Data Structures

Binary Search Trees

CS 225

Brad Solomon

July 14, 2022

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Contacting CS 225 Admin Staff

Need an extension? Missed an exam? Have a logistic issue or emergency?

Email: cs225admin@lists.cs.illinois.edu

Exam Logistics

https://courses.engr.illinois.edu/cs225/fa2022/exams/

Future Exams show

Current Exam

Exam

Exam 1

60 points

Register

Exam Availability 09/11 - 09/13

There will be one programing question and a few short answer questions on C++. The programing question will be in an envirome...

Past Exams

Exam

Exam 0

60 points

Register

Exam Availability 08/30 - 09/01

Exam 0 is designed to be a low-stress introduction to the CBTF exam environment. This quiz is on foundational knowledge you ...

Learning Objectives

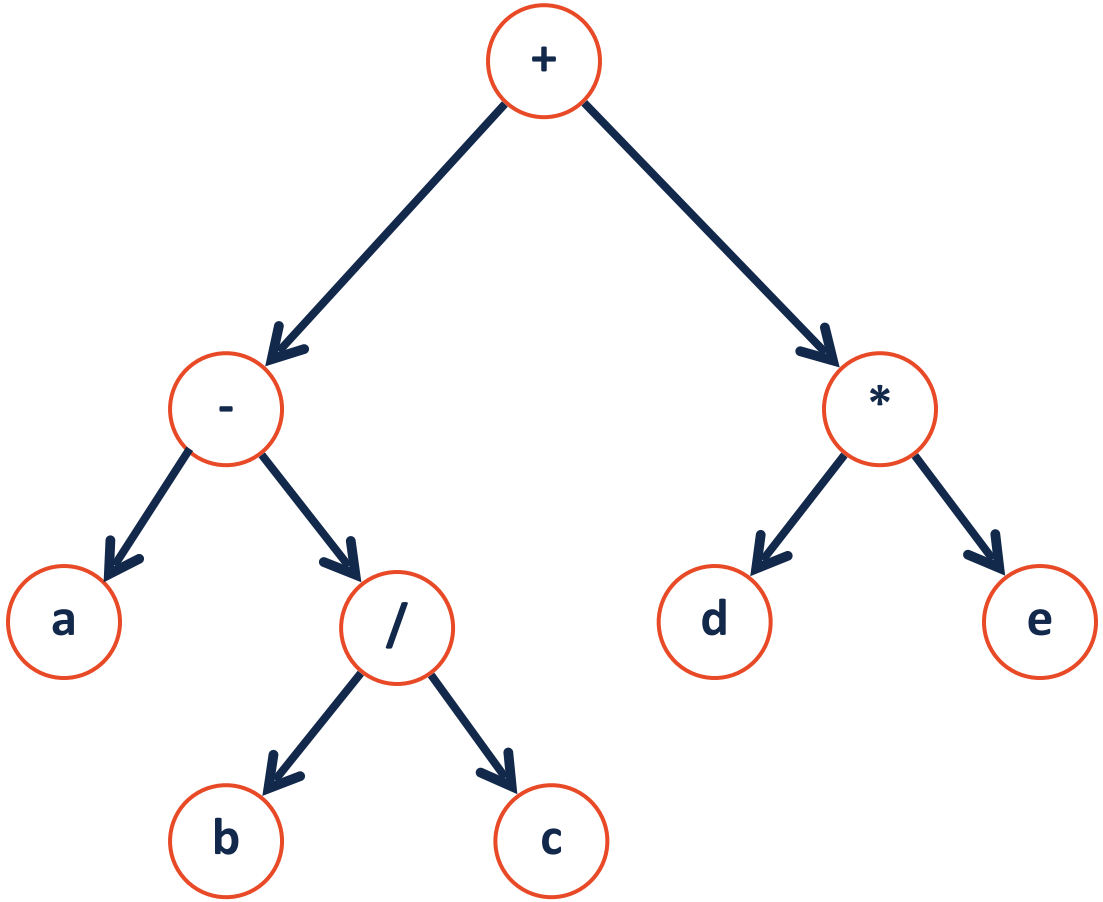
Conceptualize and code tree traversals

Review dictionaries and binary trees

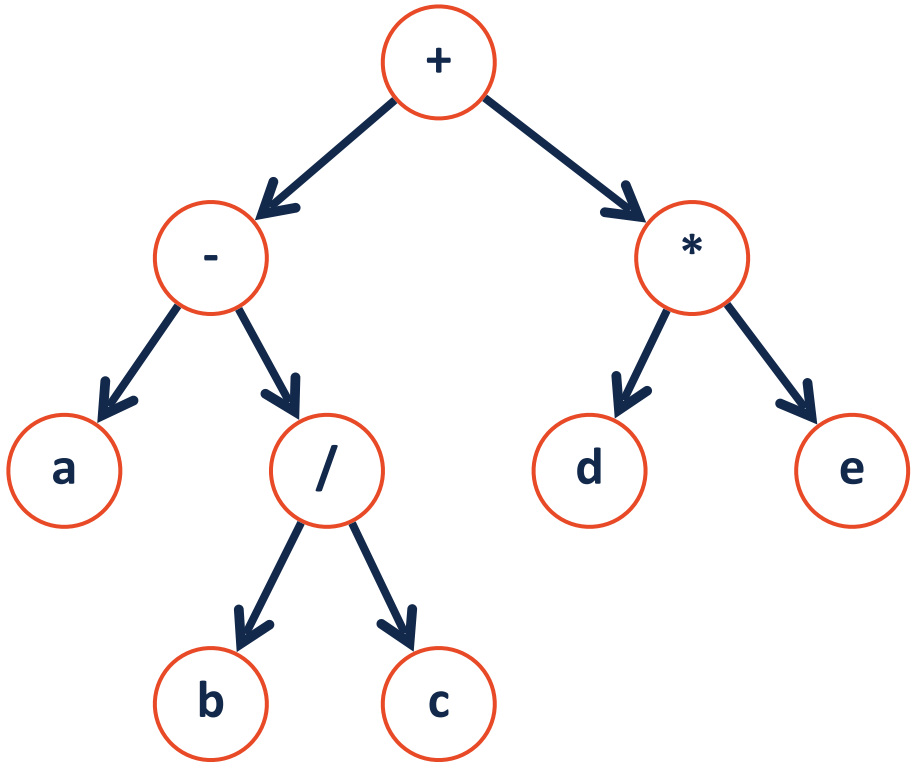
Introduce the binary search tree

Conceptualize and code BST functions

Traversal

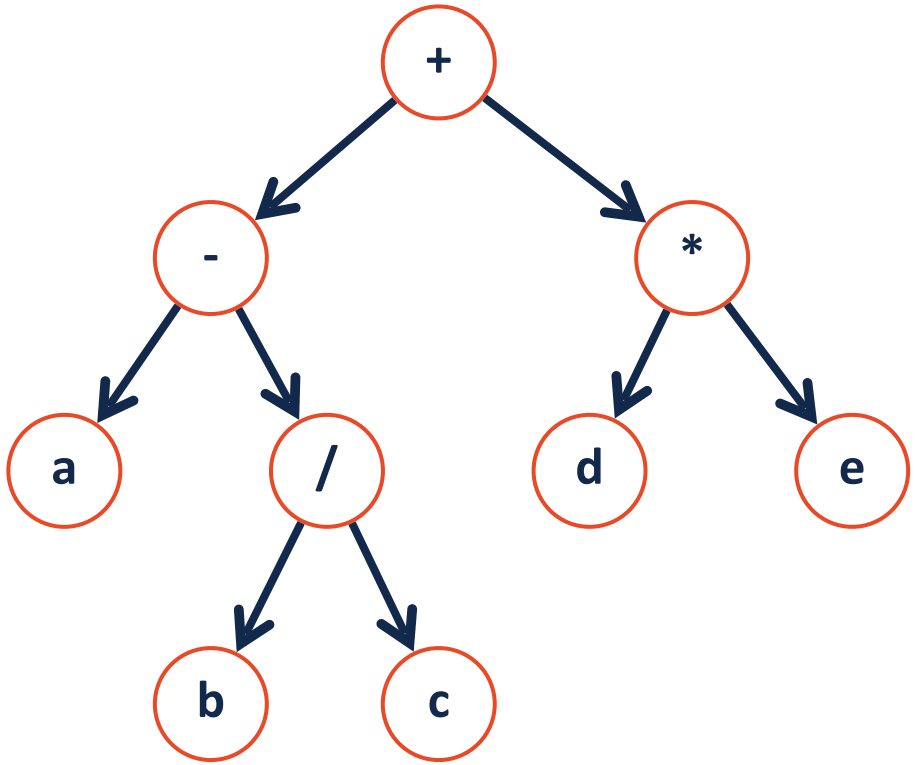


Traversals



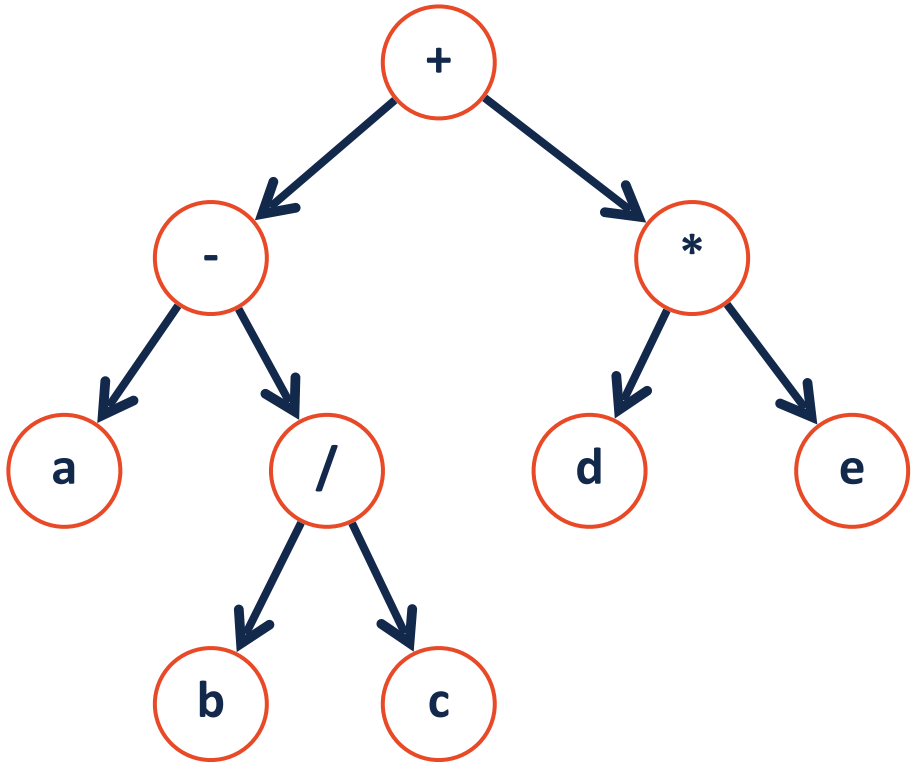

```
template<class T>
void BinaryTree<T>::__Order(TreeNode * root)
{



}
```

Traversals



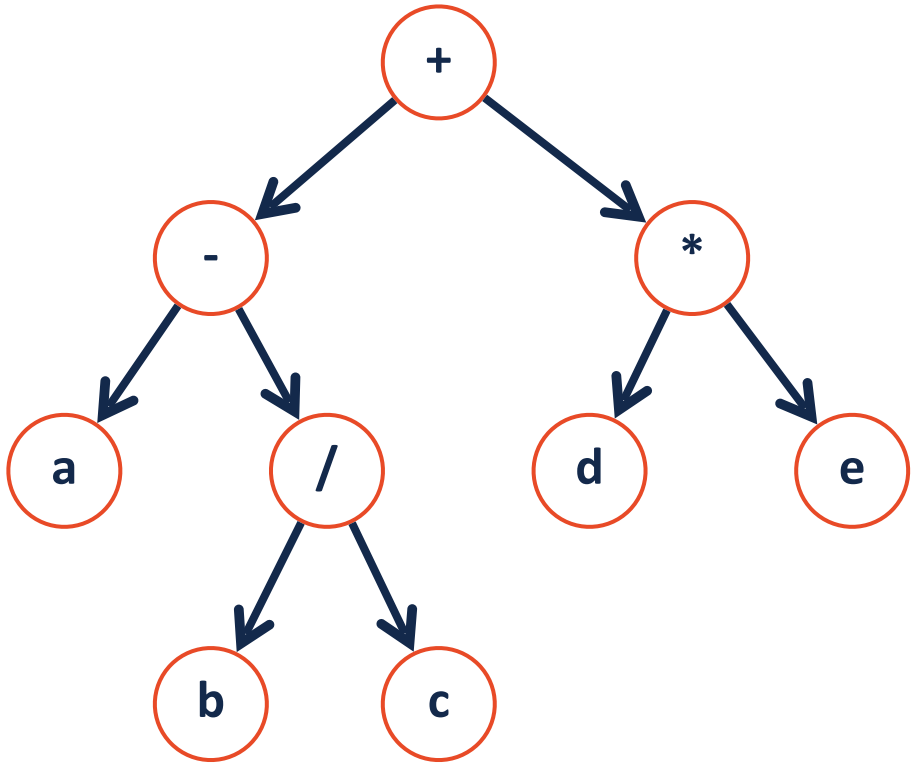

```
1  template<class T>
2  void BinaryTree<T>::__Order(TreeNode * root)
3  {
4     if (root != NULL) {
5
6        ____________________________;
7
8        ___Order(root->left);
9
10       ____________________________;
11
12       ___Order(root->right);
13
14       ____________________________;
15
16    }
17 }
18
19
```

Traversals

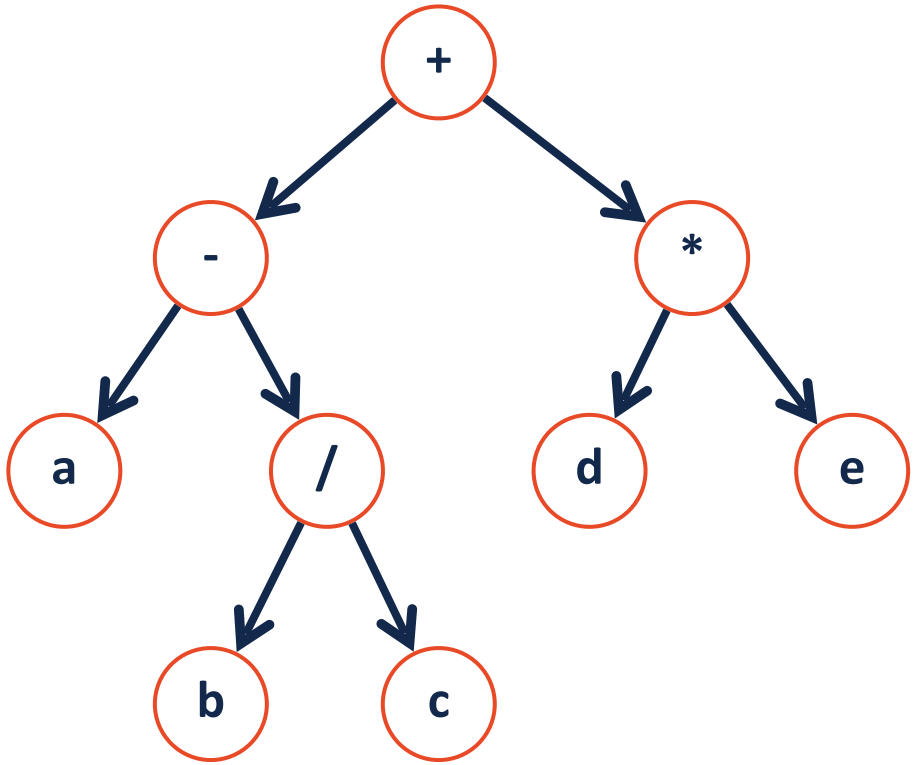



```
1  template<class T>
2  void BinaryTree<T>::__Order(TreeNode * root)
3  {
4     if (root != NULL) {
5
6        ____________________________;
7
8        ___Order(root->left);
9
10       ____________________________;
11
12       ___Order(root->right);
13
14       ____________________________;
15
16    }
17 }
18
19
```

A Different Type of Traversal



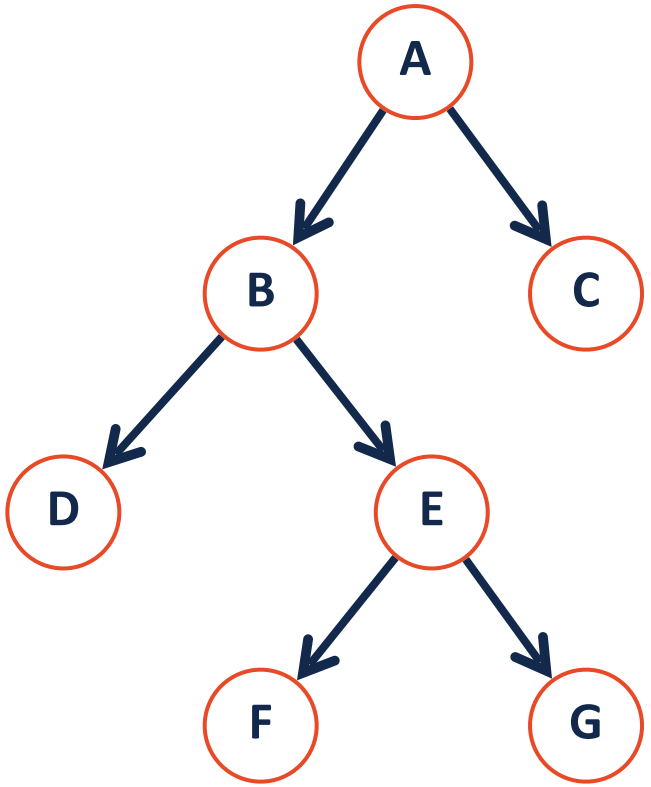
A Different Type of Traversal




```
template<class T>
void BinaryTree<T>::lOrder(TreeNode * root)
{

    Queue<TreeNode*> q;
    q.enqueue(root);

    while( q.empty() == False){

        TreeNode* temp = q.head();
        process(temp);

        q.dequeue();

        q.enqueue(temp->left);
        q.enqueue(temp->right);
    }

}
```

Traversal vs Search

Traversal



Search

Dictionary ADT

Data is often organized into key/value pairs:

Word ➔ Definition

Course Number ➔ Lecture/Lab Schedule

Node ➔ Edges

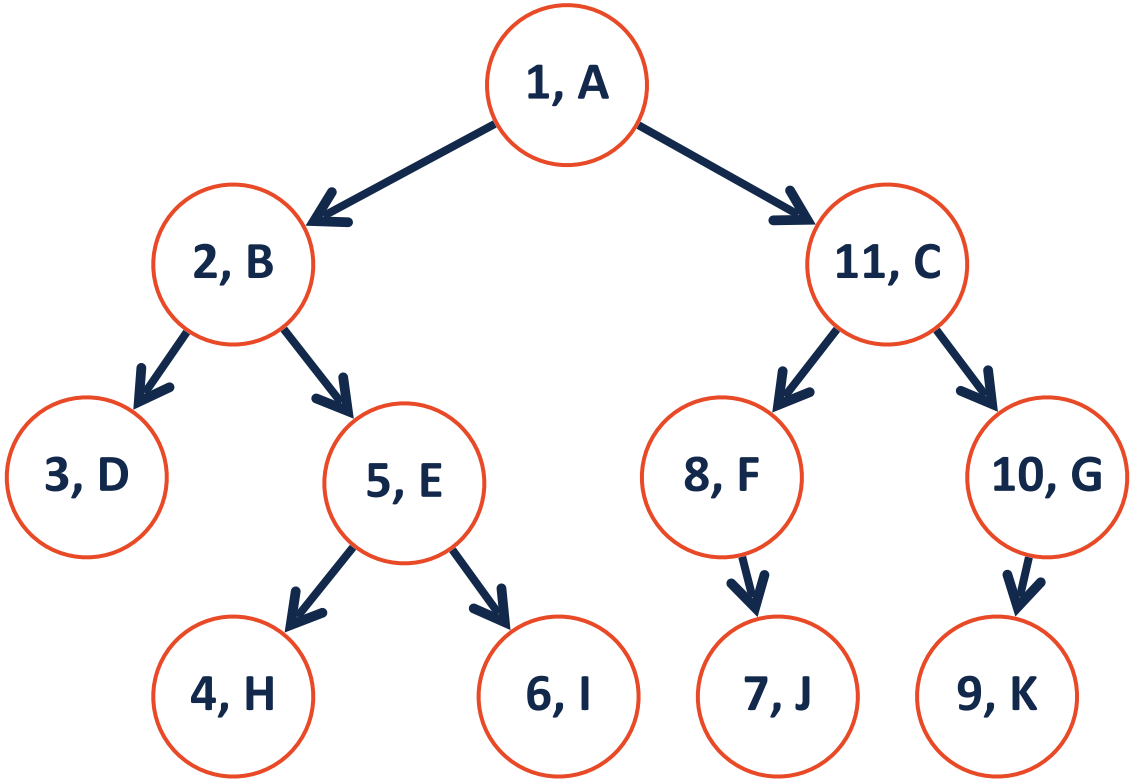
Flight Number ➔ Arrival Information

URL ➔ HTML Page

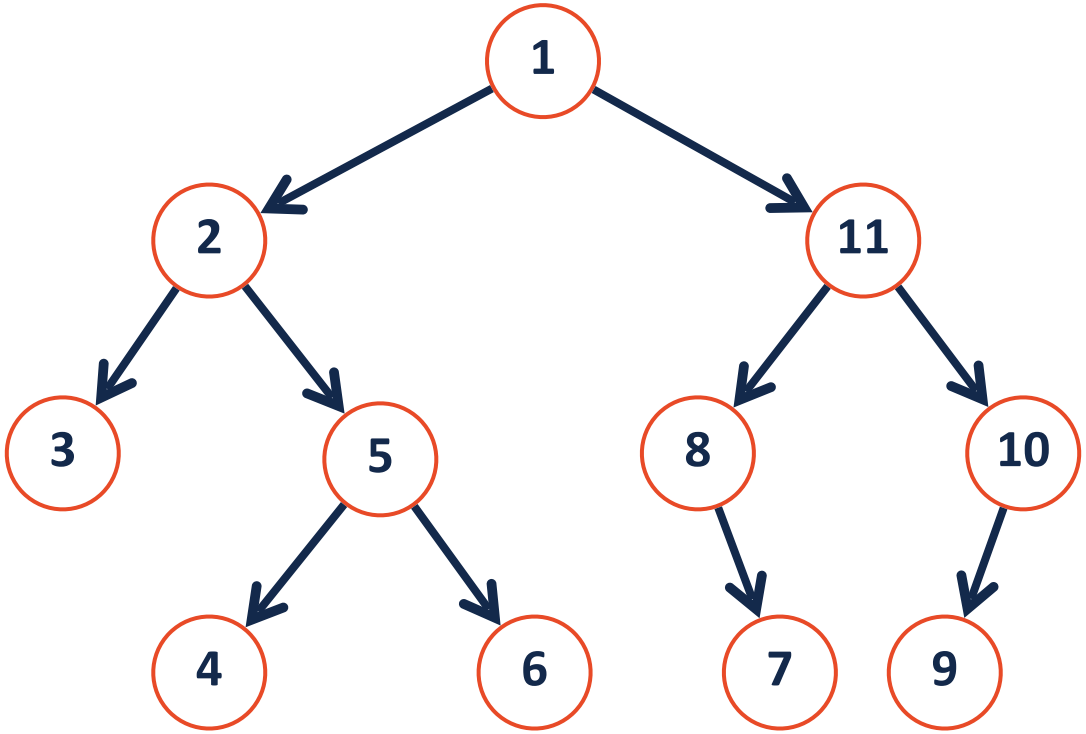
Dictionary.h

```
1  #pragma once
2
3
4  class Dictionary {
5    public:
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20   private:
21     // ...
22 };
23
24
25
26
27
28
```

Binary Trees



Binary Trees



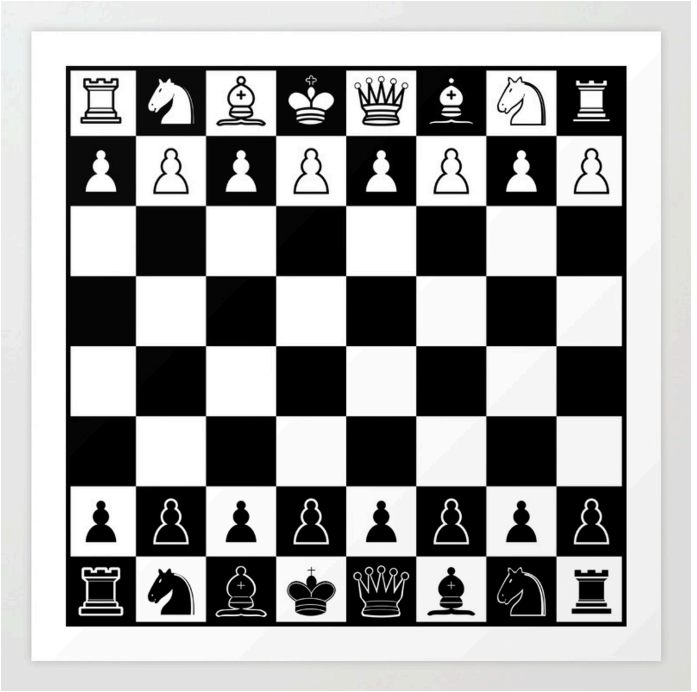
Search: Breadth First vs Depth First

Breadth First Search (BFS)

Depth First Search (DFS)

Choosing a search algorithm

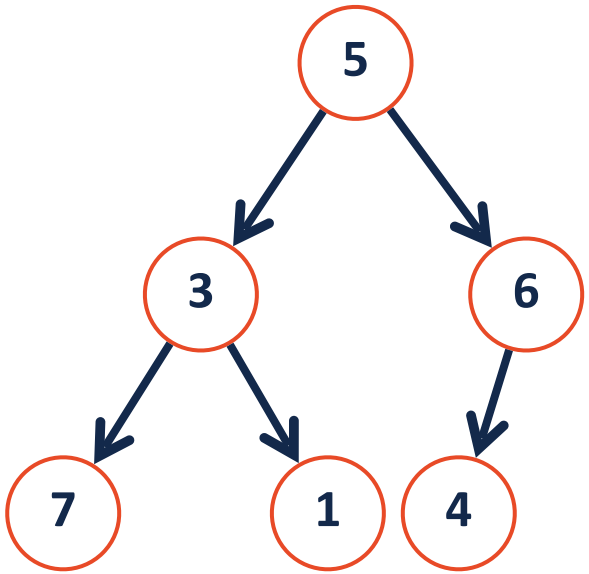
The average ‘branch factor’ for a game of chess is ~31. If you were searching a decision tree for chess, which search algorithm would you use?

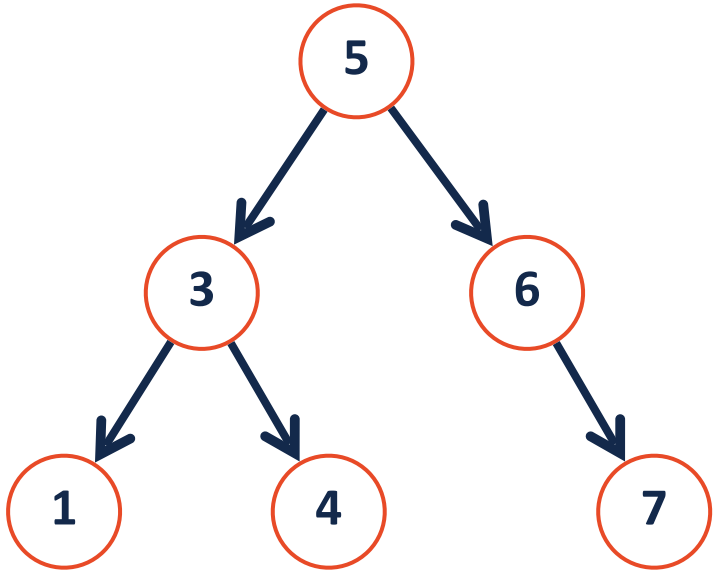


Improved search on a binary tree

7	3	1	5	4	6

1	3	4	5	6	7





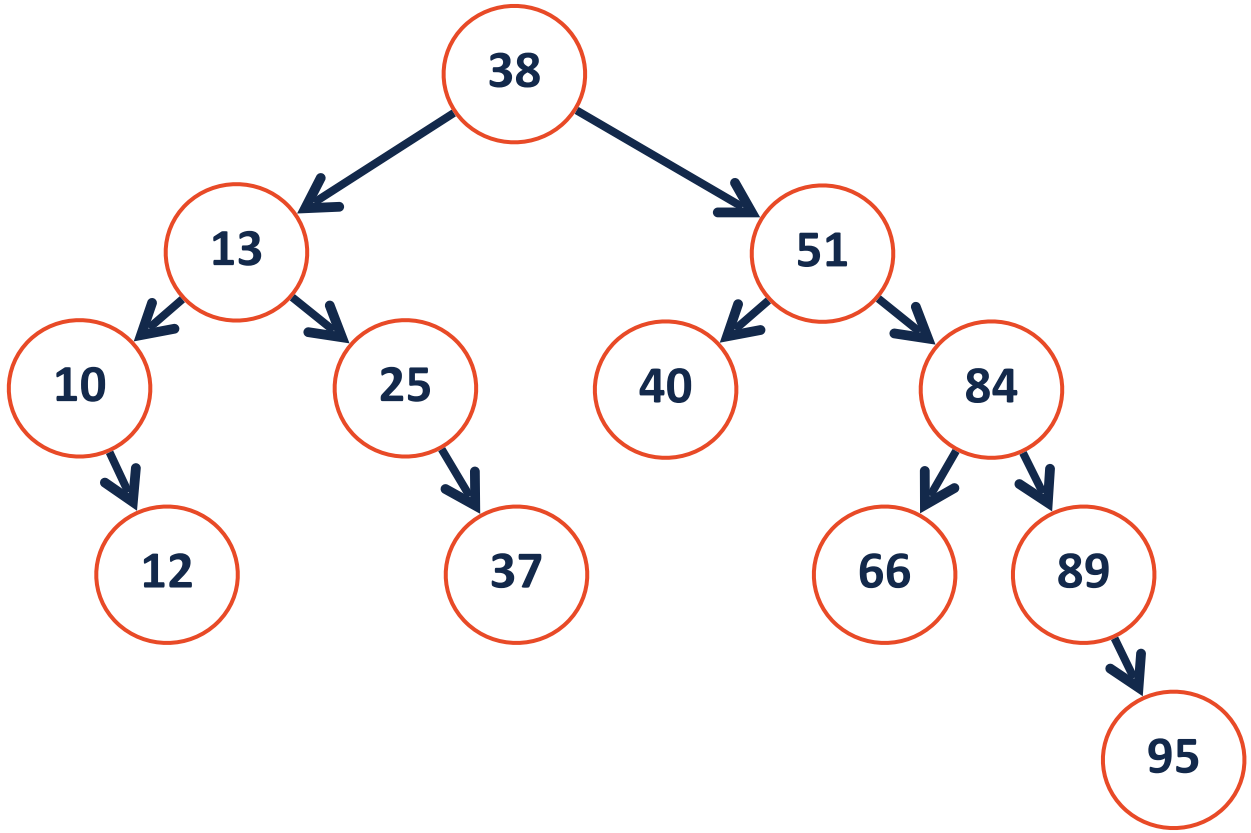
Binary Search Tree (BST)

A **binary search tree** T is either:

-

OR

-

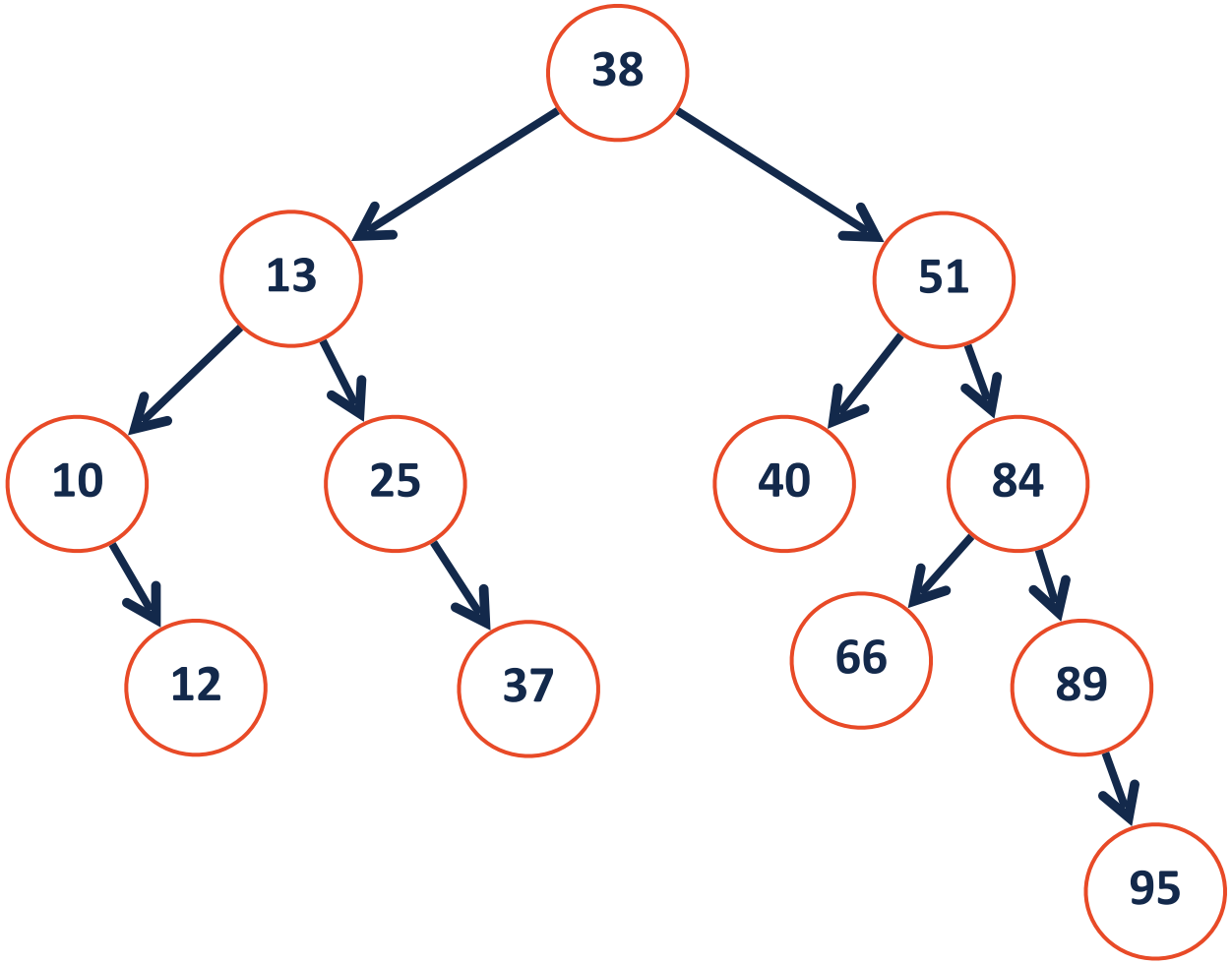


BST.h

```
 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   private:
12     struct TreeNode {
13        TreeNode *left, *right;
14        K  key;
15        V  value;
16        TreeNode();
17     };
18
19     TreeNode *head_;
20 };
21
22
23
```

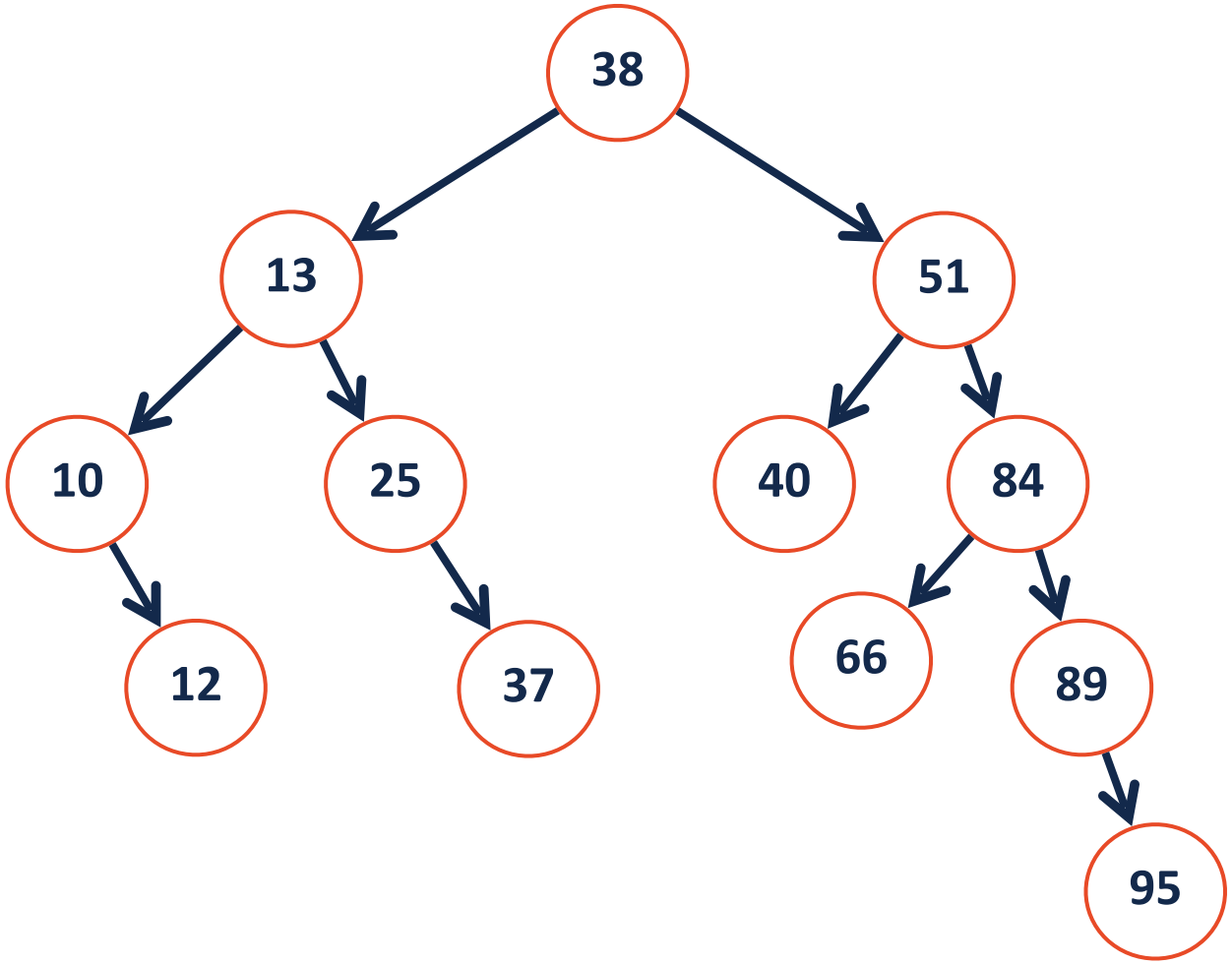
BST Find

`find(66)`



BST Find

`find(9)`



BST.h

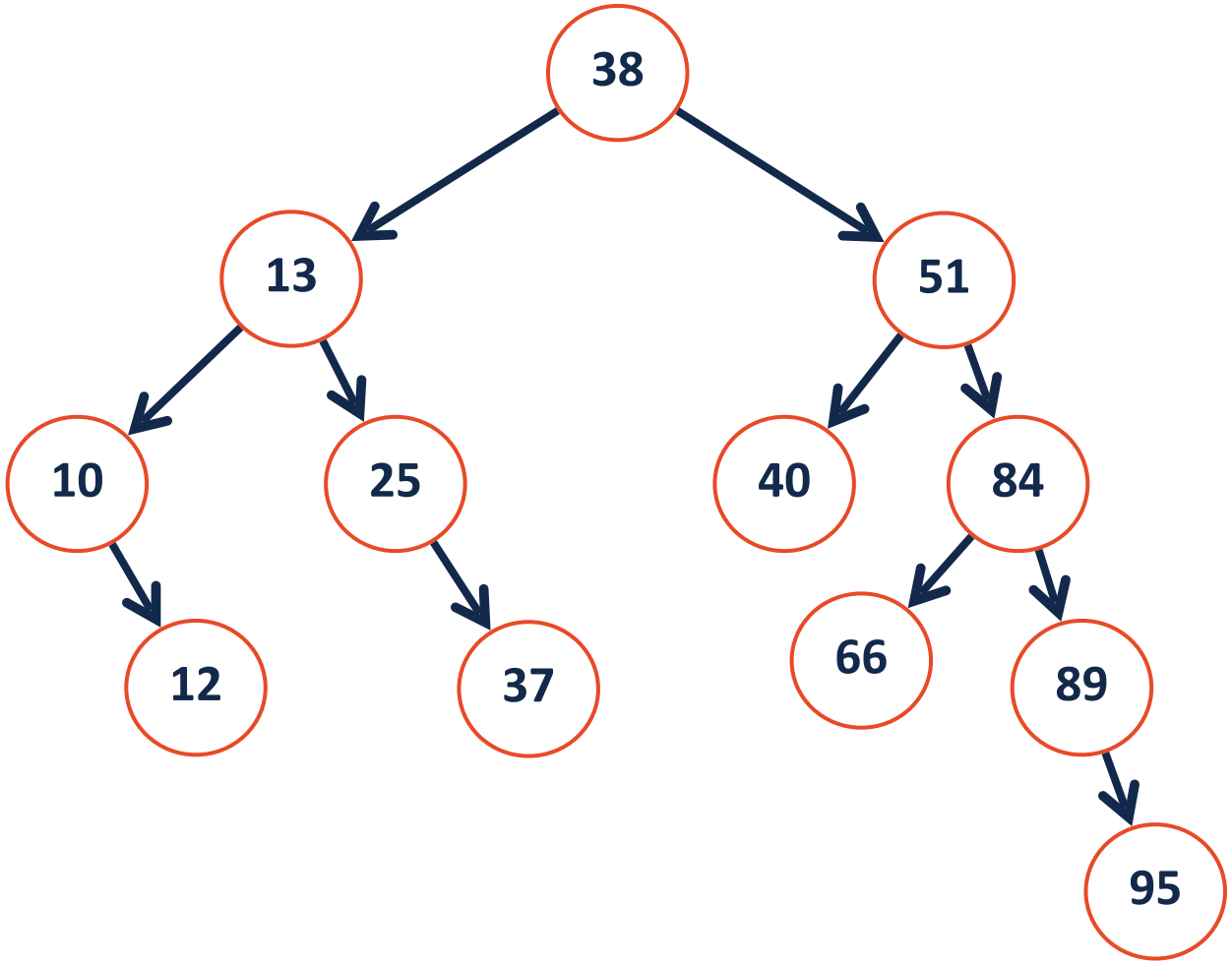
```
1  template<typename K, typename V>
2
3  ____________________________ _find(TreeNode *& root, const K & key) {
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23 }
```

BST.h

```
template<typename K, typename V>

V find(const K & key) const {









}
```

BST Insert

`insert(33)`



BST.h

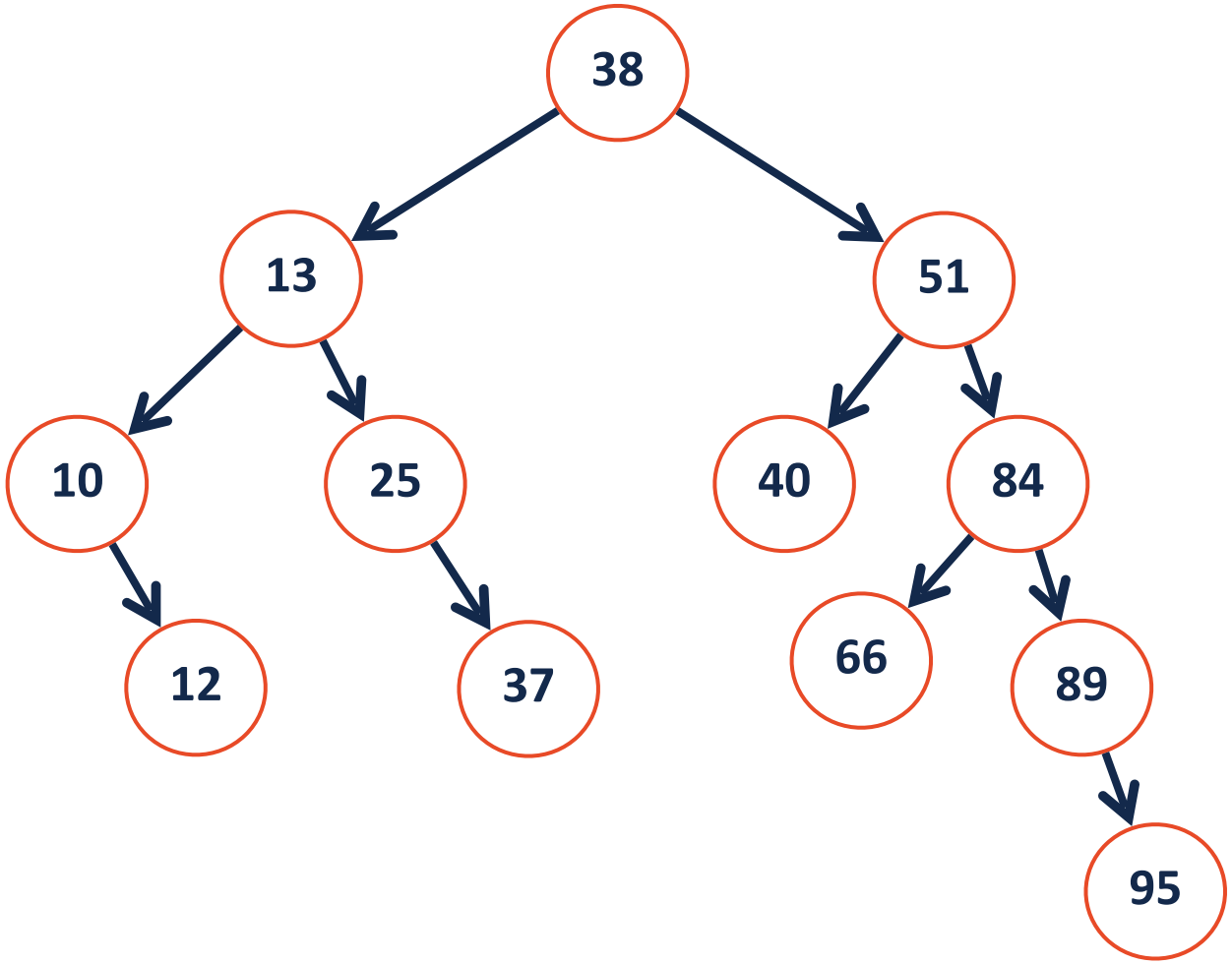
```
1  template<typename K, typename V>
2
3  void insert(const K & key, V val) {
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23 }
```

BST Insert

What binary tree would be formed by inserting the following sequence of integers: `[3, 7, 2, 1, 4, 8, 0]`

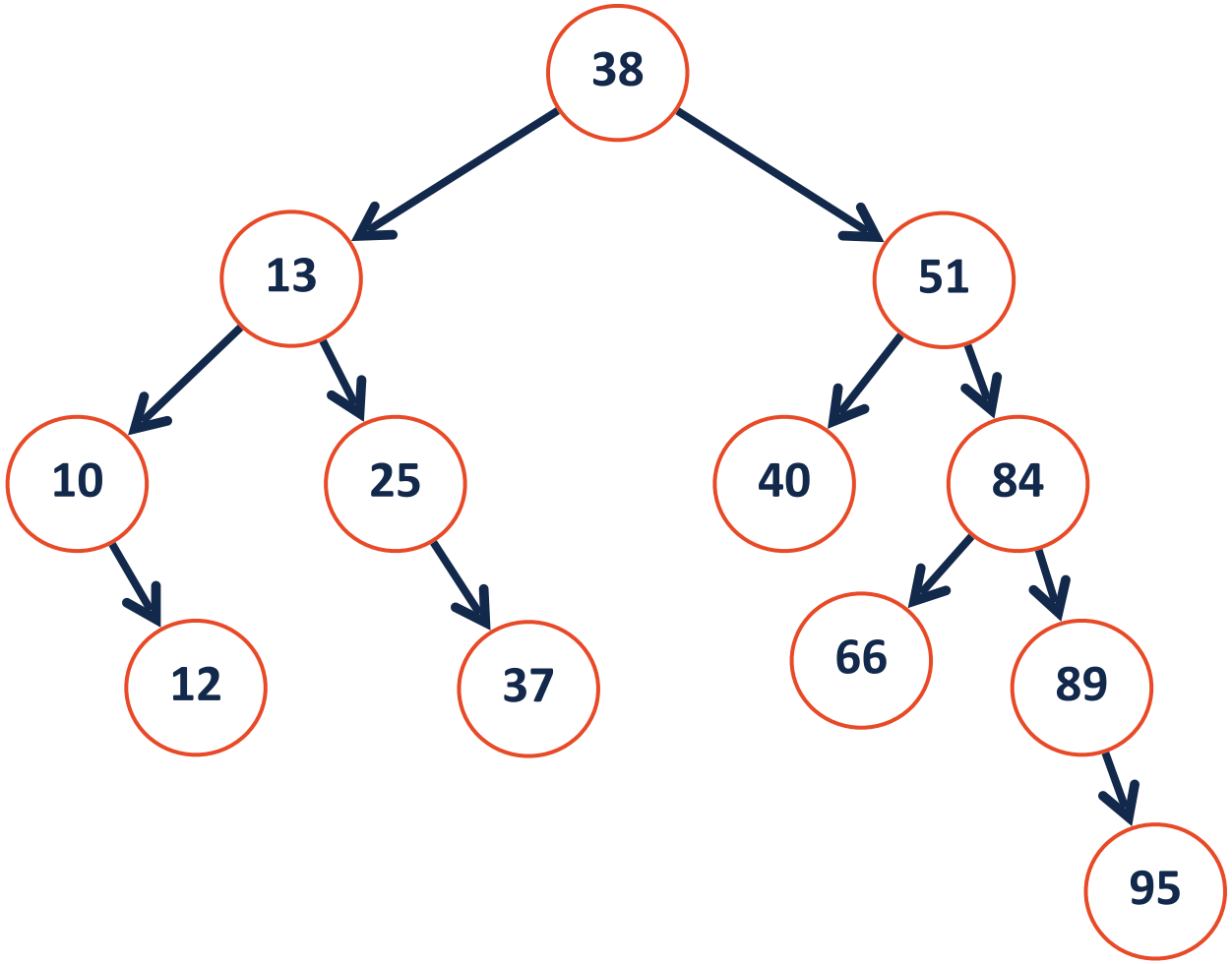
BST Remove

`remove(40)`



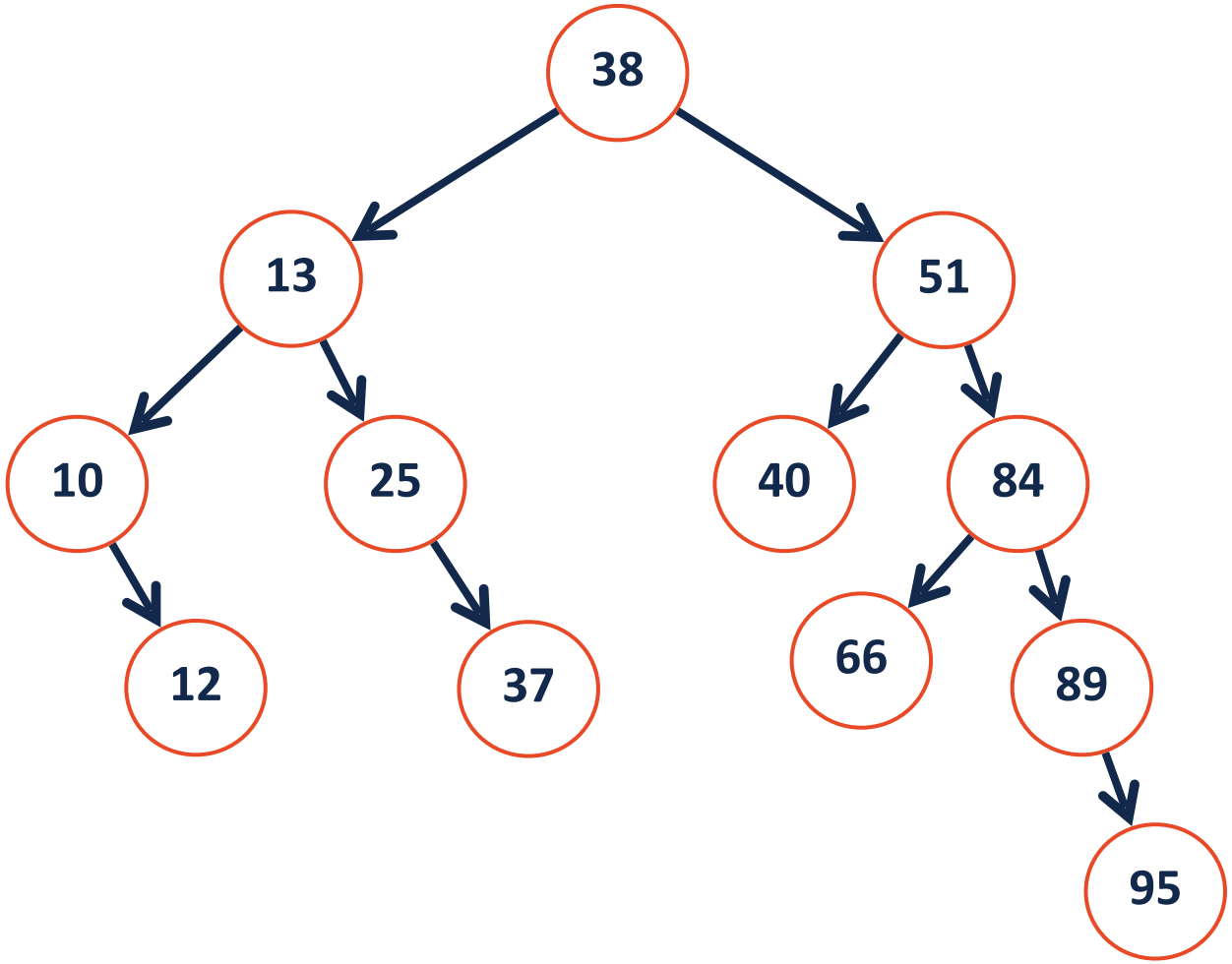
BST Remove

`remove(25)`



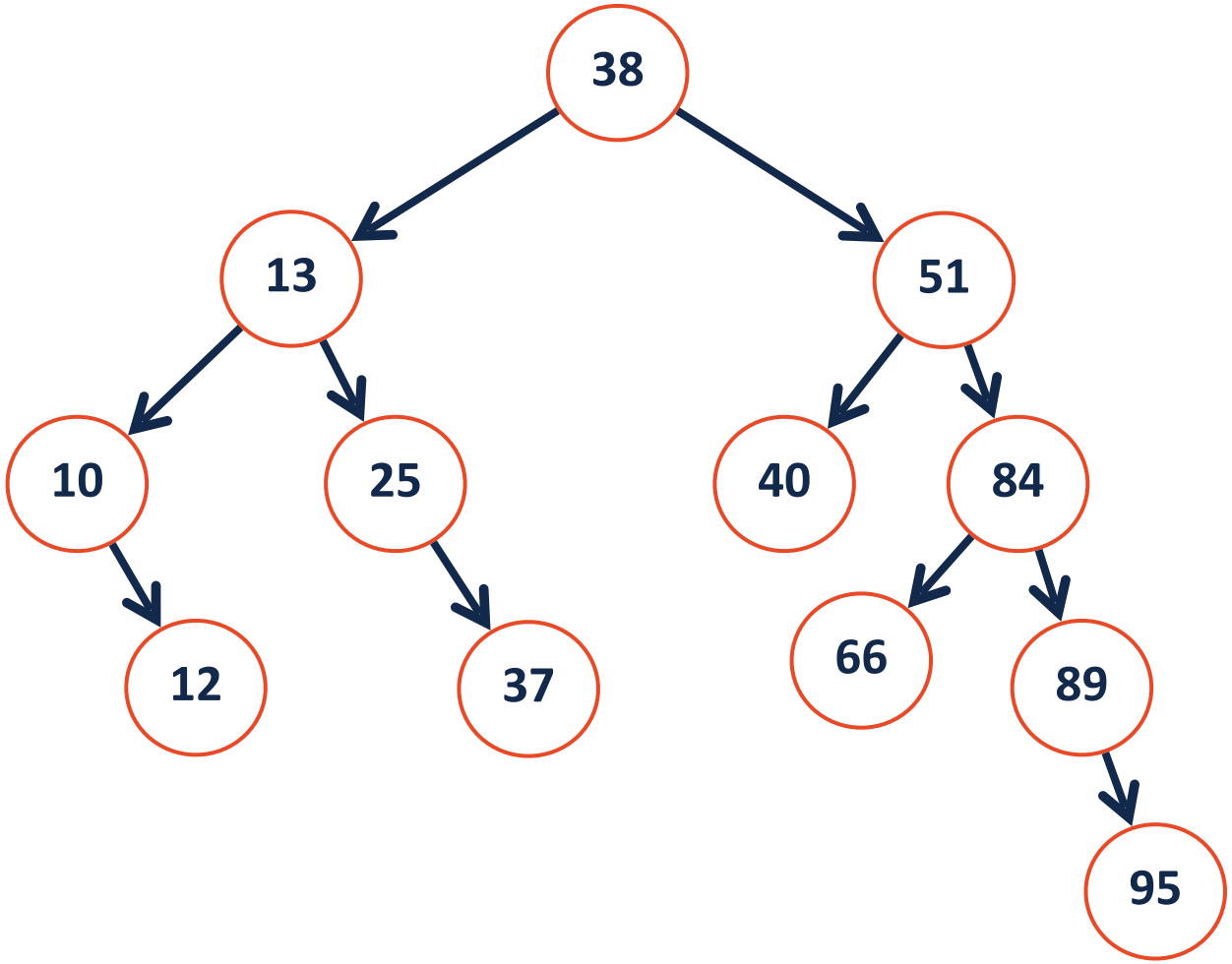
BST Remove

`remove(13)`



BST Remove

`remove(51)`



BST Analysis – Running Time

Operation	BST Worst Case
find	
insert	
delete	
traverse	