CSE 220 LAB № 8

Name:Shihab Shahriear Antor | ID-20301113 | sec-07 01/01/2021

Solution (Problem 1 to 7)

Listing 1: Binary Tree

```
1  class TreeNode:
2      def __init__(self, elem):
3          self.elem = elem
4          self.r = None
5          self.l = None
6
7      def heightOfTree(self, n):
8          if n is not None:
9              left_dep = self.heightOfTree(n.l)
10             right_dep = self.heightOfTree(n.r)
11
12             if left_dep > right_dep:
13                 return left_dep + 1
14             else:
15                 return right_dep + 1
16         else:
17             return 0
18
19     def getlevel(self, n, val, level):
20         if n == None:
21             return 0
22         if n.elem == val:
23             return level
24         d_level = self.getlevel(n.l, val, level+1)
25
26         if d_level != 0:
27             return d_level
28         d_level = self.getlevel(n.r, val, level+1)
29
30         return d_level
31
32     def levelOfNode(self, parent, val):
33         return self.getlevel(parent, val, 1)
34
35
```

```
36      def preOrder(self, parent, root):
37          if parent:
38              l = self.levelOfNode(root, parent.elem) - 1
39              space = " " * l*4
40              prefix = space + "|__" if l != 0 else u"\u6839"
41
42              print(prefix, parent.elem)
43              print()
44              self.preOrder(parent.l, root)
45              self.preOrder(parent.r, root)
46
47
48      def postOrder(self, parent, root):
49          if parent:
50              l = self.levelOfNode(root, parent.elem) - 1
51              space = " " * l*4
52              prefix = space + u"|\u0305\u0305 " if l != 0 else u"\u6839"
53
54              self.postOrder(parent.l, root)
55              self.postOrder(parent.r, root)
56              print(prefix, parent.elem)
57              print()
58
59
60      def inOrder(self, parent, root):
61          if parent:
62              l = self.levelOfNode(root, parent.elem)-1
63              space = " " * l*4
64              prefix = space + "|--> " if l != 0 else u"\u6839"
65
66              self.inOrder(parent.l, root)
67              print(prefix, parent.elem)
68              print()
69              self.inOrder(parent.r, root)
70
71      def twinTree(self, x, y):
72          if x is None and y is None:
73              return True
74          if x is not None and y is  not None:
75              return ((x.elem == y.elem) and self.twinTree(x.l, y.l) and ↩
                    self.twinTree(x.r, y.r))
76          return False
77
78      def xeroxTree(self, n):
79          if n is None:
80              return None
81
```

```
        parent_copy = TreeNode(n.elem)
        parent_copy.l = self.xeroxTree(n.l)
        parent_copy.r = self.xeroxTree(n.r)

        return parent_copy

    def tester(self):

        #Tree 1
        root1 = TreeNode("Anime")

        root1.l = TreeNode("Shounen")
        root1.l.l = TreeNode("DRAGON BALL")
        root1.l.r = TreeNode("Naruto")

        root1.r = TreeNode("Thriller")
        root1.r.l = TreeNode("Death Note")
        root1.r.r = TreeNode("PSYCHO-PASS")

        #Tree 2
        root2 = TreeNode("Anime")

        root2.l = TreeNode("Shounen")
        root2.l.l = TreeNode("DRAGON BALL")
        root2.l.r = TreeNode("Naruto")

        root2.r = TreeNode("Thriller")

        print("\n#================# 1(Height) #================#\n")
        height = root1.heightOfTree(root1)
        print("Height of the Tree: ", height)

        print("\n#================# 2(level) #================#\n")
        n = "Shounen"
        level = root1.levelOfNode(root1, n)
        print(f"Level of '{n}': ", level)

        print("\n#================# 3(Pre Order) #================#\n")

        root1.preOrder(root1, root1)

        print("\n#================# 4(Post Order) #================#\n")
        root1.postOrder(root1, root1)

        print("\n#================# 5(In Order) #================#\n")
        root1.inOrder(root1, root1)

```

```
129         print("\n#================# 6(Same tree or not) ↩
                #================#\n")
130         print("#case01\n")
131 
132         #Case 01
133         xerox1 = root1.twinTree(root1, root2)
134         if xerox1:
135             print(u"    \u2713 Two trees are exactly same")
136         else:
137             print(u"    \u2717 Two trees are not same")
138 
139         print("\n#case02\n")
140         #Tree 2 Extention
141         root2.r.l = TreeNode("Death Note")
142         root2.r.r = TreeNode("PSYCHO-PASS")
143 
144         #Case 02
145         xerox2 = root1.twinTree(root1, root2)
146         if xerox2:
147             print(u"    \u2713 Two trees are exactly same")
148         else:
149             print(u"    \u2717 Two trees are not same")
150 
151         print("\n#================# 7(copy tree) #================#\n")
152 
153         cy = root1.xeroxTree(root1)
154         print("\n#===== [Original Tree] =====#\n")
155         root1.preOrder(root1, root1)
156         print("\n#===== [Clone Tree] =====#\n")
157         root1.preOrder(cy, cy)
158         print("\n#===== [ End ] =====#\n")
159 
160 #================# Problem (1 -> 7) #================#
161 ob = TreeNode(None)
162 ob.tester()
163 
164 #================# #================#
```

Outputs (Problem 1 to 7)

Listing 2: Outputs (1 -> 7)

```
#================# 1(Height) #================#

Height of the Tree:  3

#================# 2(level) #================#

Level of 'Shounen':  2

#================# 3(Pre Order) #================#

Anime
    |__ Shounen
        |__ DRAGON BALL
        |__ Naruto
    |__ Thriller
        |__ Death Note
        |__ PSYCHO-PASS


#================# 4(Post Order) #================#

        |--  DRAGON BALL
        |--  Naruto
    |--  Shounen
        |--  Death Note
        |--  PSYCHO-PASS
    |--  Thriller
Anime


#================# 5(In Order) #================#

        |-->  DRAGON BALL
    |-->  Shounen
        |-->  Naruto
Anime
        |-->  Death Note
    |-->  Thriller
        |-->  PSYCHO-PASS


```

```
43
44 #================# 6(Same tree or not) #================#
45
46 #case01
47
48         Two trees are not same
49
50 #case02
51
52         Two trees are exactly same
53
54 #================# 7(copy tree) #================#
55
56
57 #===== [Original Tree] =====#
58
59 Anime
60     |__ Shounen
61         |__ DRAGON BALL
62         |__ Naruto
63     |__ Thriller
64         |__ Death Note
65         |__ PSYCHO-PASS
66
67 #===== [Clone Tree] =====#
68
69 Anime
70     |__ Shounen
71         |__ DRAGON BALL
72         |__ Naruto
73     |__ Thriller
74         |__ Death Note
75         |__ PSYCHO-PASS
76
77 #===== [ End ] =====#
```

Solution (Problem 8)