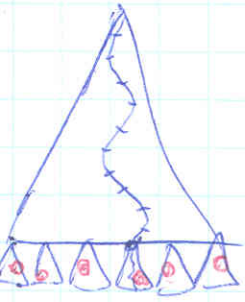
y-fast trees [Willard 1984]

6.897
L 9.5
Mar. 3, 2005

- store the "location" of all 1 bits in tree
  i.e. store all prefixes of binary rep. of $x \in S$ in a hash table
  $\Rightarrow O(w)$ update, $O(nw)$ space $\rightarrow$ fix later
- for $x \in \mathcal{U}$, path from root to x reads $1\,1\cdots 1\,0\,0\cdots 0$ (root … leaf x)
- can find transition from 1 to 0 by binary search $\Rightarrow O(\lg w)$
  i.e. lowest ancestor y of x containing node
- if $x \in \text{left}(y)$ then $\min(\text{right}(y)) = \text{successor}(x)$
- if $x \in \text{right}(y)$ then $\max(\text{left}(y)) = \text{predecessor}(x)$
  ↑ augment
- store elements of S in linked list to find other succ/pred.

Indirection: cluster elements into consec. groups of size $\Theta(w)$

- store each group with balanced search tree $\Rightarrow O(\lg w)$
- use hash table solution on representative from each group
  ↳ any element from $\mathcal{U}$ "in the right ~~place~~ range
  $\Rightarrow O(n)$ space



- Predecessor/Successor:
  - hash table finds "pred. & succ." representatives
  - search in the two corresponding clusters/groups
- Insert/Delete:
  - insert/delete in group with representative = pred(x) (nonstrict) in hash table
  - if group grows by $2\times$ or shrinks to $\frac{1}{4}\times$ [or to 0]
    then split or merge with neighbor & possibly resplit
    $\Rightarrow$ change $O(1)$ representatives
    $\Rightarrow O(w)$ cost to update hash table
  - happens only after $\Omega(w)$ updates
    $\Rightarrow O(1)$ amortized
- general transform: keep $O(\lg w)$ query, reduce { update $w^{O(1)} \rightarrow \lg w$; space $nw^{O(1)} \rightarrow n$