Data Structures

Binary Tree Creation

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

So far

- We learned manual creation
- 3 Traversal methods
- Let’s create a tree structure
- Then see how to **add edges**
- We'll create a tree class containing a **root node**

```
class Node:...

class BinaryTree:
    def __init__(self, value):
        self.root = Node(value)

    def _print_inorder(self, current):
        if not current:
            return
        self._print_inorder(current.left)
        print(current.val, end=' ')
        self._print_inorder(current.right)

    def print_inorder(self):
        self._print_inorder(self.root)
```

Inorder Traversal

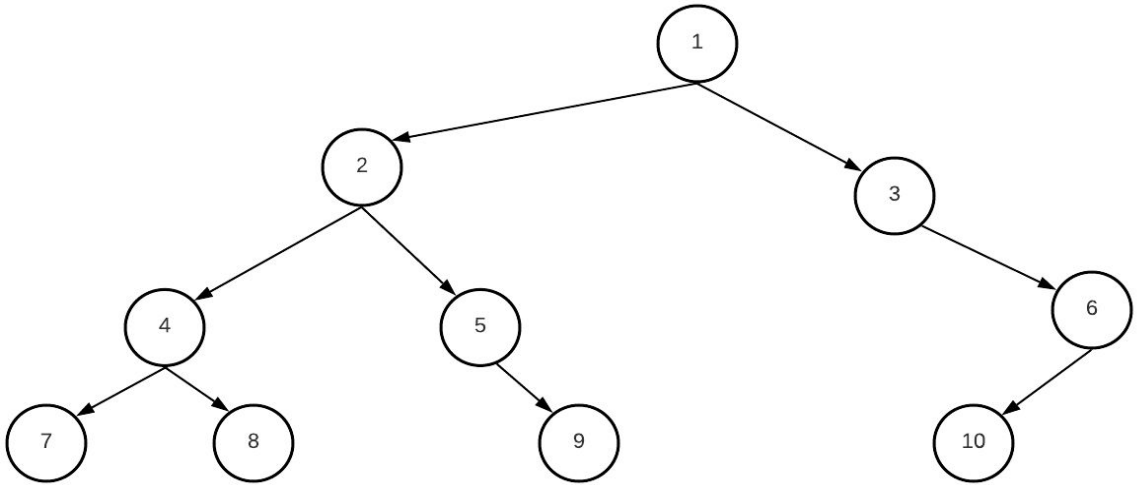
- We can code the inorder traversal in 2 ways as following

```
def _print_inorder(self, current):
    if not current:
        return
    self._print_inorder(current.left)
    print(current.val, end=' ')
    self._print_inorder(current.right)

def print_inorder(self):
    self._print_inorder(self.root)
```

```
def print_inorder(self):
    def inorder(current):
        if not current:
            return
        inorder(current.left)
        print(current.val, end=' ')
        inorder(current.right)

    inorder(self.root)
```

How to construct such a tree?

- Here is 1 way: for each leaf node:
- **Add path nodes**
  - 1 ⇒ 2 ⇒ 4 ⇒ 7
  - 1 ⇒ 2 ⇒ 4 ⇒ 8
  - 1 ⇒ 2 ⇒ 5 ⇒ 9
  - 1 ⇒ 3 ⇒ 6 ⇒ 10
- **Add path directions**
  - LLL
  - LLR
  - LRR
  - RRL



Construction

- It's also good to verify that *paths don't conflict*!
  - Any path passing 'through' a node that has already been created must check that the node has the same value as previously

```
    def add(self, values_lst, direction_lst):...


if __name__ == '__main__':
    tree = BinaryTree(1)
    tree.add([2, 4, 7], ['L', 'L', 'L'])
    tree.add([2, 4, 8], ['L', 'L', 'R'])
    tree.add([2, 5, 9], ['L', 'R', 'R'])
    tree.add([3, 6, 10], ['R', 'R', 'L'])

    tree.print_inorder()
    # 7 4 8 2 5 9 1 3 10 6
```

Construction

- We already set the root value
- Each path is for the **remaining** children

```
def add(self, values_lst, direction_lst):
    assert len(values_lst) == len(direction_lst)

    current = self.root
    # iterate on the path, all necessary nodes
    for i, val in enumerate(values_lst):
        if direction_lst[i] == 'L':
            if not current.left:
                current.left = Node(values_lst[i])
            else:
                assert current.left.val == values_lst[i]
            current = current.left
        else:
            if not current.right:
                current.right = Node(values_lst[i])
            else:
                assert current.right.val == values_lst[i]
            current = current.right
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."