



CS 225

Data Structures

October 5 – Binary Search Tree (BST)

G Carl Evans



Traversal vs. Search

Traversal vs. Search:

- **Traversal** visits every node in the tree exactly once.
- **Search** finds one element in the tree.

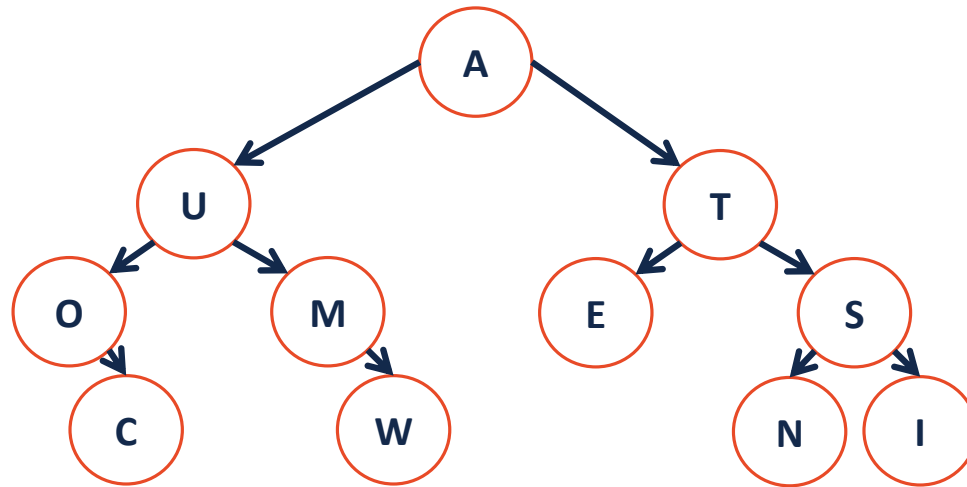


Search: Breadth First vs. Depth First

Strategy: Depth First Search (DFS) / Traversal

Strategy: Breadth First Search (BFS) / Traversal

Search Running Times on a Binary Tree





Dictionary ADT

Data is often organized into key/value pairs:

Word → Definition

Course Number → Lecture/Lab Schedule

Node → Incident Edges

Flight Number → Arrival Information

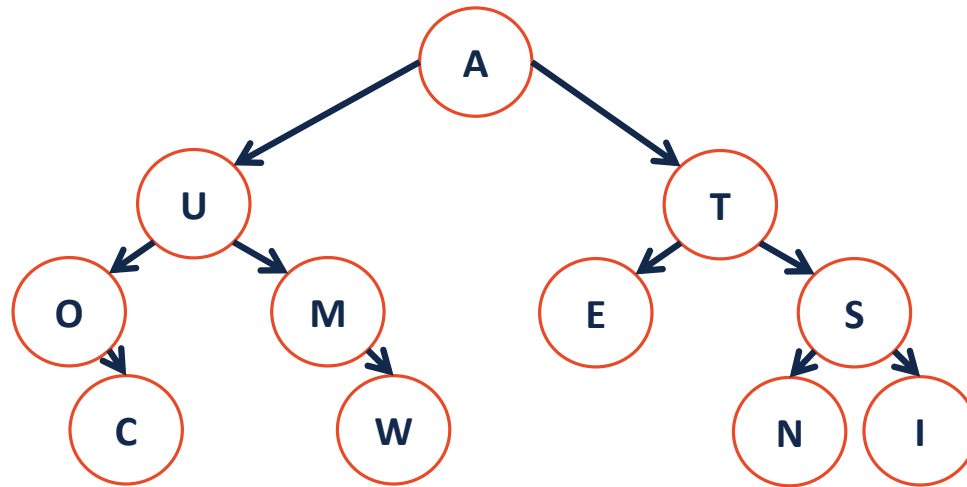
URL → HTML Page

...

Dictionary.h

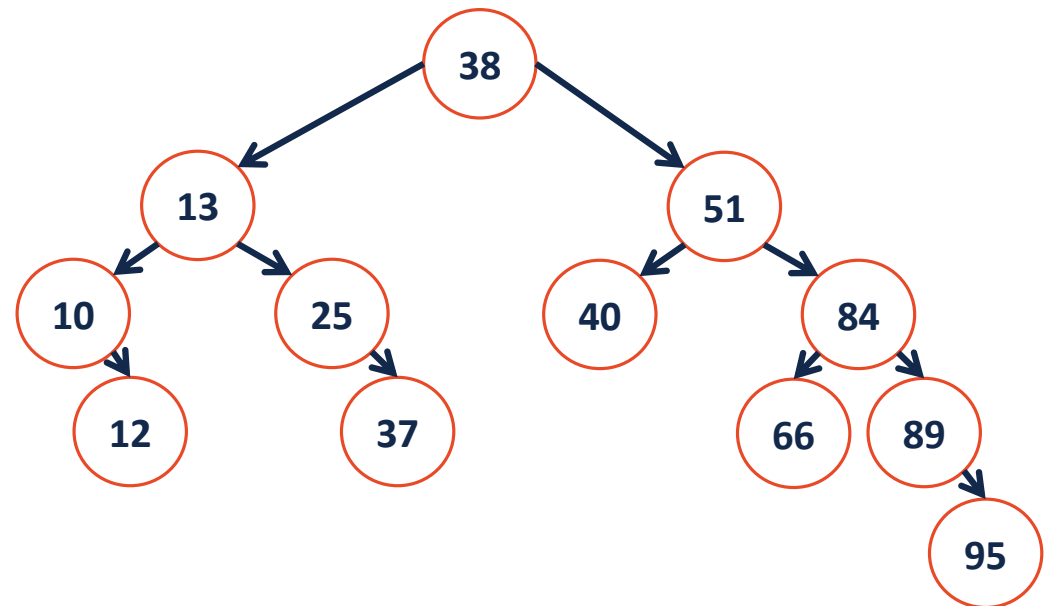
```
1 #pragma once
2
3
4 class Dictionary {
5     public:
6
7
8
9
10
11
12
13
14
15     private:
16
17
18
19 };
20
21 #endif
22
```

Binary Tree as a Search Structure



Binary _____ Tree (BST)

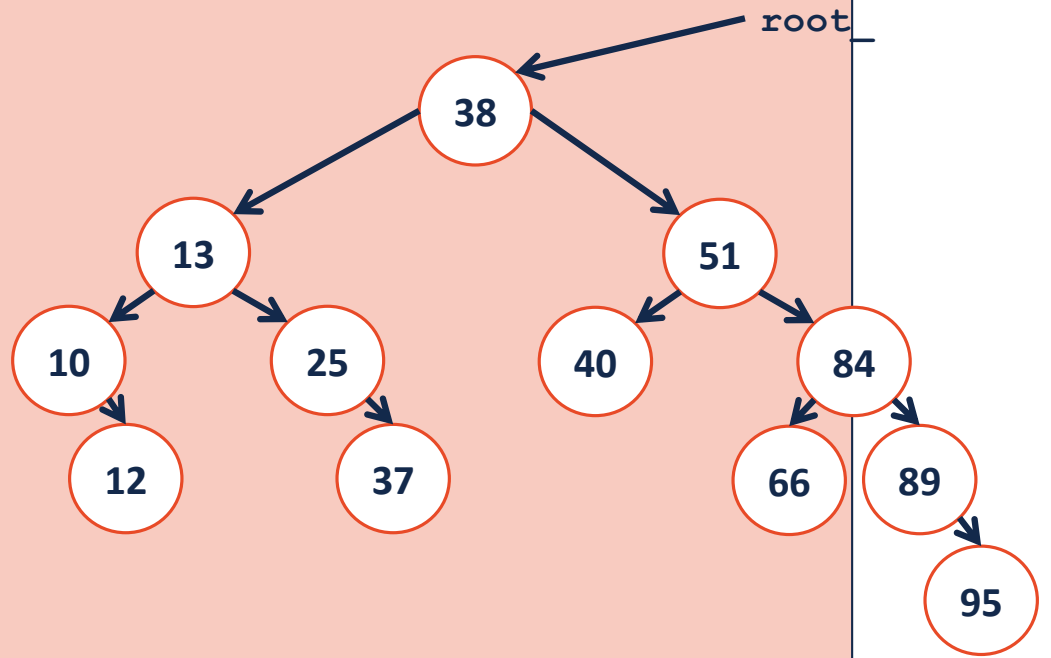
A **BST** is a binary tree **T** such that:



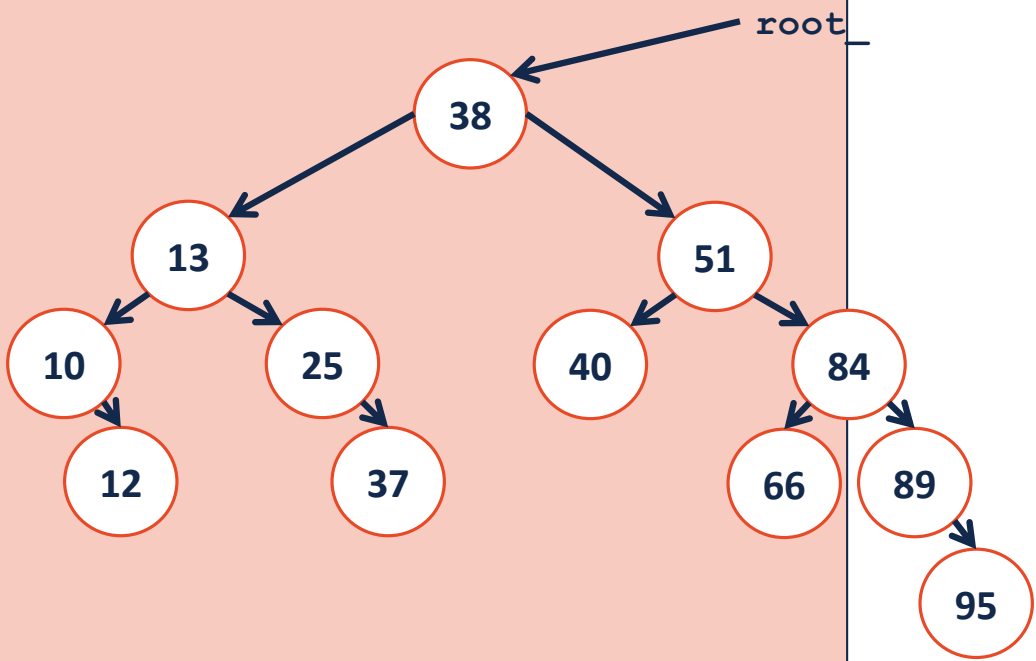
BST.h

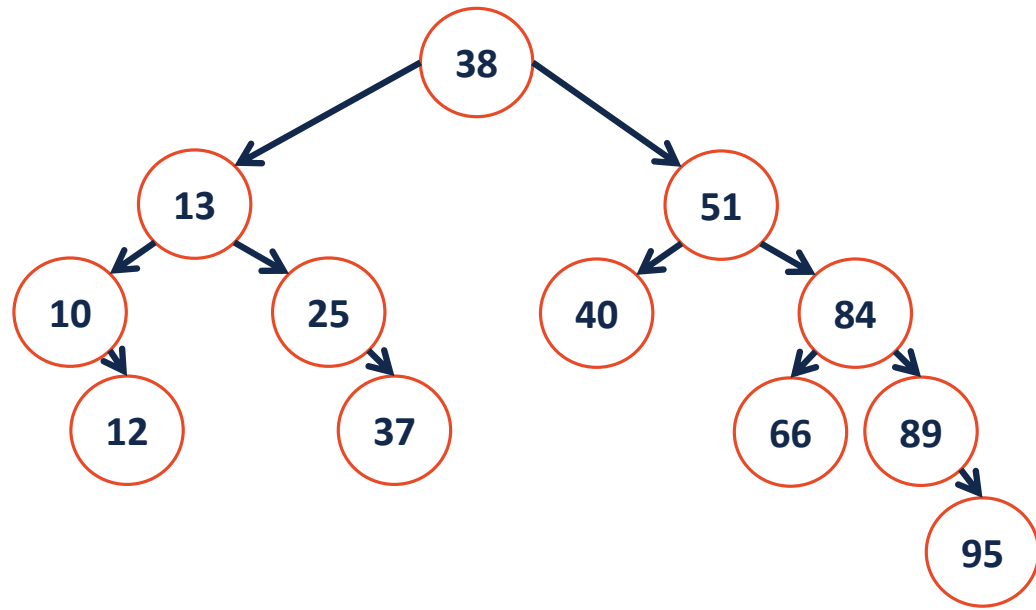
```
1 #pragma once
2
3 template <typename K, typename V>
4 class BST {
5     public:
6         BST();
7         void insert(const & K key, V value);
8         V remove(const K & key);
9         V find(const K & key) const;
10
11
12     private:
13         struct TreeNode {
14             TreeNode *left, *right;
15             K key;
16             V value;
17             TreeNode(const K & k, const V & v) : key(k), value(v),
18                 left(NULL), right(NULL) { }
19         };
20
21         TreeNode *head_;
22     };
```

```
1  template<typename K, typename V>
2  _____ find(const K & key) {
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26 }
```

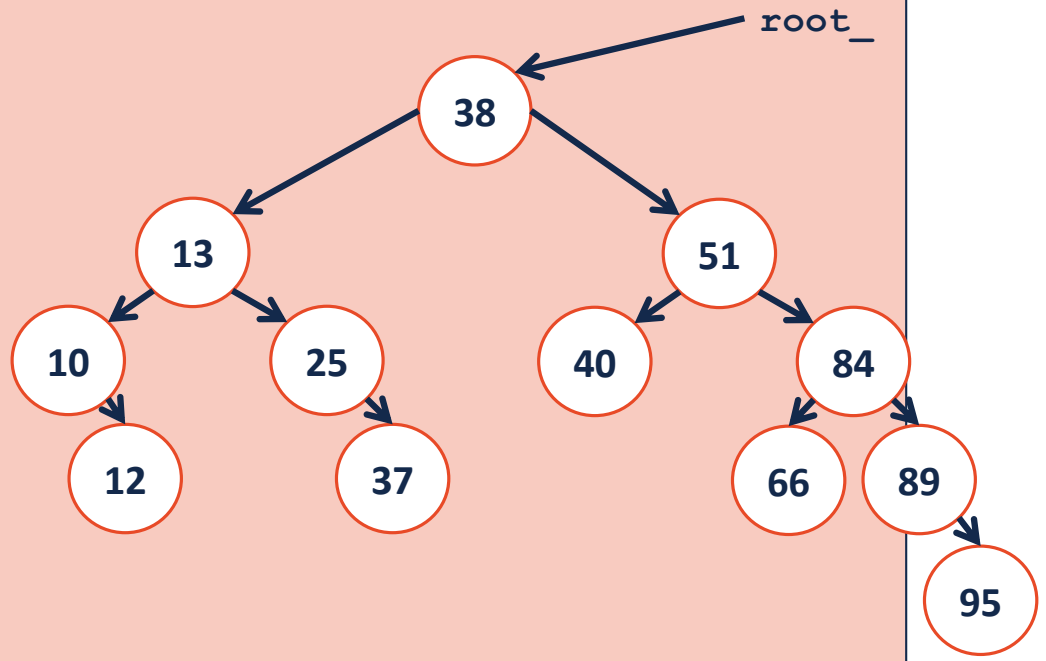


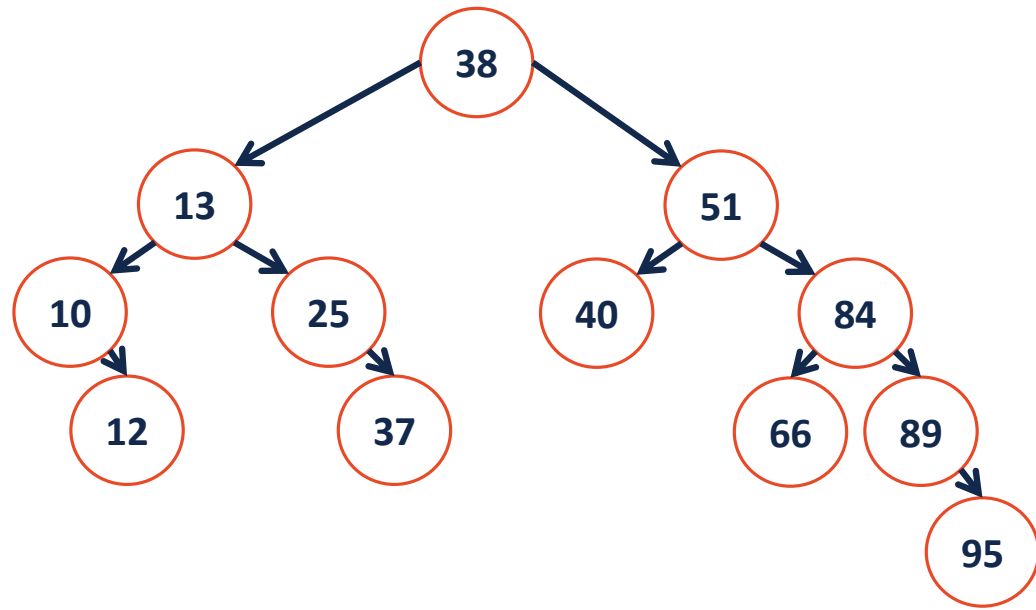
```
1 template<typename K, typename V>
2     _____ _find(TreeNode *& root, const K & key) {
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26 }
```

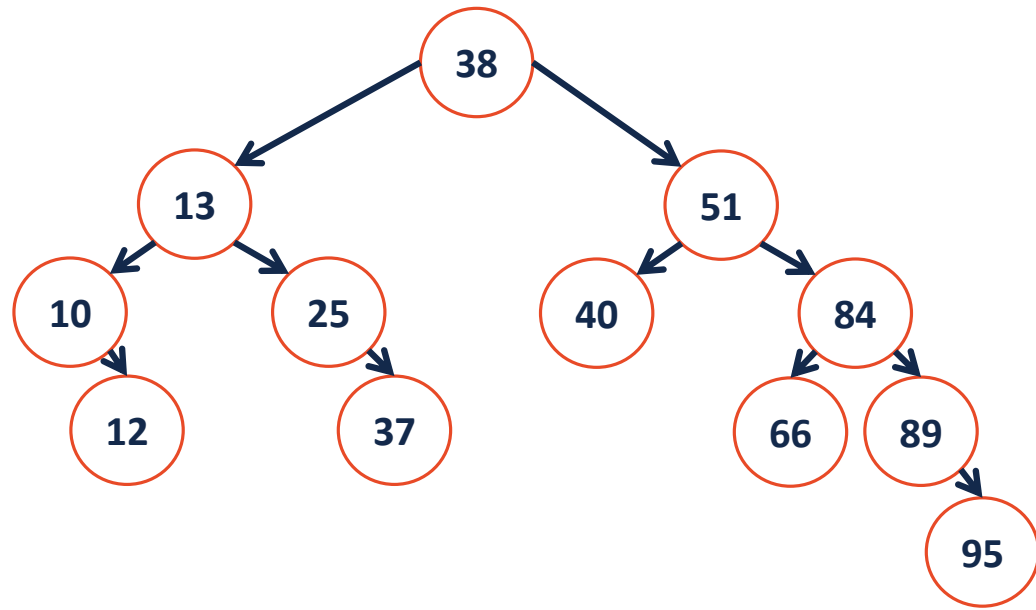




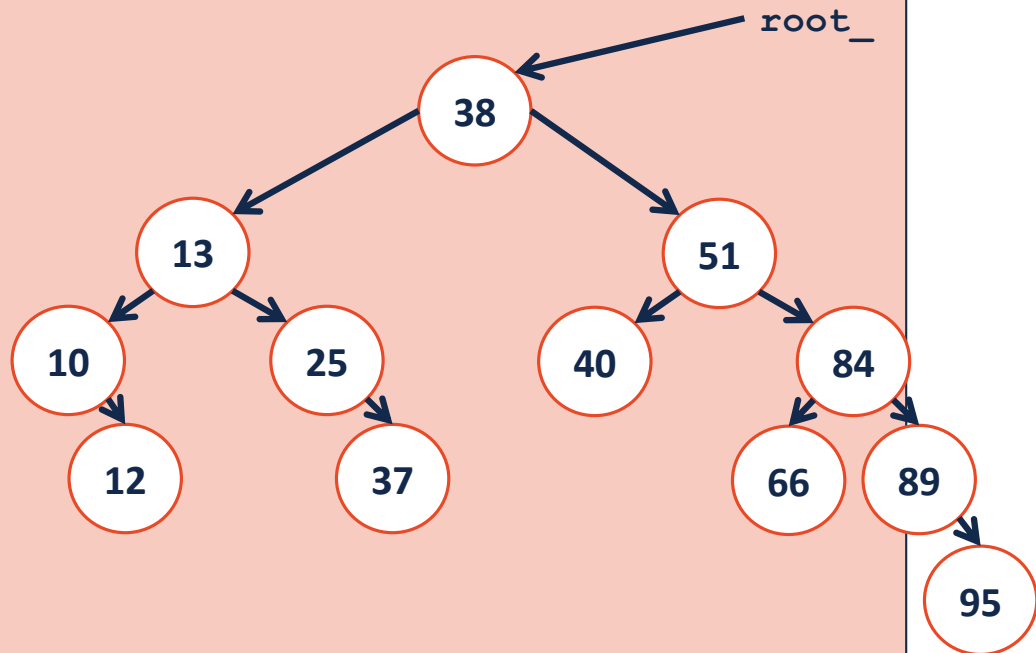
```
1 template<typename K, typename V>
2 _____ _insert(TreeNode *& root, const K & key) {
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26 }
```

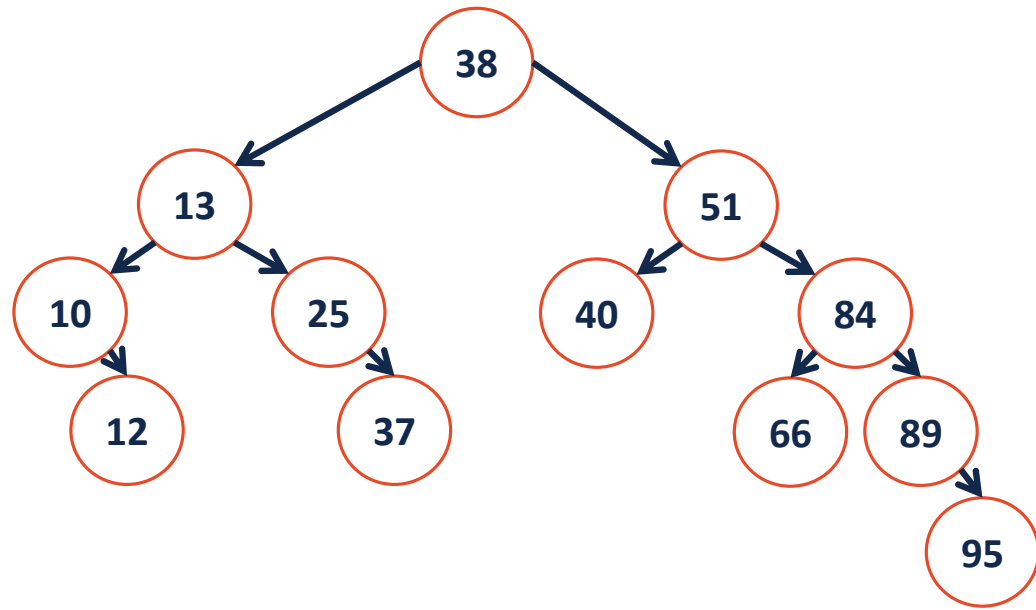




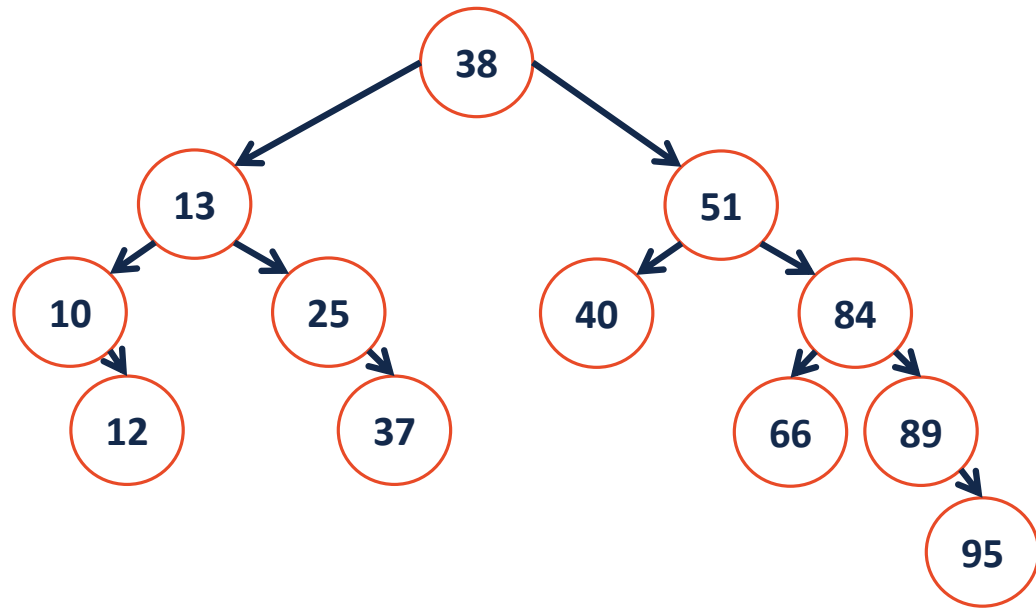


```
1  template<typename K, typename V>
2  _____ _remove(TreeNode *& root, const K & key) {
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26 }
```

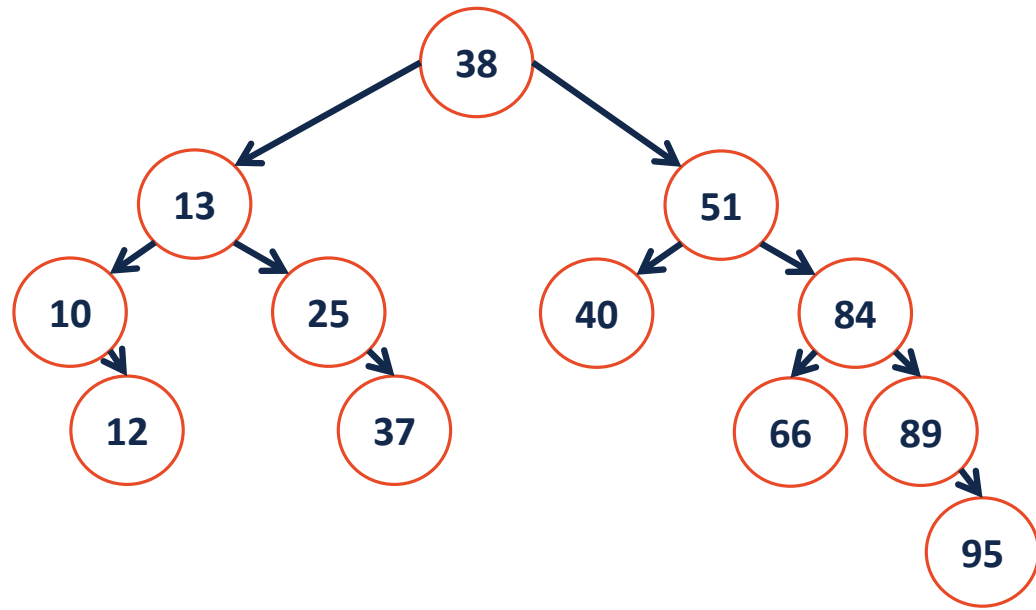




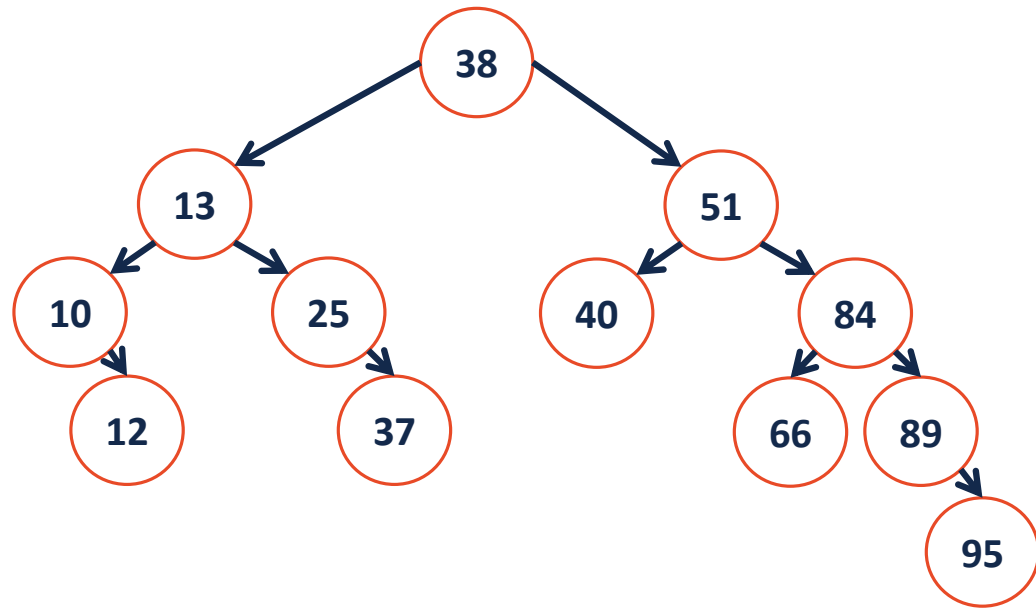
remove (25) ;



`remove (40) ;`



`remove(10);`



`remove (13) ;`