Finger search trees

Goal

Keep sorted lists subject to the following operations:

find(x,L)

insert(x,L)

delete(x,L)

catenate(L1,L2) : Assumes that all items in L2 are greater than all items in L1.

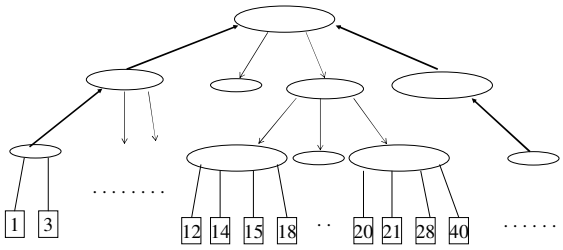
split(x,L) : returns two lists one with all item less than or equal to x and the other with all items greater than x.

Goal (cont)

In addition, we want to speed up operations near the ends of the list.

Take a regular search tree and reverse the direction of the pointers on the leftmost and the rightmost spines (paths).

Finger 2-4 trees



Finger 2-4 trees

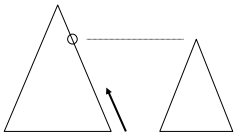
Start the search in parallel from the leftmost and rightmost nodes on the spines.

Search for an element at distance d from one of the endpoints of the list takes O(log d) time.

Insertions and deletions still take O(log n) worst case time but O(log d) amortized time (including the search time).

Finger trees catenation

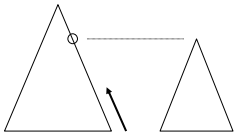
Like for regular trees but the search on the spine of the tall tree for a node of the same height as the small tree starts from the lowest node on the spine.



Finger trees catenation (cont)

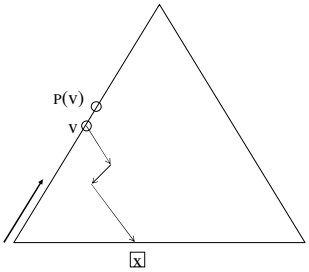
Catenation takes O(log n) on the worst-case

But amortized O(1 + min{h1, h2}) =
O(1 + min{log (n1), log (n2))



7

Splitting finger search trees

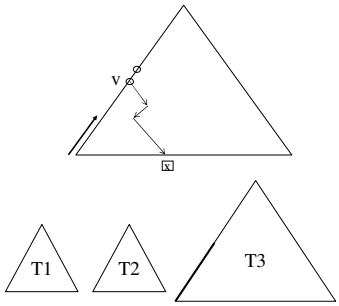


Catenate bottom-up trees to the left of the path from v to x and to the right of the path from v to x. Obtain T1 and T2.

Delete v as a child from p(v), rebalance as in deletion. Obtain T3.

8

Splitting finger search trees (cont)



Make T1 and T2 finger trees.

Fix the spine of T3.

Return T1 and the tree obtained from the catenation of T2 and T3.

9

Split -- analysis

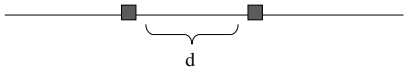
Split takes O(log n) worst-case.

But O(log d) amortized.

10

Homogenous finger search trees

Want to be able to get from every element to every other element in time proportional to the logarithm of the distance between them.

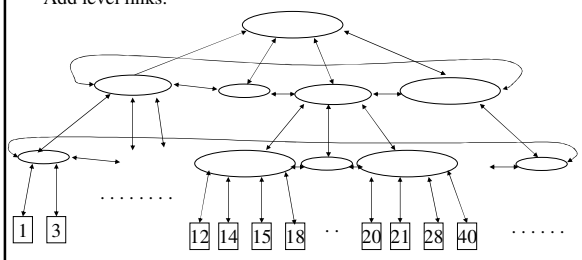


O(log min {d, n-d})

11

Homogenous finger search trees (cont)

Add level links.



12

Search

Start from x and search for y.

Say x < y.

Keep going up until one of the following conditions holds:

1) You hit the right path of the tree

2) Your right neighbor has a key which is not smaller than y

3) You are on the left path and your neighbor on the right path has a key smaller than y

Search down one or two subtrees

13

Search (analysis)

Suppose the search went up to level h.

Consider the node x reached on the way up on level h-1.

The leftmost subtree of the right neighbor of x contains only items larger than x and smaller than y.

==> $d \geq 2^{h-2}$

The rightmost subtree of the left neighbor of x contains items either smaller than x or larger than y.

==> $n-d \geq 2^{h-2}$

14