

Course Name: Advanced Programming Lab

Experiment 2.2

Student Name: UTKARSH JOSHI

Branch: BE CSE

Semester: 5th

Subject Name: Advanced Programming Lab

UID: 21BCS9158

Section/Group: 802-A

Date of Performance: 21.09.2023

Subject Code: 21CSP314

AIM:

Solve the following problems on hackerrank:

- 1.Tree:Top view
- 2.Balanced forest

Q1) Tree:Top view:

Given a pointer to the root of a binary tree, print the top view of the binary tree.

The tree as seen from the top the nodes, is called the top view of the tree.

Top View :

Complete the function and print the *topview* resulting values on a single line separated by space.

CODE:



Course Name: Advanced Programming Lab

```
31         else:
32             current.right = Node(val)
33             break
34         else:
35             break
36     """
37     Node is defined as
38     self.left (the left child of the node)
39     self.right (the right child of the node)
40     self.info (the value of the node)
41     """
42     def topView(root):
43         hm={}
44         queue=[]
45         queue.append((root,0))
46         while(queue):
47             q=queue.pop(0)
48             if q[1] not in hm:
49                 hm[q[1]]=q[0].info
50             if q[0].left:
51                 queue.append((q[0].left,q[1]-1))
52             if q[0].right:
53                 queue.append((q[0].right,q[1]+1))
54         for k, v in sorted(hm.items()):
55             print(str(v)+' ', end='')
56
```

Line: 55 Col: 34

OUTPUT:

Congratulations

You solved this challenge. Would you like to challenge your friends?



Next Challenge

Test case 0

Compiler Message

Success

Test case 1

Test case 2

Input (stdin)

Download

```
1 6
2 1 2 5 3 6 4
```

Test case 3

Test case 4

Expected Output

Download

```
1 1 2 5 6
```

Test case 5

Course Name: Advanced Programming Lab

QUES 2) BFS: shortest reach: Function

Description:

Complete the *balancedForest* function in the editor below. It must return an integer representing the minimum value of that can be added to allow creation of a balanced forest, or if it is not possible.

balancedForest has the following parameter(s):

- *c*: an array of integers, the data values for each node
- *edges*: an array of 2 element arrays, the node pairs per edge

CODE:

```
1  #!/bin/python3
2
3  from operator import attrgetter
4  from itertools import groupby
5  from sys import stderr
6
7  class Node:
8      def __init__(self, index, value):
9          self.index = index
10         self.value = value
11         self.children = []
12
13     def readtree():
14         size = int(input())
15         values = readints()
16         assert size == len(values)
17         nodes = [Node(i, v) for i, v in enumerate(values)]
18         for _ in range(size - 1):
19             x, y = readints()
20             nodes[x-1].children.append(nodes[y-1])
21             nodes[y-1].children.append(nodes[x-1])
22         return nodes
23
24     def readints():
```

Course Name: Advanced Programming Lab

```

25     return [int(fld) for fld in input().strip().split()]
26
27 def findbestbal(nodes):
28     if len(nodes) == 1:
29         return -1
30     rootify(nodes[0])
31     # print([(n.index, n.value, n.totalval) for n in nodes], file=stderr)
32     best = total = nodes[0].totalval
33     dumminode = Node(None, None)
34     dumminode.totalval = 0
35     sortnode = []
36     for k, g in groupby(sorted([dumminode] + nodes, key = attrgetter('totalval'))
, attrgetter('totalval')):
37         sortnode.append(list(g))
38         total = nodes[0].totalval
39         for ihi, n in enumerate(sortnode):
40             if 3 * n[0].totalval >= total:
41                 break
42         else:
43             assert False
44         ilo = ihi - 1
45         for ihi in range(ihi, len(sortnode)):
46             hi = sortnode[ihi][0].totalval
47             lo = sortnode[ilo][0].totalval

```

```

48         while 2 * hi + lo > total:
49             if lo == 0:
50                 return -1
51             if (total - lo) % 2 == 0:
52                 x = (total - lo) // 2
53                 for lonode in sortnode[ilo]:
54                     if uptototalval(lonode, x + lo):
55                         return x - lo
56             ilo -= 1
57             lo = sortnode[ilo][0].totalval
58             if len(sortnode[ihi]) > 1:
59                 return 3 * hi - total
60             hinode = sortnode[ihi][0]
61             if 2 * hi + lo == total:
62                 for lonode in sortnode[ilo]:
63                     if uptototalval(lonode, hi) != hinode:
64                         return hi - lo
65             y = total - 2 * hi
66             if uptototalval(hinode, 2 * hi) or uptototalval(hinode, hi + y):
67                 return hi - y
68
69 def rootify(root):
70     root.parent = root.jumpup = None
71     root.depth = 0

```

Course Name: Advanced Programming Lab

```

72     bfnode = [root]
73     i = 0
74     while i < len(bfnode):
75         node = bfnode[i]
76         depth = node.depth + 1
77         jumpup = uptodepth(node, depth & (depth - 1))
78         for child in node.children:
79             child.parent = node
80             child.children.remove(node)
81             child.depth = depth
82             child.jumpup = jumpup
83             bfnode.append(child)
84         i += 1
85     for node in reversed(bfnode):
86         node.totalval = node.value + sum(child.totalval for child in node.
            children)
87
88     def uptodepth(node, depth):
89         while node.depth > depth:
90             if node.jumpup.depth <= depth:
91                 node = node.jumpup
92             else:
93                 node = node.parent
94         return node
95
96     def uptototalval(node, totalval):
97         try:
98             print('uptototalval(%s,%s)' % (node.index, totalval), file=stderr)
99             while node.totalval < totalval:
100                 if node.parent is None:
101                     return None
102                 if node.jumpup.totalval <= totalval:
103                     node = node.jumpup
104                 else:
105                     node = node.parent
106             print((node.index, node.totalval), file=stderr)
107             if node.totalval == totalval:
108                 return node
109             else:
110                 return None
111         except Exception:
112             return None
113
114     ncases = int(input())
115     for _ in range(ncases):
116         print(findbestbal(readtree()))
117

```




Course Name: Advanced Programming Lab

OUTPUT:

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

[Next Challenge](#)

✓ Test case 0

✓ Test case 1

✓ Test case 2

✓ Test case 3

✓ Test case 4

✓ Test case 5

✓ Test case 6

Compiler Message

Success

Input (stdin) [Download](#)

```
1 2
2 5
3 1 2 2 1 1
4 1 2
5 1 3
6 3 5
7 1 4
8 3
9 1 3 5
```