CS 180 – Search Trees

Note Title 11/9/2009

Announcements

- HW due Wed.
- review session Wed. in class
- midterm on Thursday in class
- schedule/lecture notes should be updated now, so tree & heap lectures are online.

Topics on this exam:

- Linked lists (& end of stacks/queues)
- Vectors
- Lists (& iterators)
- Sorting
- Trees
- Heaps

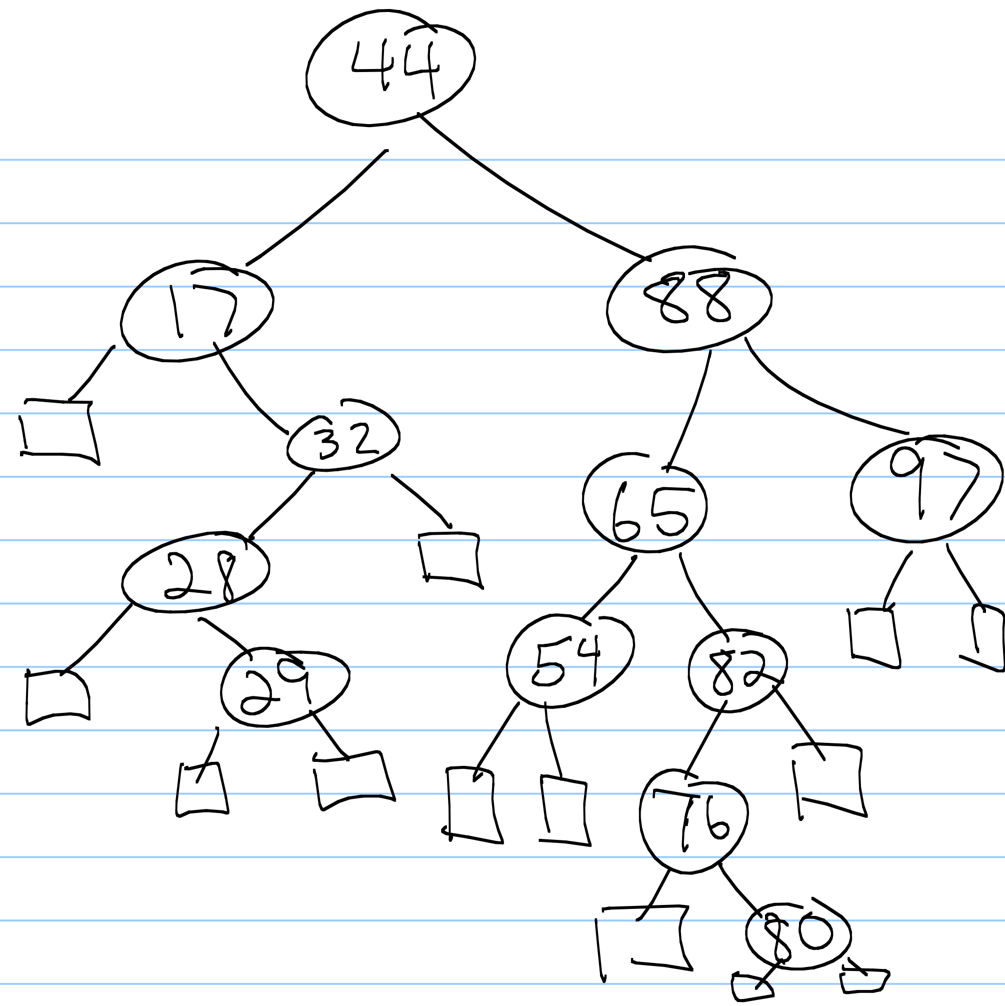
} no coding - see HW problems

Note: Asymptotic running times are fair game also!!

Binary Search Trees

A binary tree such that each internal node v of T stores a key k, and:

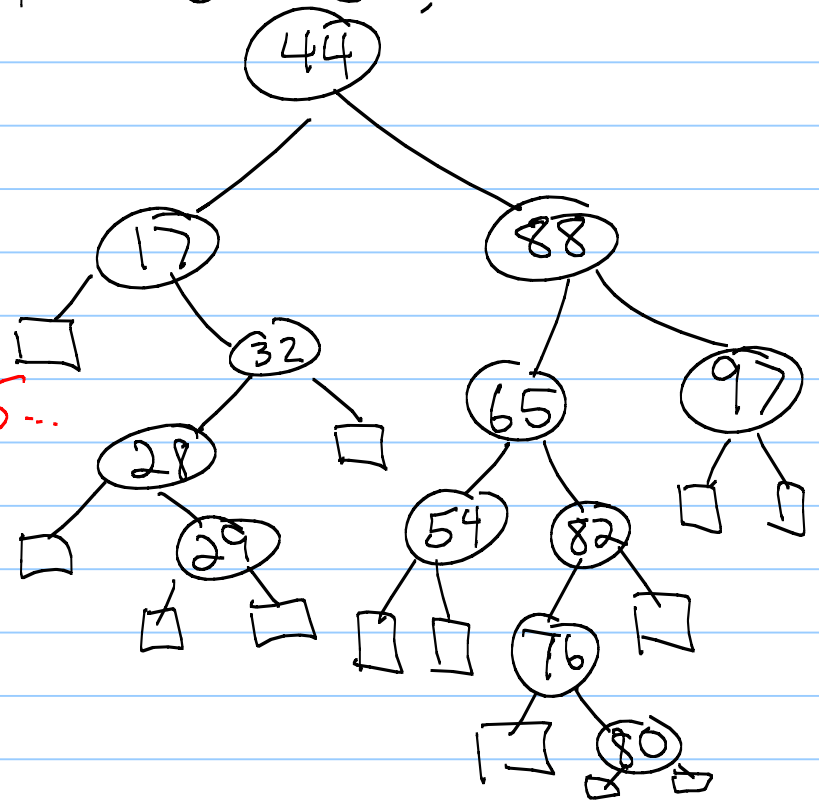
- keys stored at nodes in the left subtree of v are less than or equal to v
- keys store in the right subtree are greater than or equal to v.


44
17
88
32
65
97
28
54
82
29
76
80

Q: What type of traversal will print the elements in sorted order?

preorder

17 28 29 32 44 54 65...



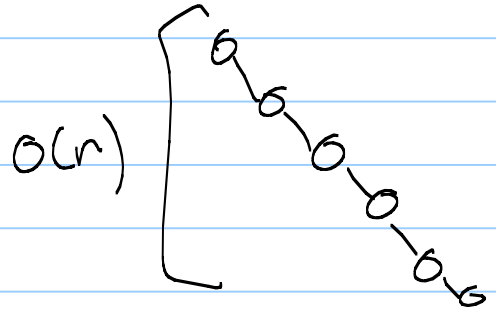
How do we search for an element in the tree?

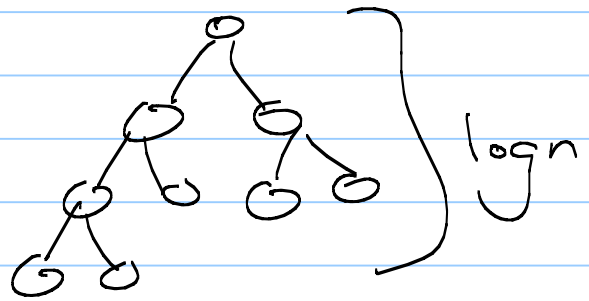
```
TreeSearch(k, v):
    Input: key k to search for & a node v in tree
    Output: A node w of T s.t. either w holds
            key k or w is the leaf node where
            k would belong

    if key(v) == k
        return v
    else if key(v) < k
        TreeSearch(k, v->left)
    else if key(v) > k
        TreeSearch(k, v->right)
    else // at a leaf
        return v
```

How long does searching take?

Spend $O(1)$ time at each node
recurse at most $O(h)$ times,
where h = height of tree
(visit at most one node per level)

$\Rightarrow O(h)$ time





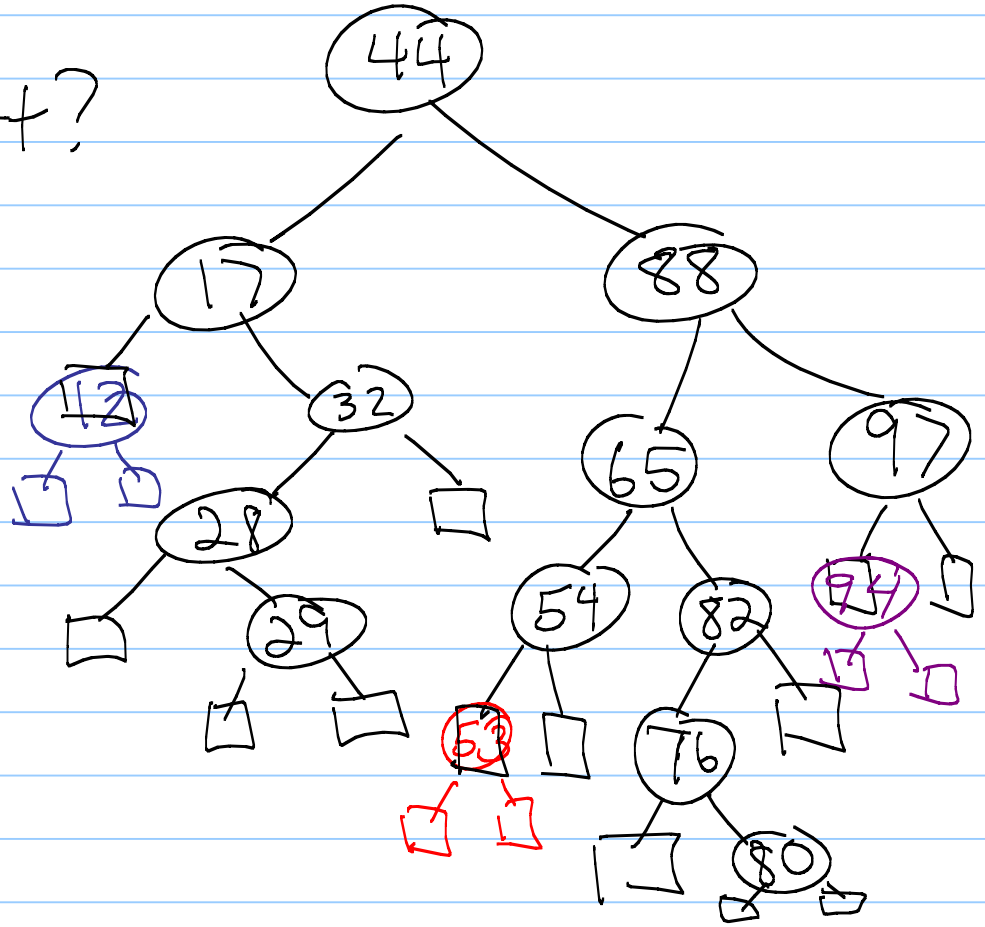
Update Operations

- How do we insert?

insert(53)

insert(94)

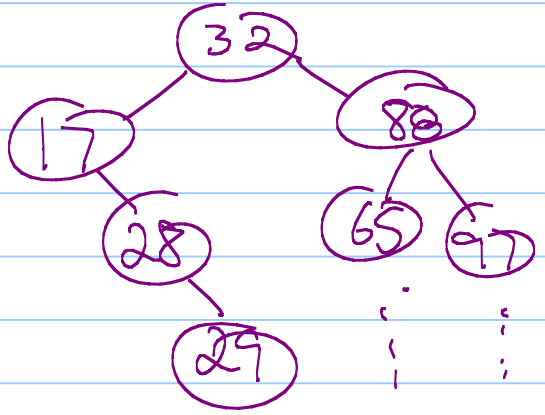
insert(12)

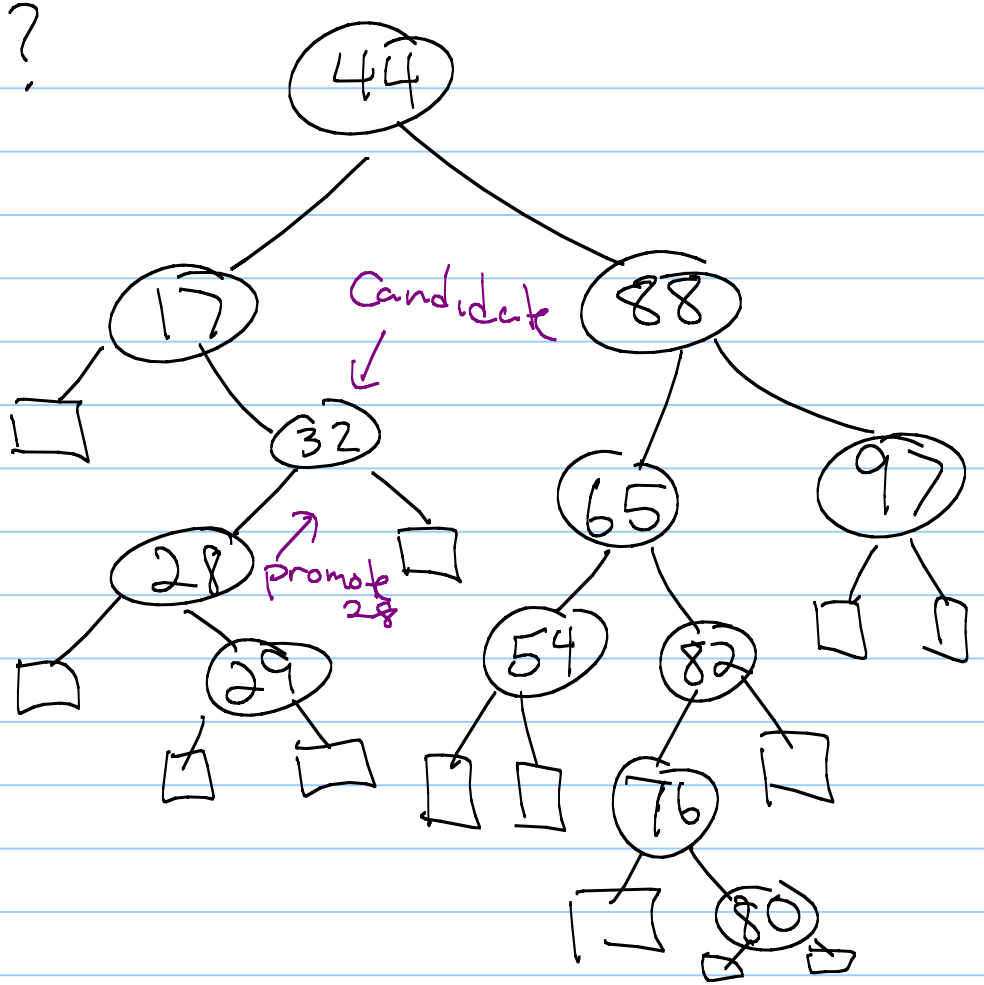


How do we remove?

2 cases: node to remove has a leaf child or doesn't

delete(44)



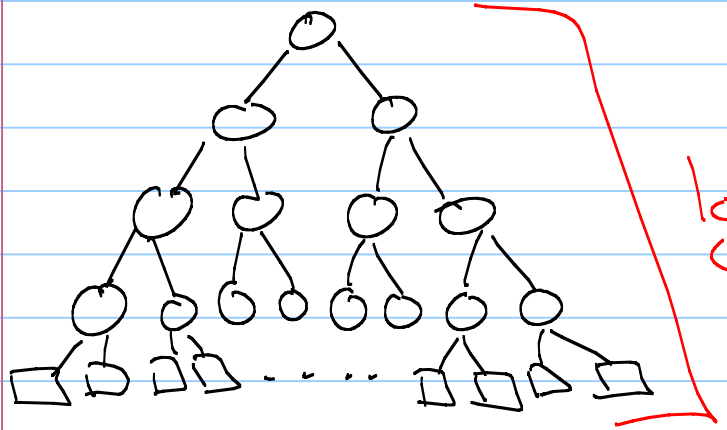


How long do insert & remove take?

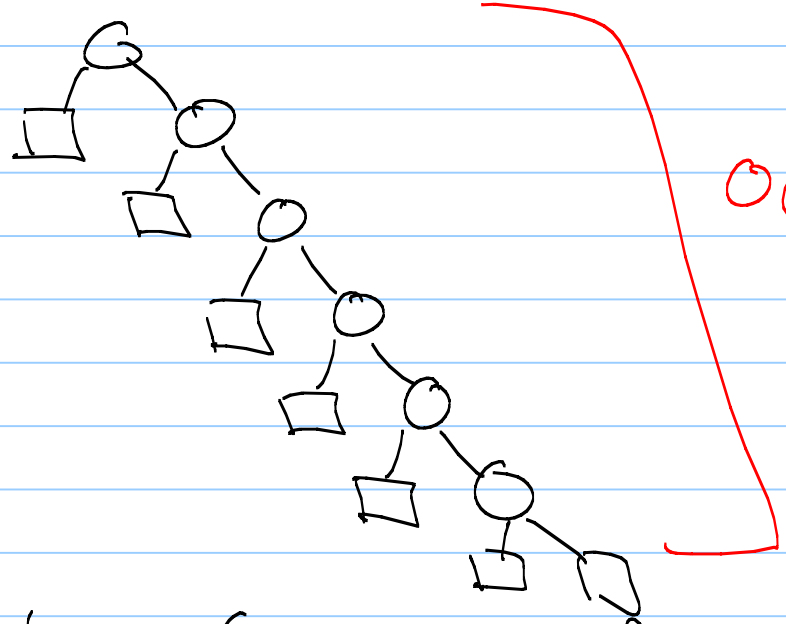
O(h) O(h)

TreeSearch time + O(1) to add or remove a single node

Best versus worst case:



log n



O(n)

How deep is each tree (in terms of n = # of nodes in tree)?

A better tree structure:
AVL trees

attempt to keep the tree balanced

- For every internal node v of T, the height of v's children differ by at most 1.

$O(h)$
$=$
$O(\log n)$