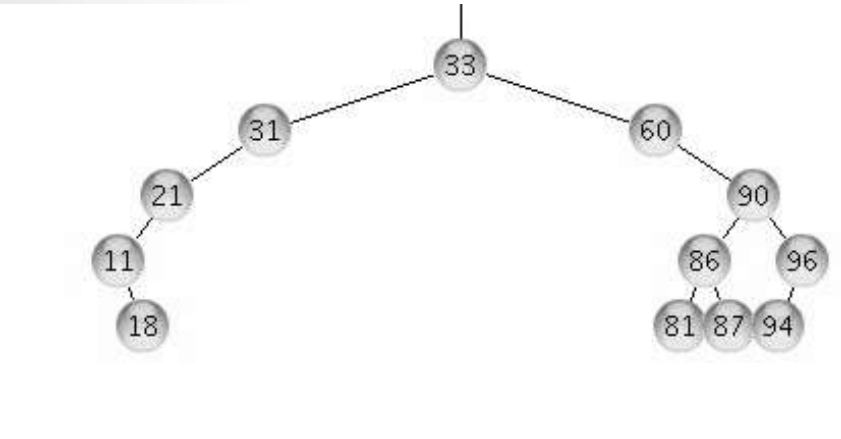
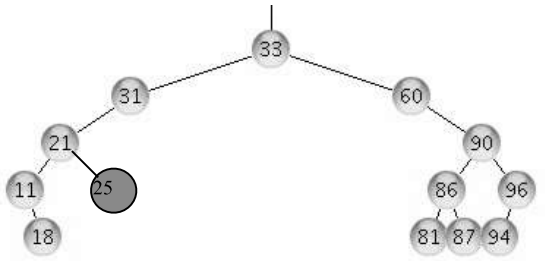
Inserting in a BST

- insert 25



Inserting in a BST

- insert 25
  - There is only one place where 25 can go



- //create and insert node with key k in th[illegible]
- void insert (v, k) {

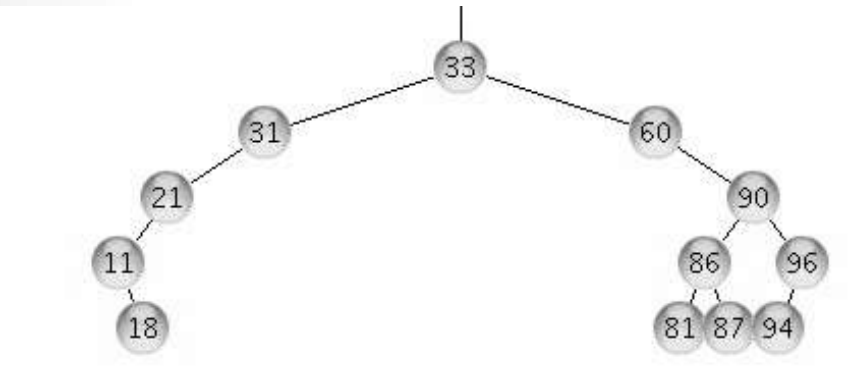
```
    //this can only happen if inserting in an empty tree
    if (v == null) return new BSTNode(k);
    if (k <= v.getData()) {
         if (v.left() == null) {
             //insert node as left child of v
             u = new BSTNode(k);
             v.setLeft(u);
        } else {
            return insert(v.left(), k);
        }
    } else //if (v.getData() > k) {
        ...
    }
}
```

Inserting in a BST

- Analysis:
  - similar with searching
  - traverses a path from the root to the inserted node
  - O(depth of inserted node)
  - this is O(h), where h is the height of the tree

Deleting in a BST

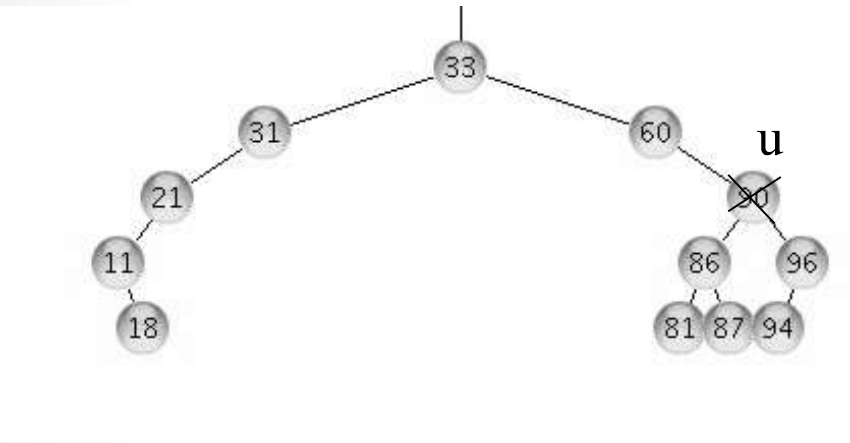
- delete 87
- delete 21
- delete 90



- case 1: delete a
  - if x is left of its parent, set parent(x).left = null
  - else set parent(x).right = null
- case 2: delete a node with one child
  - link parent(x) to the child of x
- case 2: delete a node with 2 children
  - ??

Deleting in a BST

- delete 90



- copy in u 94 and delete 94
  - the left-most child of right(x) ← node has <=1 child
- or
- copy in u 87 and delete 87
  - the right-most child of left(x) ← node has <=1 child

Deleting in a BST

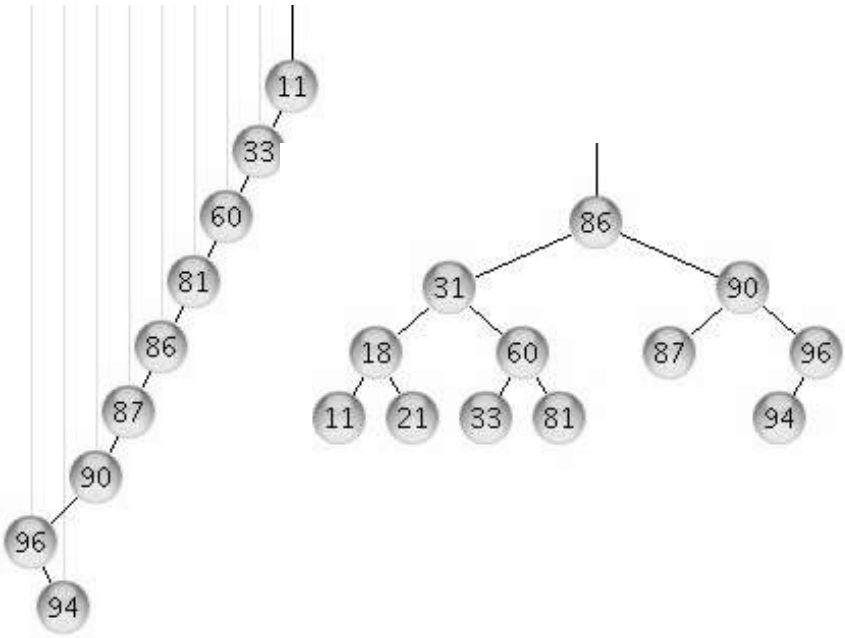
- Analysis:
  - traverses a path from the root to the deleted node
  - and sometimes from the deleted node to its left-most child
  - this is O(h), where h is the height of the tree

BST performance

- Because of search property, all operations follow one root-leaf path
  - insert: O(h)
  - delete: O(h)
  - search: O(h)

- We know that in a tree of n nodes
  - h >= lg (n+1) - 1
  - h <= n-1

- So in the worst case h is O(n)
  - BST insert, search, delete: O(n)
  - just like linked lists/arrays



BST performance

- worst-case scenario
  - start with an empty tree
  - insert 1
  - insert 2
  - insert 3
  - insert 4
  - ...
  - insert n

- it is possible to maintain that the height of the tree is Theta(lg n) at all times
  - by adding additional constraints
  - perform rotations during insert and delete to maintain these constraints

- Balanced BSTs: h is Theta(lg n)
  - Red-Black trees
  - AVL trees
  - 2-3-4 trees
  - B-trees