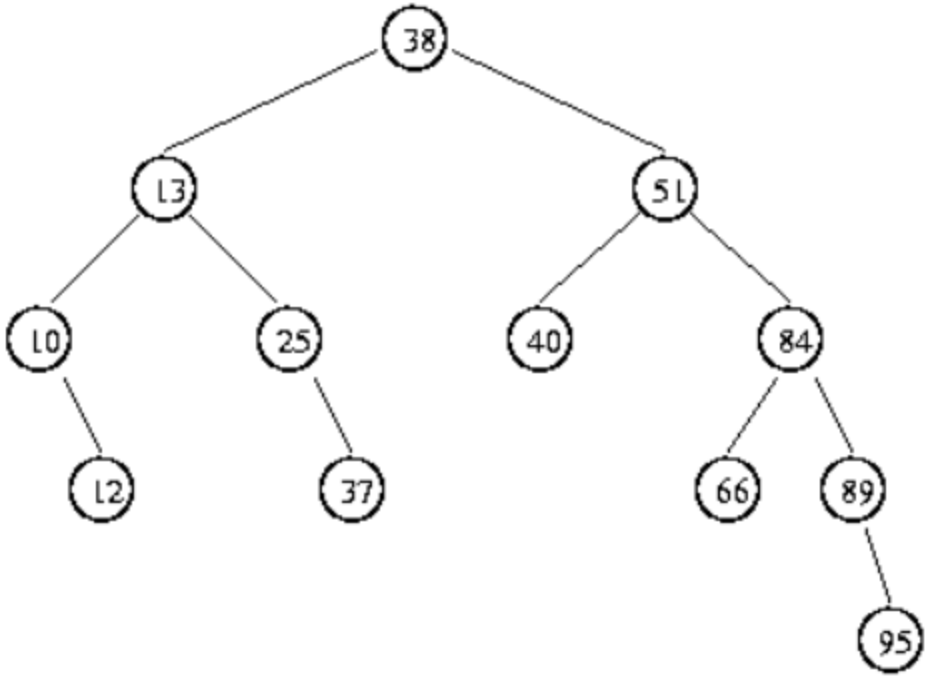
Announcements:

Exam 4 starts

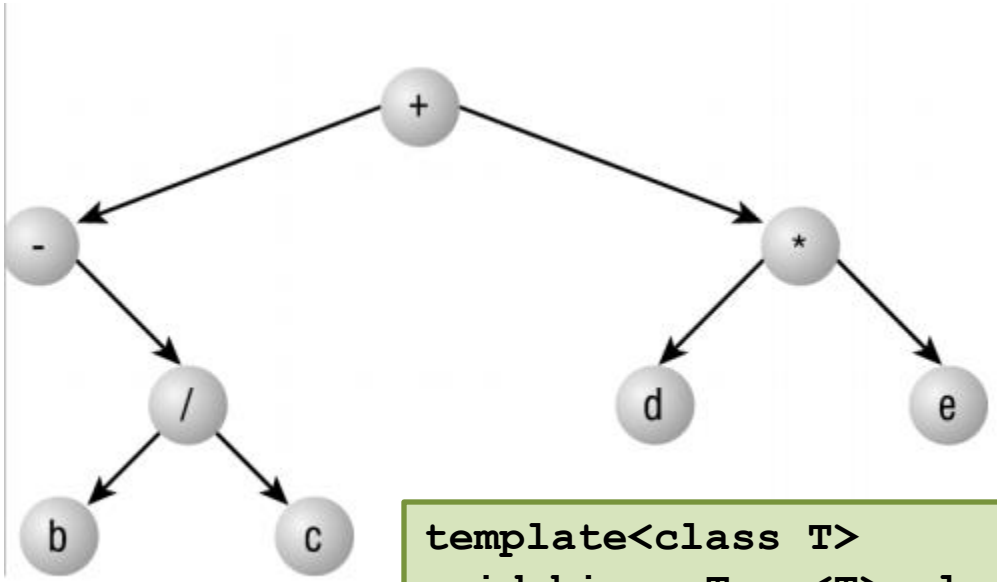
Sunday, 3/12

```
template<class T>
void binaryTree<T>::clear(treeNode * croot){
  if (root != null) {
    clear(root->left);
    clear(root->right);
    delete root;
    root = null;
  }
}
```

Tree Traversals:



Something completely different…



```
template<class T>
void binaryTree<T>::levelOrder(treeNode * croot) {



}
```

Running Time

ADT: Dictionary

Suppose we have the GPAs for every course…

Course ID	Average GPA
CS 225	3.08
ECE 391	2.93
CS 241	2.73
MATH 241	2.70
ECE 220	2.88

…and we want to retrieve a GPA, given a course.

Other key/value examples:

UIN -> Advising Record

Course Number -> Schedule info

Color -> BMP

Vertex -> Set of incident edges

Flight number -> arrival information

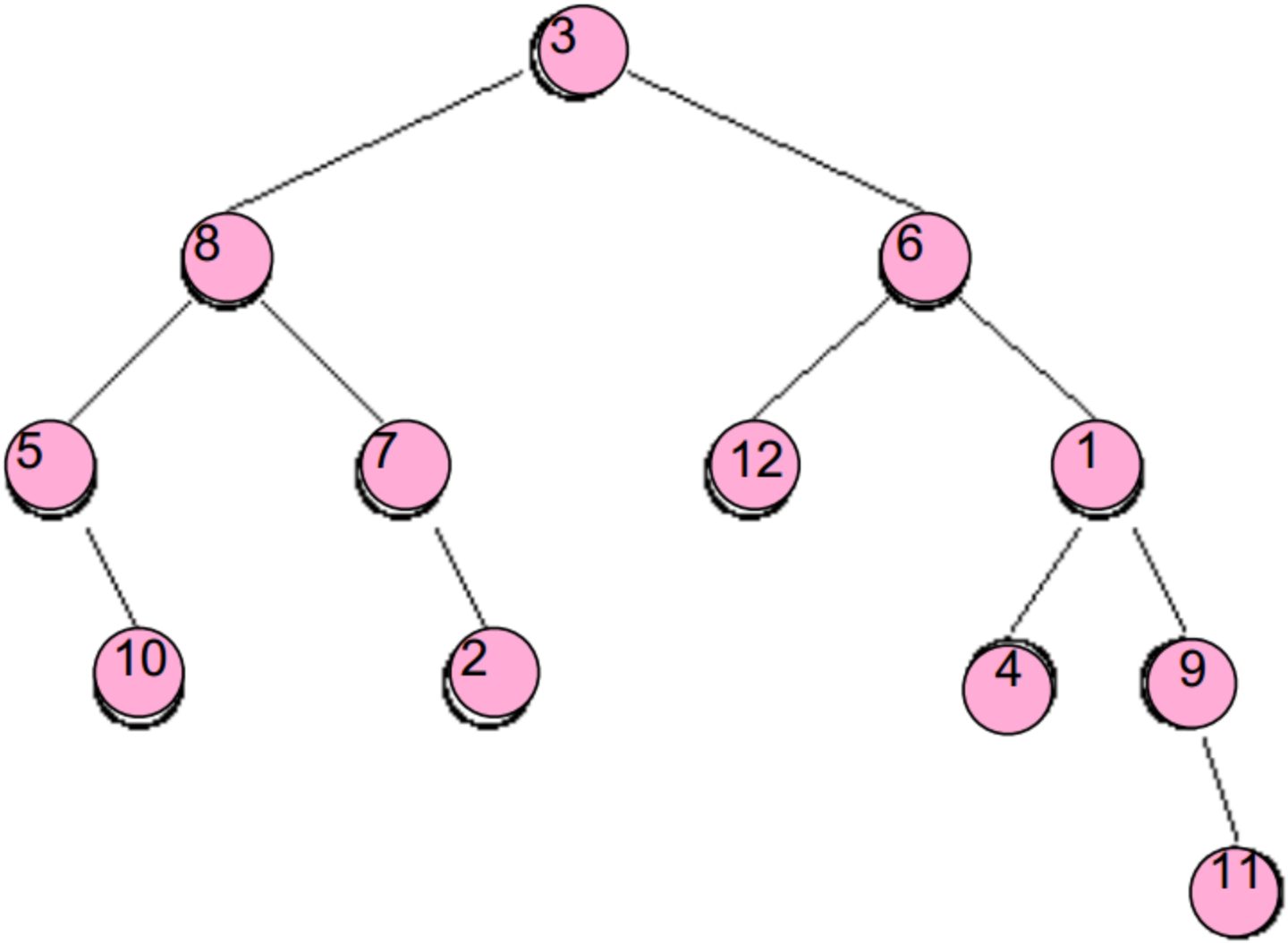
URL -> html page

A dictionary is a structure supporting the following:

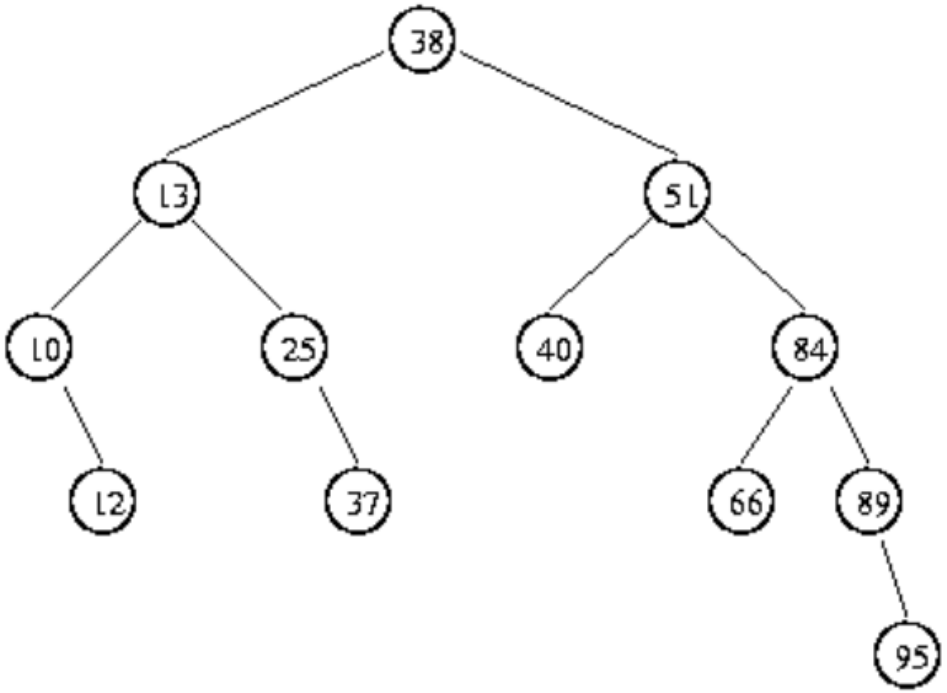
```
void insert(K & k, D & d);
void remove(K & k);
D    find  (K & k);
```

Dictionary using Binary Trees as a search structure

Find Me:



Binary _____ Tree

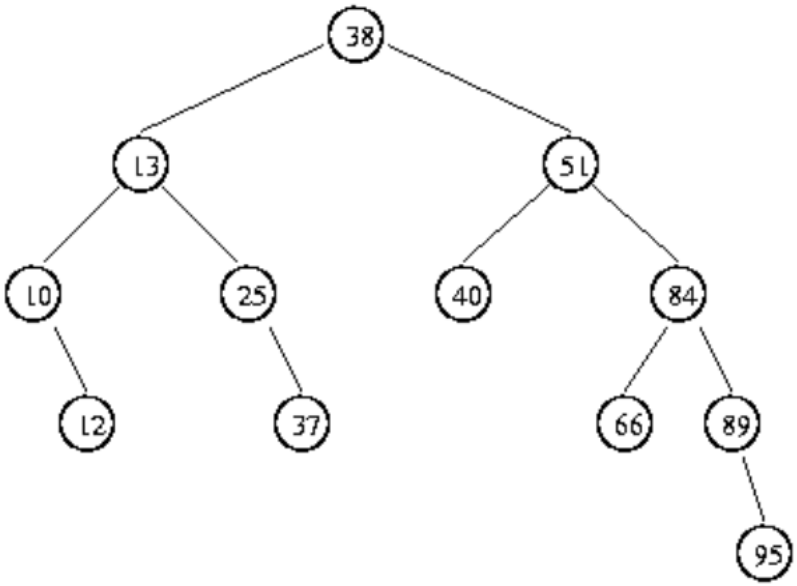


A Binary Search Tree (BST) is a binary tree, T, such that:

- _____, OR
- $T = \{r, T_L, T_R\}$ and
  - $x \in T_L \rightarrow$ _____
  - $x \in T_R \rightarrow$ _____

and

Dictionary AST: BST Implementation



insert

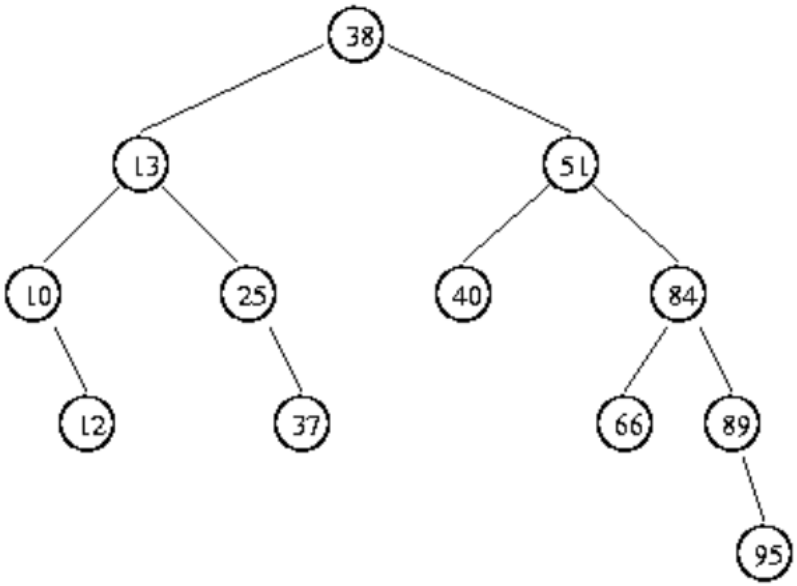
remove

find

traverse

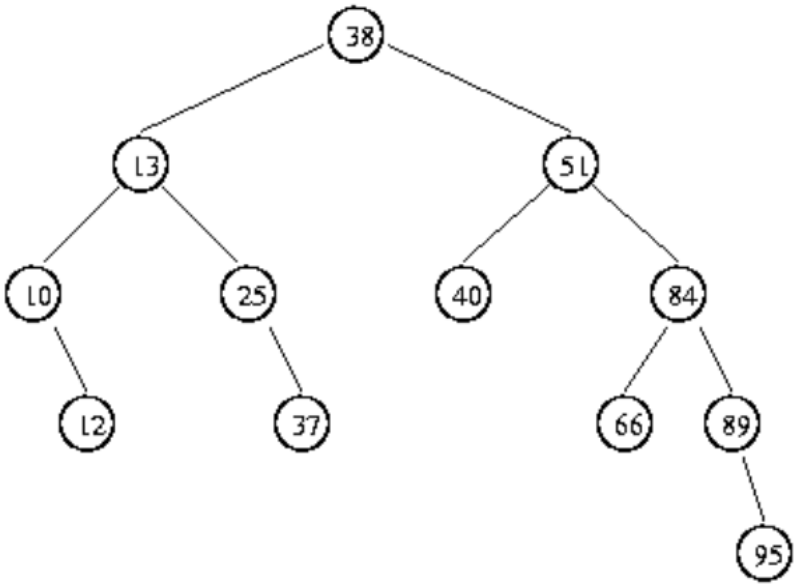
```
template <class K, class D>
class Dictionary {
  public:
    // constructor for empty tree.
  private:
    struct treeNode {
      D data;
      K key;
      treeNode * left;
      treeNode * right;
    };
    treeNode * root
};
```

BST - find




```
_____________________________________
   (treeNode *cRoot, const K & key)
{
  if (cRoot == NULL)




  else if (cRoot->key == key)




  else if




  else




}
```

BST - insert




```
   (treeNode *cRoot, const K & key,
    const D & data)
{
  if (cRoot == NULL)




  else if (cRoot->key == key)




  else if (key < cRoot->key)




  else



}
```