Module 4 - Dictionaries and Balanced Search Trees

Graham Cooper

June 2nd, 2015

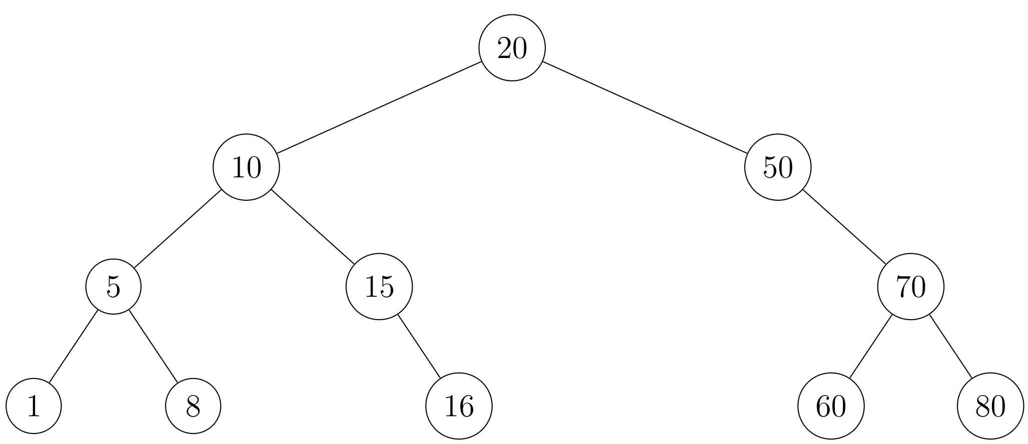
Dictionaries

- An ADT
- Data (key, value) pairs
- operations: search, insert, delete

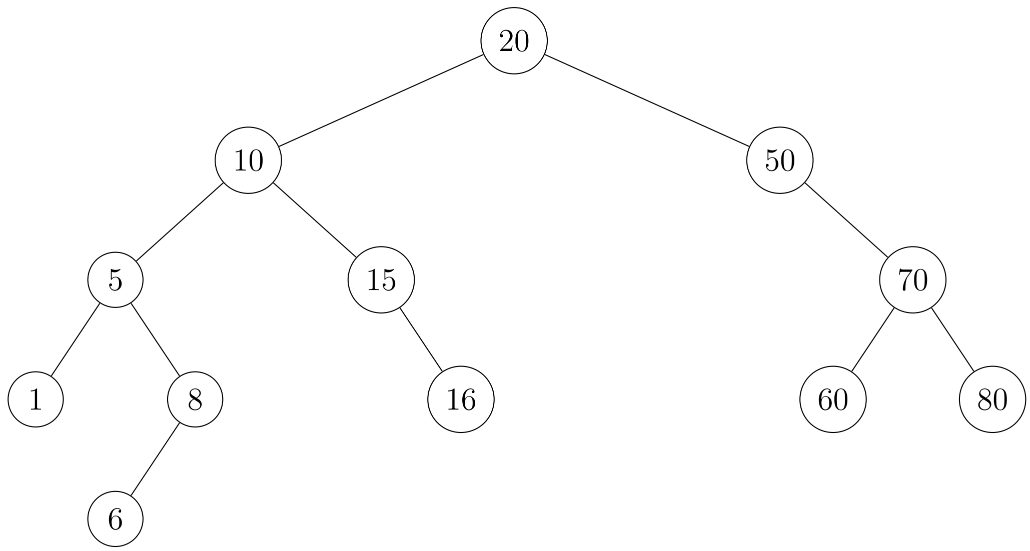
Data Structures for Dictionaries:

- unsorted array or linked list
  - search: O(n)
  - insert: O(1)
  - Delete: O(n)
- sorted array
  - search - binary search O(logn)
  - insert O(n)
  - delete O(n)

0.1 BST



Insert 6



Delete in a BST

- if n is a leaf - just delete it
- if n is a node with one child, replace it with its child
- if n has two children, replace with the predecessor (rightmost on the left) or sucessor (left most)

Fun with AVL trees (control)

insert(y)
- insert as a leaf like usual bst
—-Move up, update balance factors
—— if x.balance factor $\in$ {-2, 2}
——— fin(x)
Fin is called at most once after that bf, are all fixed (no need to update higher levels)

fin(x)
if x.bf = -2 (too heavy on the left)
{ – if x.left.bf = 1 then
—- x.left $\rightarrow$ rotate(left)
– n $\rightarrow$ rotate Right }

if x.bf = +2 (too heavy on right)
{
– if x.right.bf = -1
—- x.right $\rightarrow$ rotateRight
– x $\rightarrow$ rotateLeft()
}

Delete(j)
– as usual BST, replcae with successor/predecessor
– move from location of seccessor, predecessor
– – move up
– – – if x.bf $\leftarrow$ {-2,2}
– – – – fin(x)

fin may be called log(n) times because the height changes.

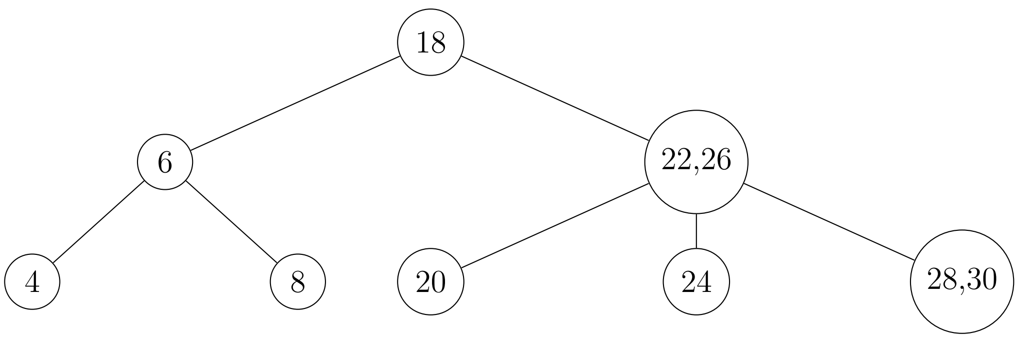
insertion
– Insert as a usual BST
– – O(height)
– move up check balance factor, apply fin() if neccessary
– time for fin $\rightarrow$ O(1)
In total, time for insert: Θ(height)

Height of AVL:
let N(h) denote the minumum number of nods in an AVL tree with height h.
N(h) =
0 if h = -1
1 if h = 0
N(h-1) + N(h-2) + 1 else

N(h) = fionacci(h+3) - 1

$= \text{roof}(\frac{p^{h+3}}{5}) - 1$
where p = $\frac{1+\sqrt{5}}{2}$

B-Tree (beautiful tree)



An (a,b) tree B-Tree

1. An ordered tree
2. Each internal node has at least a, and at most b children, root has at least 2, at most b children
3. A node with k children $\rightarrow$ k-1 key value pairs
   – An (roof($\frac{u}{2}, u$)) B-tree is order u B-tree, eg u = 2$\rightarrow$ order b-tree $\rightarrow$ A(2,3)-tree

Insertion
–Insert at a leaf – overfilled noes send the middle key to the parent and split
Deletion
– As BST, the removed key is replaced by successor/predeccesor (which is a leaf)
– if a node becomes underloaded
– – if $\exists$ a sibling with an extra key (more than 'a' keys)
– – – take the key from parent and parent gets a key from the sibling.