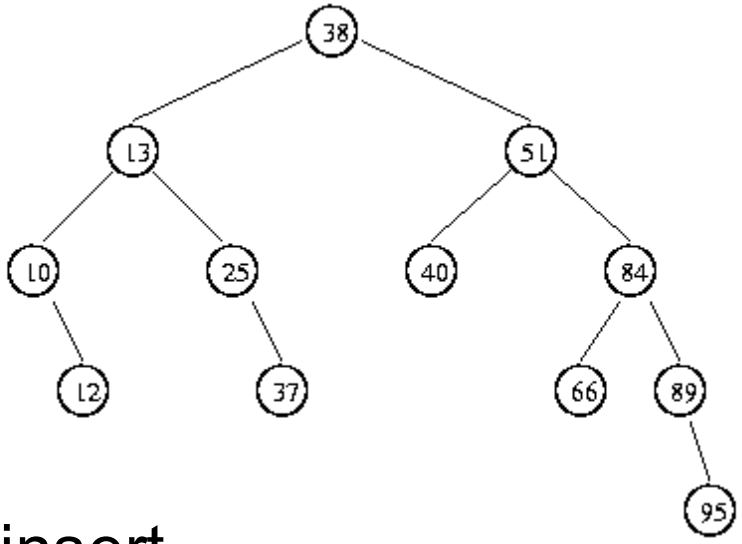
Announcements

MP5 available, due 10/30, 11:59p. EC due 10/23, 11:59p.

http://www.qmatica.com/DataStructures/Trees/AVL/AVLTree.html



insert

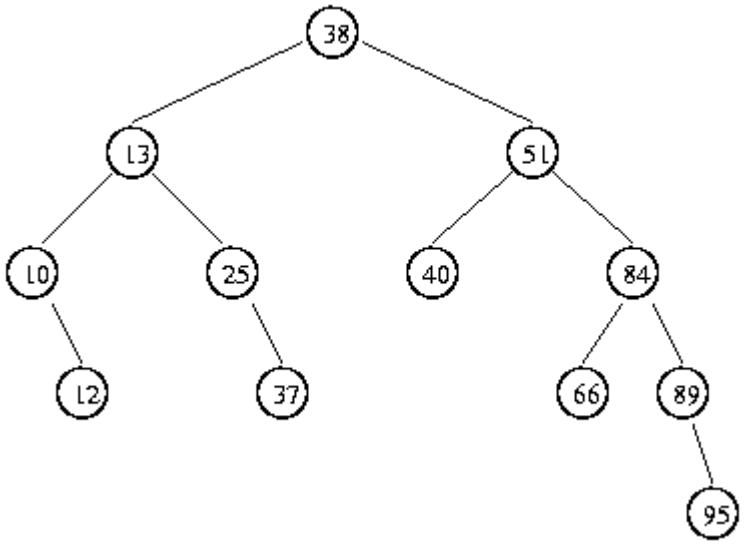
remove

find

traverse

```
template <class K, class D>
class Dictionary{
public:
// ctor for empty tree, + …
private:
   struct treeNode{
      D data;
      K key;
      treeNode * left;
      treeNode * right;
   };
   treeNode * root
};
```

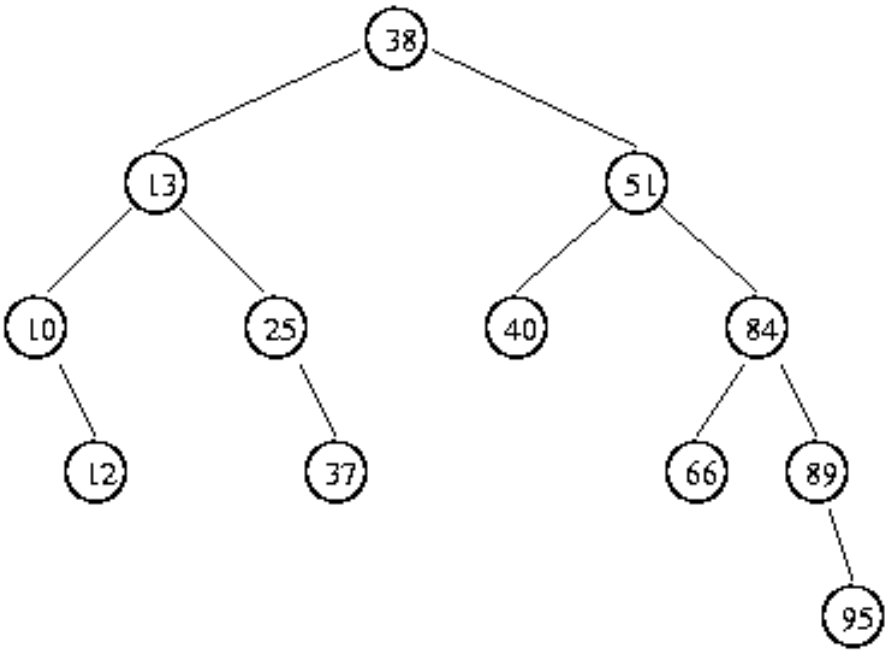
Binary Search Tree - Remove



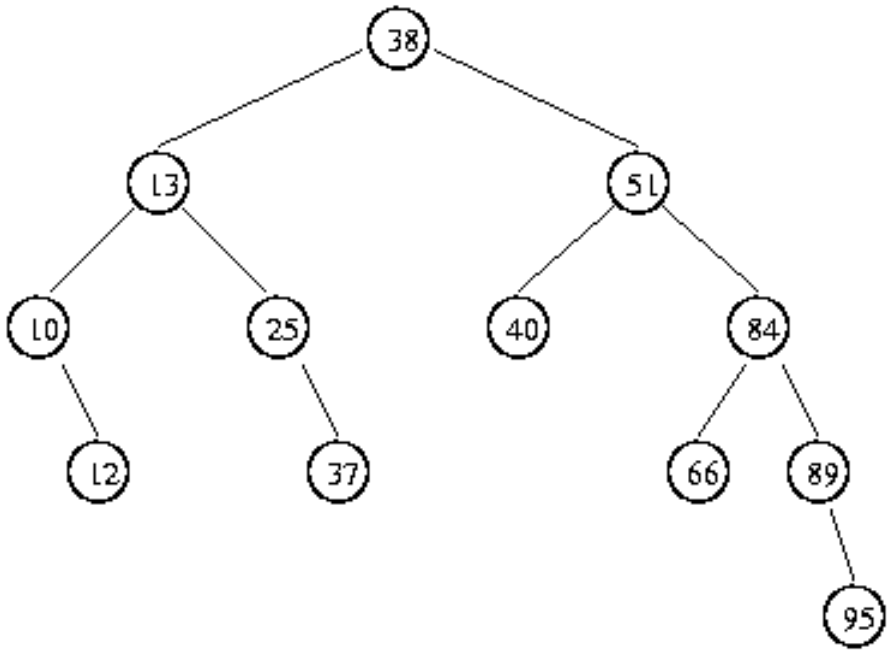

```
void BST<K>::remove(treeNode * & cRoot, const T & d) {
   if (cRoot != NULL) {
      if (cRoot->key == d)
        doRemoval(cRoot);
      else if (k < cRoot->key)
        remove(cRoot->left, d);
      else
        remove(cRoot->right, d);
  }
}
```

Binary Search Tree - Remove

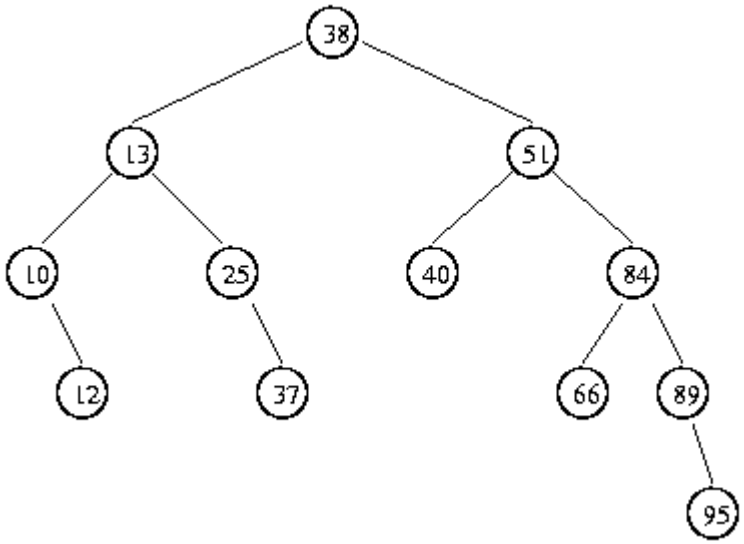
```
T.remove(37);
T.remove(10);
```




```
T.remove(13);
```



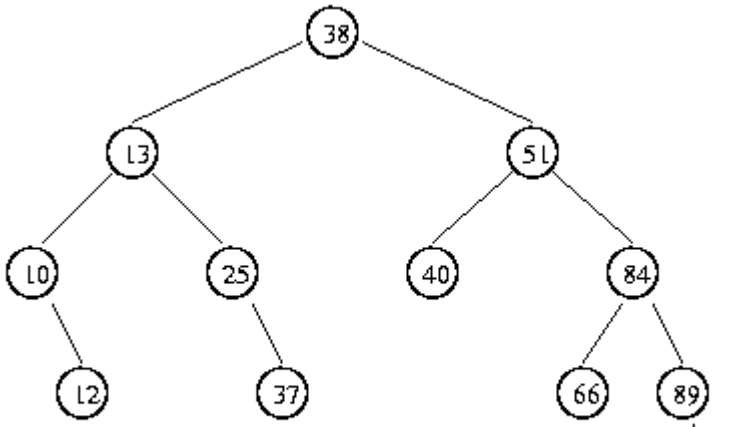
Binary Search Tree - Remove




```
void BST<K>::re
   if (cRoot !=
      if (cRoot
        doRem
      else if (
        remov
      else
        remov
  }
}
```

```
void BST<K>::doRemoval(treeNode * & cRoot) {
   if ((cRoot->left == NULL) && (cRoot->right == NULL))
      ____ChildRemove(cRoot);
   else if ((cRoot->left != NULL) && (cRoot->right != NULL))
      ____ChildRemove(cRoot);
   else
      ____ChildRemove(cRoot);
}
```

Binary Search Tree - Remove



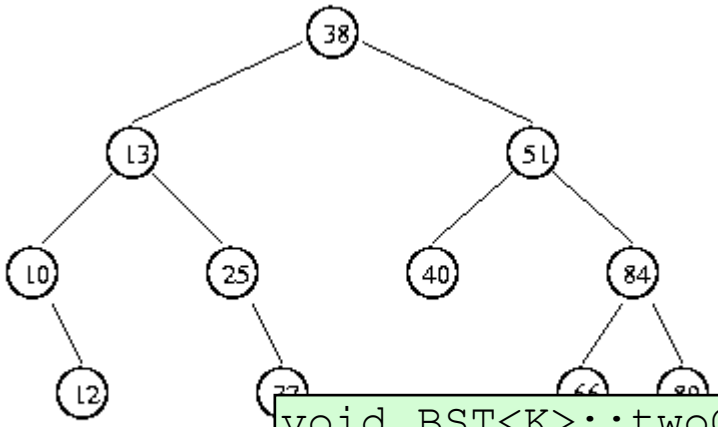

```
void Dictionary
   if (cRoot !=
      if (cRoot
        doRem
      else if (
        remov
      else
        remov
  }
}
```

```
void Diction
   if ((cRoo
      noChil
   else if (
      twoChildRemove(cRoot);
   else
      oneChi
}
```

```
void BST<K>::noChildRemove(treeNode * & cRoot) {
   treeNode * temp = cRoot;
   cRoot = NULL;
   delete temp;
}
```

```
void BST<K>::oneChildRemove(treeNode * & cRoot) {
   treeNode * temp = cRoot;
   if (cRoot->left == NULL) cRoot = cRoot->right;
   else cRoot = cRoot->left;
   delete temp;
}
```

Binary Search Tree - Remove



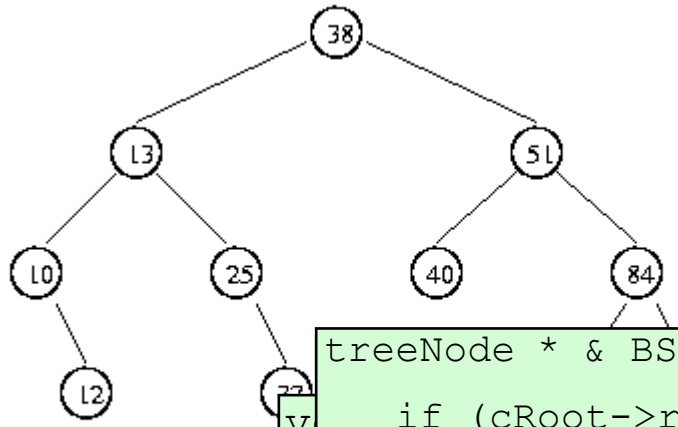

```
void BST<K>::twoChildRemove(treeNode  * & cRoot) {
   treeNode *  iop = rightMostChild(cRoot->left);
   cRoot->key = iop->key;
   doRemoval(iop);
}
```

```
void BST<K>::re
   if (cRoot !=
      if (cRoot
        doRem
      else if (
        remov
      else
        remov
  }
}
```

```
voi
   i
   else if
      twoC
   else
      oneChildRemove(cRoot);
}
```

```
   }
      delete temp;
   }
```

Binary Search Tree - Remove




```
treeNode * & BST<K>::rightMostChild(treeNode * & cRoot) {
   if (cRoot->right == NULL) return cRoot;
   else return rightMostChild(cRoot->right);
}
```

```
   doRemoval(iop);
}
```

```
void BST<K>::re
   if (cRoot !=
      if (cRoot
        doRem
      else if (
        remov
      else
        remov
  }
}
```

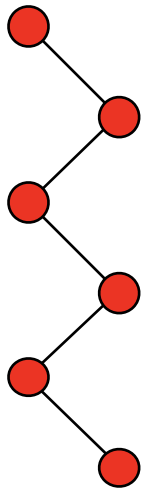
```
voi
   i
   else if
      twoC
   else
      oneChildRemove(cRoot);
}
```

```
   }
      delete temp;
   }
```

Binary Search Tree - miscellaneous characteristics and analysis

```
BST<int> myT;
myT.insert(2);
myT.insert(7);
myT.insert(15);
myT.insert(22);
myT.insert(28);
…
```

Give a sequence of inserts that result in a tree that looks like:



How many “bad” n-item trees are there?