CS 225

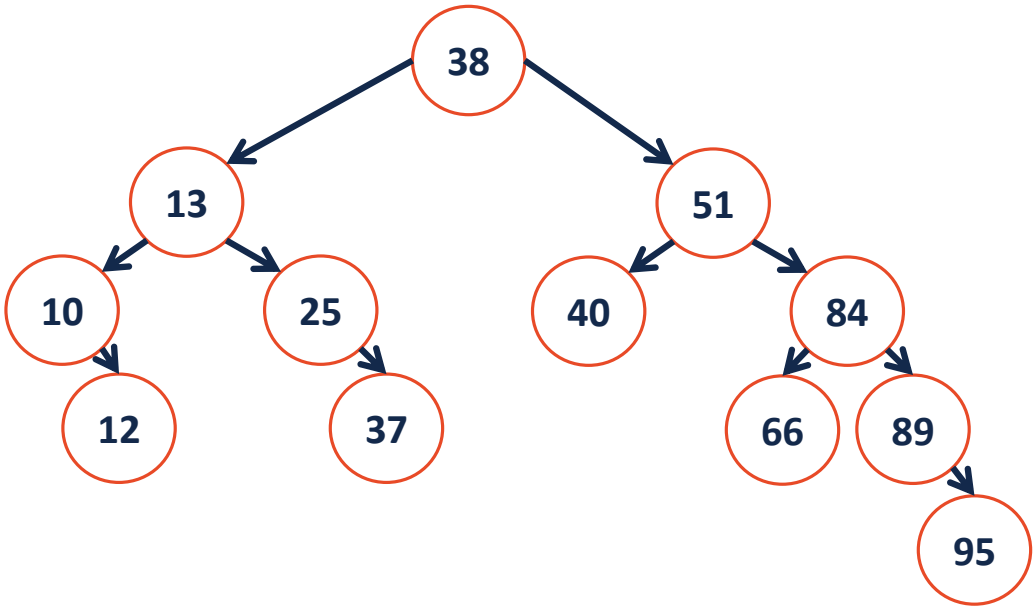
Data Structures

February 12 – BST Implementation

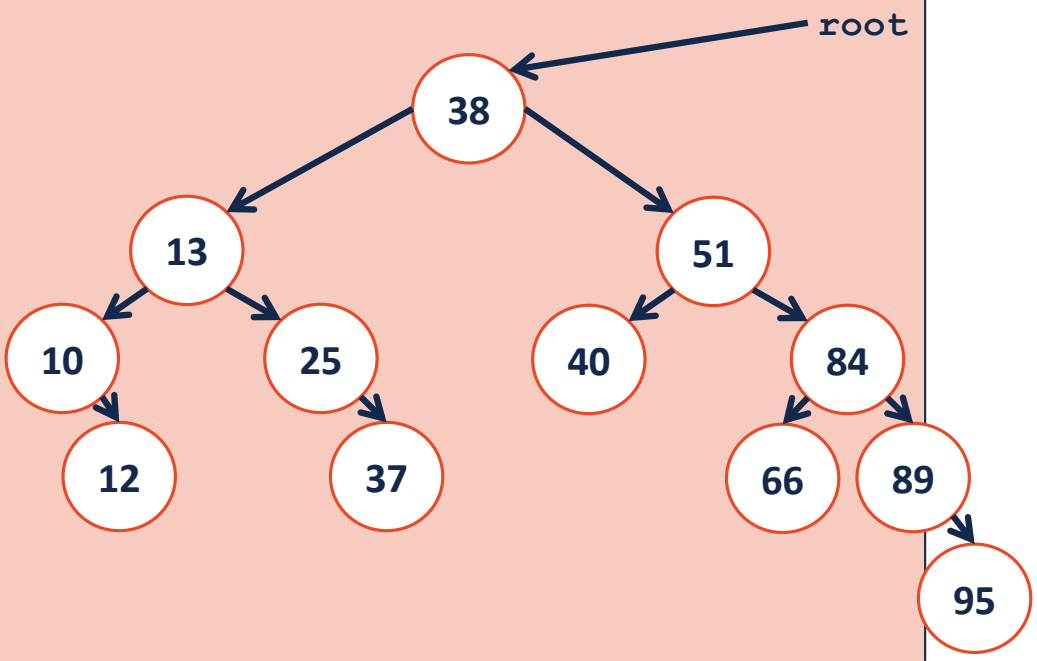
G Carl Evans

BST.h

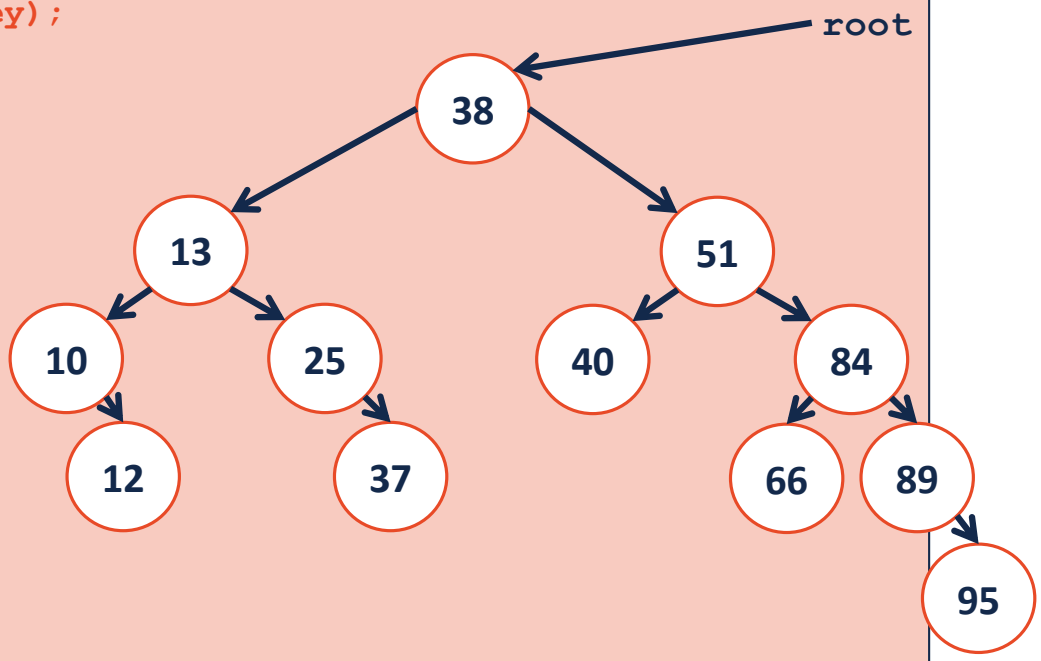
```
 1 #pragma once
 2
 3 template <class K, class V>
 4 class BST {
 5   public:
 6     BST();
 7     void insert(const K key, V value);
 8     void remove(const K & key);
 9     V find(const K & key) const;
10     TreeIterator begin();
11     TreeIterator end();
12
13   private:
14
15     struct TreeNode {
16       TreeNode *left_;
17       K key_;
18       V value_;
19       TreeNode *right_;
20     };
21     TreeNode *root_;
22 };
```

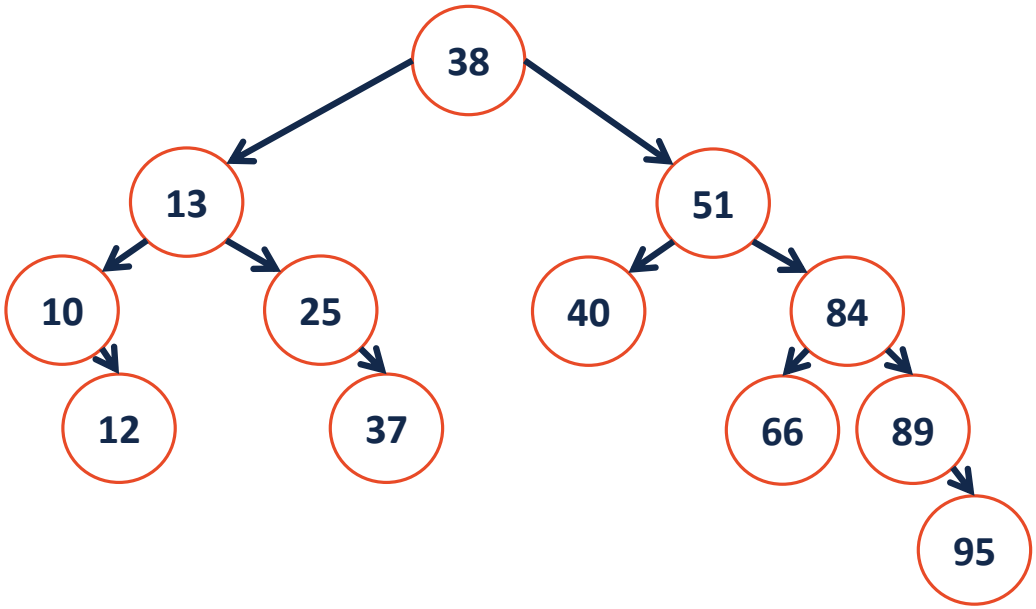

38
13
51
10
25
40
84
12
37
66
89
95

```
1  template<class K, class V>
2  TreeNode * &_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
```

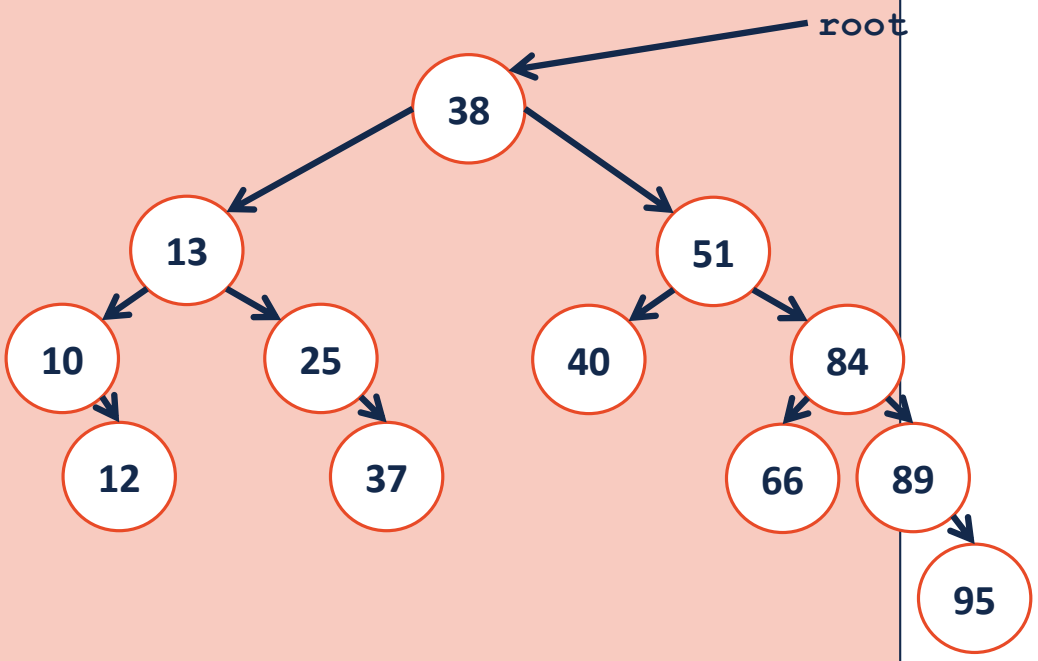


```
1  template<class K, class V>
2  TreeNode * &_find(TreeNode *& root, const K & key) {
3    if( root == nullptr )
4      { return root; }
5    if( root->key == key )
6      { return root; }
7    if( key < root->key ) {
8      return _find(root->left, key);
9    } else {
10     return _find(root->right, key);
11   }
12 }
13
14
15
16
17
18
19
20
21
22
23
24
25
26
```



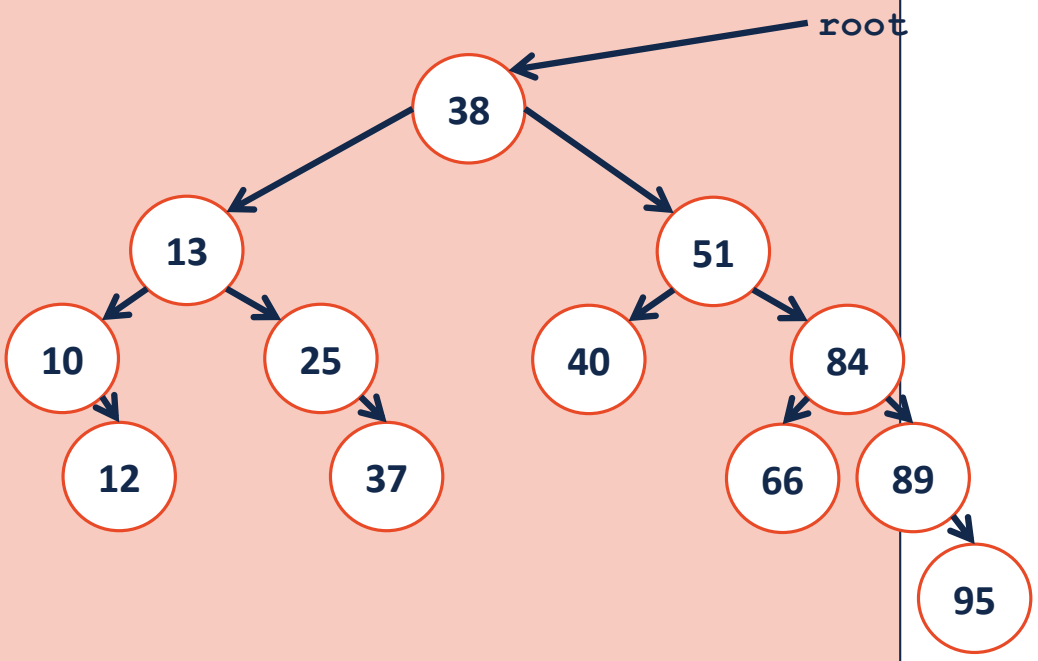

38
13
51
10
25
40
84
12
37
66
89
95

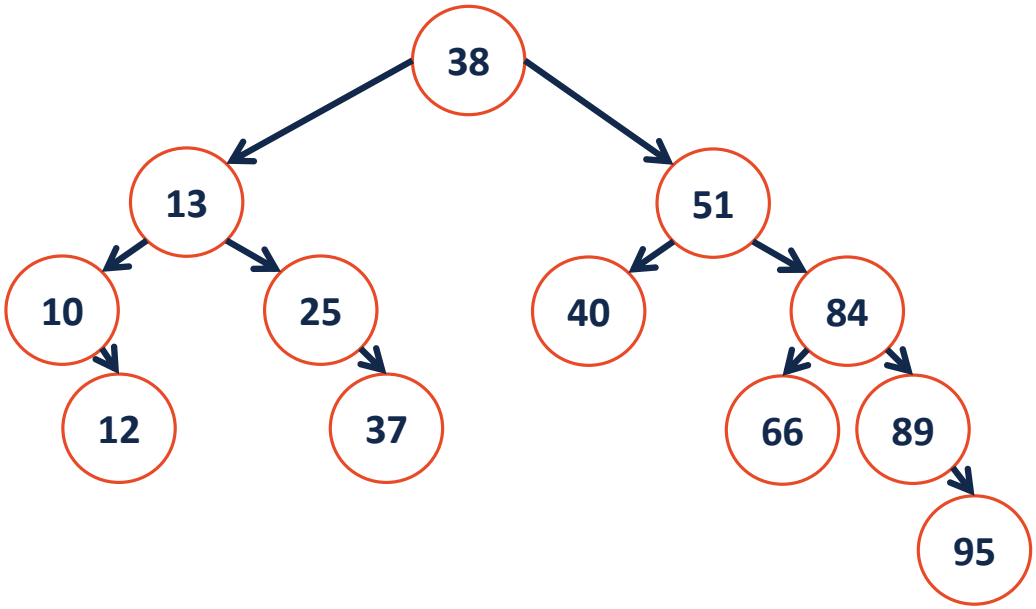
```
1  template<class K, class V>
2  void _insert(TreeNode *& root, const K & key, const V &value) {
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17 }
18
19
20
21
22
23
24
25
26
```

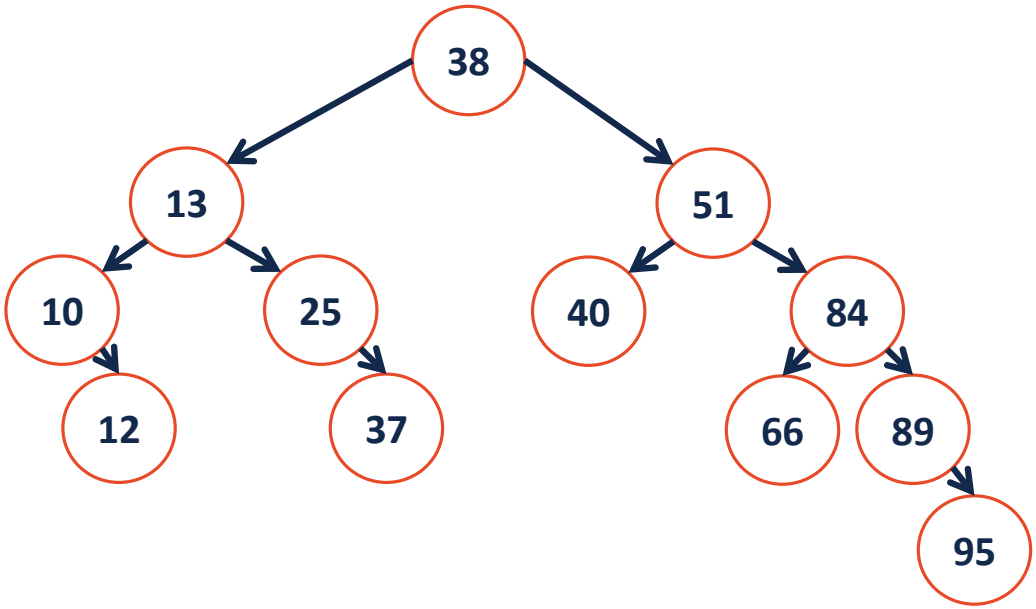


```
1  template<class K, class V>

2  void _insert(TreeNode *& root, const K & key, const V &value) {
3    TreeNode *&location = _find(root, key);
4    location = new TreeNode( key, value);
5  }
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
```

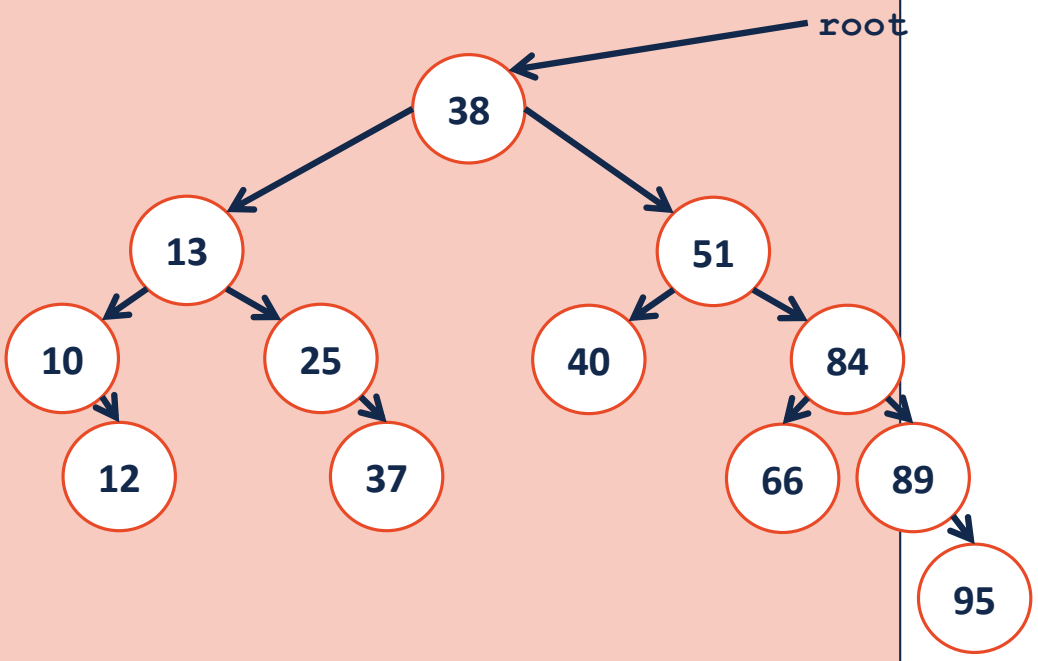


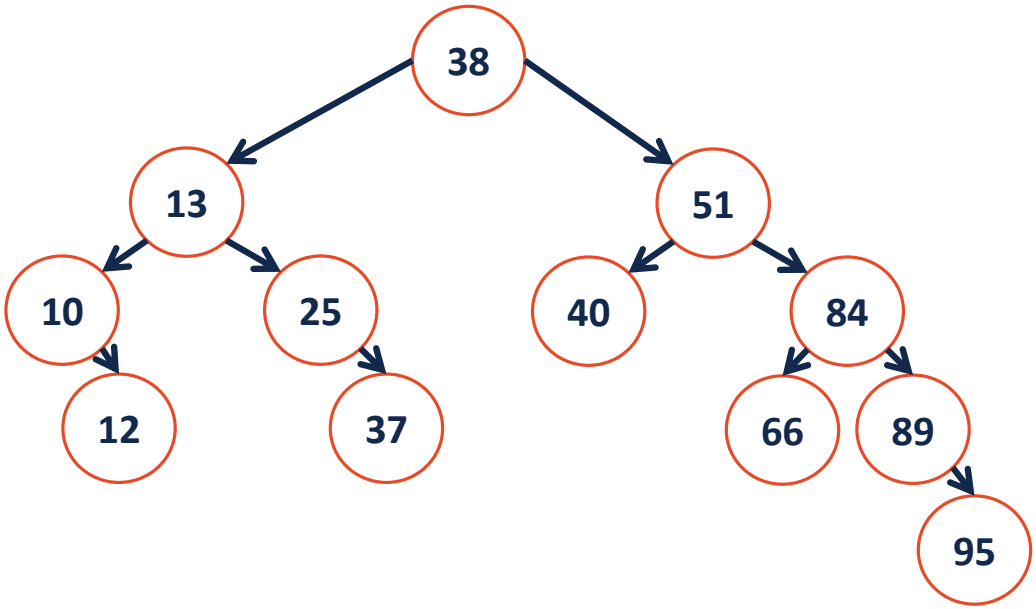

38
13
51
10
25
40
84
12
37
66
89
95




```
remove(40);
```

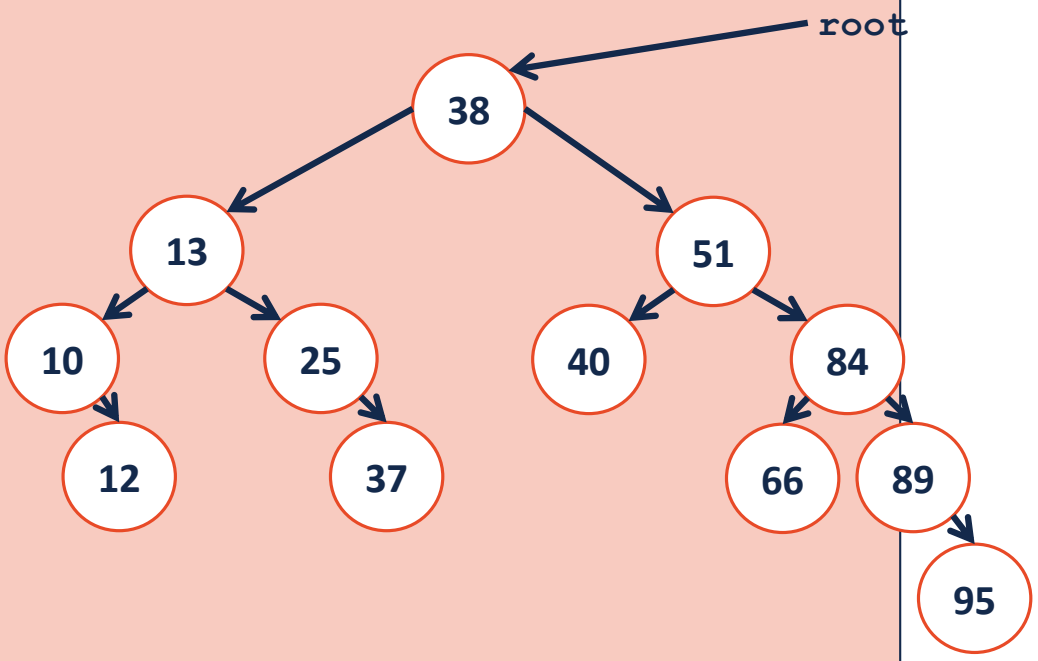
```
1  template<class K, class V>
2  _____________________________ _remove(TreeNode *& root, const K & key) {
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26 }
```

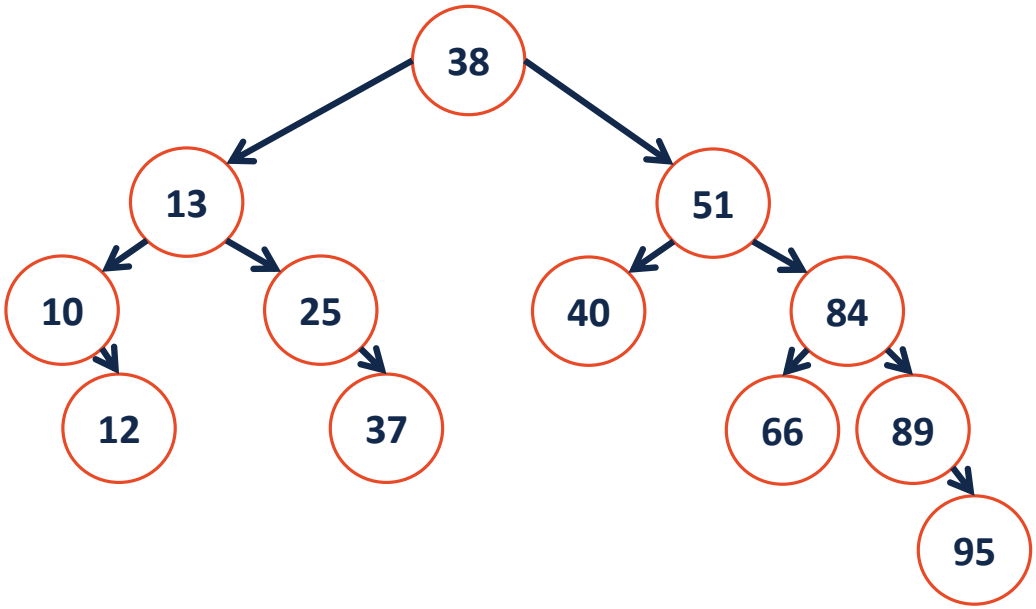




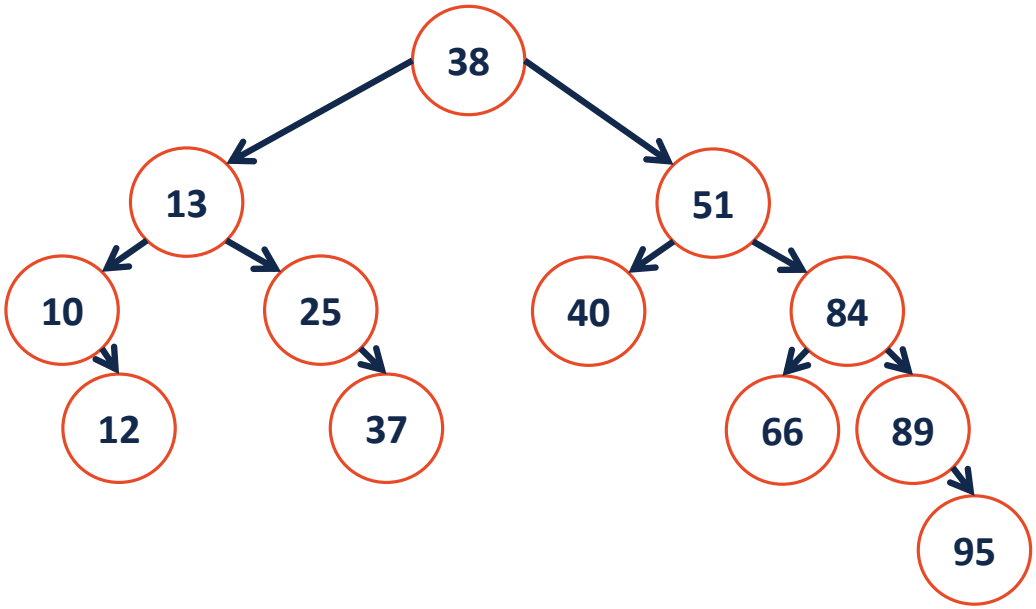

```
remove(25);
```

```
1  template<class K, class 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 }
```

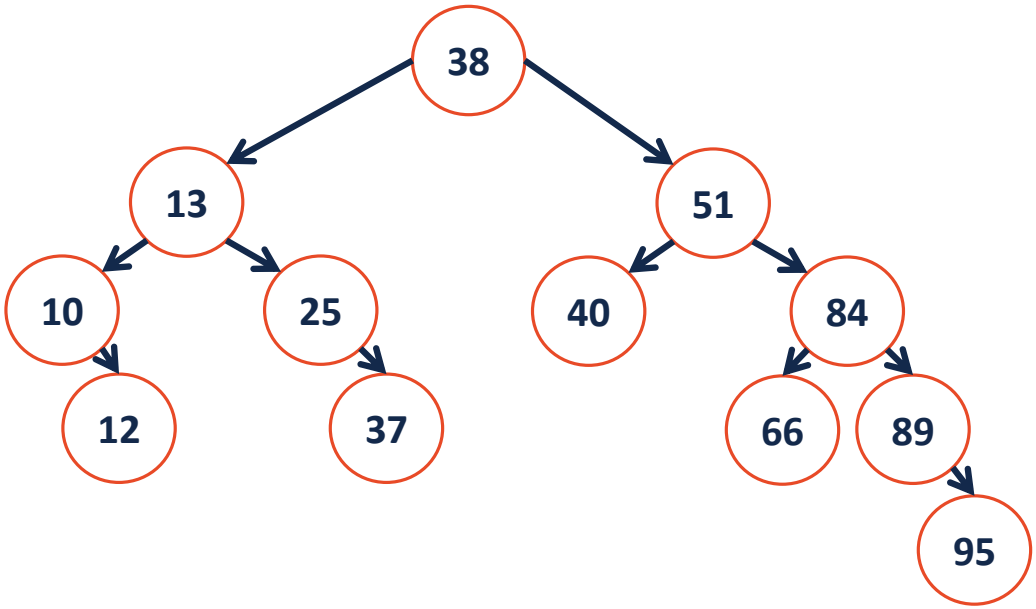





```
remove(10);
```




```
remove(13);
```


38
13
51
10
25
40
84
12
37
66
89
95

```
1  template<class K, class V>

2  void _remove(TreeNode *& root, const K & key) {
3    // find the node
4
5    // Three cases
6    // No child case remove like at the end of a list
7
8
9    // one child case remove like in a list
10
11   // two child remove swap with IOS/IOP
12   // call remove on the swapped node
13
14
15
16
17
18
19
20
21
22
23
24
25
26
```