pugh, "Skip lists: a probabilistic alternative to balanced trees," Comm. ACM 33(1990), 668-676.

Persistent data structures

Partial, full, confluent persistence

Post office problem

Construct Voronoi diagram

Slice diagram vertically at each vertex

Construct a (vertical) search tree for each slice, plus one horizontal search tree

Adjacent search trees are related by O(1) inserts/deletes

$\Rightarrow$ partial persistence applies

Obtaining persistence

1. Copy the entire structure: linear time and space

2. Copy only the nodes that change.

   Copying proliferates backward along pointers

   ⇒ structure must be acyclic

   O(logn) time/space per insert/delete on balanced binary trees

   (with only child pointers)

   = purely functional programming: can't change old nodes, can only create new ones

N. Sarnak and R. Tarjan, "Planar point location using persistent search trees," Comm. ACM 29(1986), 669-679.