

Name: S V L N Raju

ID No.: 2100031503

Section: 10

HACKERRANK SUBMISSIONS

✓ Bot saves princess



Participants: 148 Max Score: 10 Difficulty: Easy

Try Again

✓ Bot saves princess - 2



Participants: 114 Max Score: 10 Difficulty: Easy

Try Again

✓ PacMan - DFS



Participants: 78 Max Score: 10 Difficulty: Easy

Try Again

✓ Pacman A*



Participants: 52 Max Score: 10 Difficulty: Medium

Try Again

✓ PacMan - BFS



Participants: 56 Max Score: 10 Difficulty: Medium

Try Again

LEETCODE LEADERBOARD

Rank	User	Score	Time	Country
1	KLU_2100031503	61.40	933:16:31	
1	h2100032143	61.40	951:58:29	
1	h2100032091	61.40	952:08:02	
1	klu_2100031755	61.40	986:36:52	
1	h2100032079	61.40	996:06:49	
1	h2100032102	61.40	1000:10:26	
1	h2100031773	61.40	1015:07:59	
1	h2100031251	61.40	1015:24:35	
1	klu_2100031746	61.40	1018:48:07	
1	h2100031559	61.40	1019:27:47	

LEETCODE SUBMISSIONS

LeetCode

[Explore](#)
[Problems](#)
[Interview](#)
[Contest](#)
[Discuss](#)
[Store](#)

LeetCode is hiring! Apply NOW

[Premium](#)

3
 0

All My Submissions

Time Submitted	Question	Status	Runtime	Language
4 days, 16 hours ago	Water and Jug Problem	Accepted	117 ms	java
4 days, 16 hours ago	Coin Change	Accepted	30 ms	java
4 days, 17 hours ago	Sum Root to Leaf Numbers	Accepted	45 ms	python3
4 days, 17 hours ago	Sum Root to Leaf Numbers	Accepted	41 ms	python3
4 days, 17 hours ago	Sum Root to Leaf Numbers	Accepted	50 ms	python3
4 days, 17 hours ago	Path Sum	Accepted	82 ms	python3
4 days, 17 hours ago	Maximum Depth of Binary Tree	Accepted	42 ms	python

Newer

Older

LEETCODE CODES:

1. Water and Jug problem

Code: public class Solution {

```
public boolean canMeasureWater(int j1, int j2, int j) {
```

```
    if(j1>j2)
```

```
        return canMeasureWater(j2,j1,j);
```

```
    if(j > j1+j2)
```

```
        return false;
```

```
    Set<Integer> failSet = new HashSet<>();
```

```
    int X = 0;
```

```
    int Y = 0;
```

```
    while(true){
```

```

        int res = X * j1 + Y * j2;

        if(failSet.contains(res))

            return false;

        if(res == j){

            return true;

        }else if(res < j){

            Y++;

        }else{

            X--;

        }

        failSet.add(res);

    }

}

```

2.Coin Change

Code: class Solution

```

{

    public int coinChange(int[] coins, int amt)

    {

        if(amt==0)

        {

            return 0;

        }

    }

}

```

```

int[] d = new int[amt+1];

```

```

Arrays.fill(d, Integer.MAX_VALUE);

```

```

d[0]=0;

```

```

for(int i=0; i<=amt; i++){

```

```

    if(d[i]==Integer.MAX_VALUE){
        continue;
    }
    for(int coin: coins)
    {
        if(i<=amt-coin)
        {
            d[i+coin] = Math.min(d[i]+1, d[i+coin]);
        }
    }
}
if(d[amt]==Integer.MAX_VALUE)
{
    return -1;
}
return d[amt];
}}

```

3. Sum Root to Leaf Numbers

Code: class Solution:

```
def sumNumbers(self, root: Optional[TreeNode]) -> int:
```

```
    r = []
```

```
    def d(root, sum):
```

```
        nonlocal r
```

```
        sum += root.val
```

```
        if root.left == None and root.right == None:
```

```
            r.append(sum)
```

```
            return
```

```
        if root.left: d(root.left, sum * 10)
```

```
        if root.right: d(root.right, sum * 10)
```

```
    if root:
```

```
        d(root, 0)
```

```
return sum(r)
```

```
return 0
```

4.Path Sum

Code:

class Solution:

```
def hasPathSum(self, root: TreeNode, sum: int) -> bool:
```

```
    def Help(node, t1):
```

```
        if not node:
```

```
            return False
```

```
            t1 -= node.val
```

```
            if (t1 == 0) and (not node.left) and (not node.right):
```

```
                return True
```

```
            return Help(node.left, t1) or Help(node.right, t1)
```

```
        if not root:
```

```
            return
```

```
        return Help(root, sum)
```

5. Maximum Depth of Binary Tree

Code:

class Solution(object):

```
def maxDepth(self, root):
```

```
    """
```

```
    :type root: TreeNode
```

```
    :rtype: int
```

```
    """
```

```
    return 1 + max(self.maxDepth(root.left), self.maxDepth(root.right)) if root else 0
```