Python Chapter 9: Trees and Graphs

Ezequiel Torres

June 27, 2024

Table of contents

Basics of Heaps

Nodes in python are very simple. We simply use the node class and reference the node class in itself

Binary Tree Example

```
class Treenode:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

class Tree:
    def __init__(self):
        self.root = None
```

Print Binary Tree

```
def height(root):
    if root is None:
        return 0
    return max(height(root.left), \
    height(root.right))+1


def getcol(h):
    if h == 1:
        return 1
    return getcol(h-1) + getcol(h-1) + 1
```

Print Binary Tree Cont.

```
def printTree(M, root, col, row, height):
    if root is None:
        return
    M[row][col] = root.data
    printTree(M, root.left, col-pow(2, height-2), \
    row+1, height-1)
    printTree(M, root.right, col+pow(2, height-2),\
    row+1, height-1)
```

Print Binary Tree Cont.

```
def TreePrinter():
    h = height(myTree.root)
    col = getcol(h)
    M = [[0 for _ in range(col)] \
    for __ in range(h)]
    printTree(M, myTree.root, col//2, 0, h)
    for i in M:
        for j in i:
            if j == 0:
                print(" ", end=" ")
            else:
                print(j, end=" ")
        print("")
```

Print Binary Tree Cont.

```
myTree = Tree()
myTree.root = Treenode(1)
myTree.root.left = Treenode(2)
myTree.root.right = Treenode(3)
myTree.root.left.left = Treenode(4)
myTree.root.left.right = Treenode(5)
myTree.root.right.left = Treenode(6)
myTree.root.right.right = Treenode(7)
TreePrinter()
```

Preorder

```
class Node:
    def __init__(self, v):
        self.data = v
        self.left = None
        self.right = None
```

Preorder Cont.

```
def printPreorder(node):
    if node is None:
        return

    # Deal with the node
    print(node.data, end=' ')

    # Recur on left subtree
    printPreorder(node.left)

    # Recur on right subtree
    printPreorder(node.right)
```

Preorder Cont.

```
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)
root.right.right = Node(6)

# Function call
print(”Preorder traversal of binary tree is:”)
printPreorder(root)
```

Inorder

```
# Function to print inorder traversal
def printInorder(node):
    if node is None:
        return

    # First recur on left subtree
    printInorder(node.left)

    # Now deal with the node
    print(node.data, end=' ')

    # Then recur on right subtree
    printInorder(node.right)
```

Inorder Cont.

```
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)
root.right.right = Node(6)

# Function call
print("Inorder traversal of binary tree is:")
printInorder(root)
```

PostOrder

```
def printPostorder(node):
    if node == None:
        return

    # First recur on left subtree
    printPostorder(node.left)

    # Then recur on right subtree
    printPostorder(node.right)

    # Now deal with the node
    print(node.data, end=' ')
```

PostOrder Cont.

```
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)
root.right.right = Node(6)

# Function call
print("Postorder traversal of binary tree is:")
printPostorder(root)
```