

1

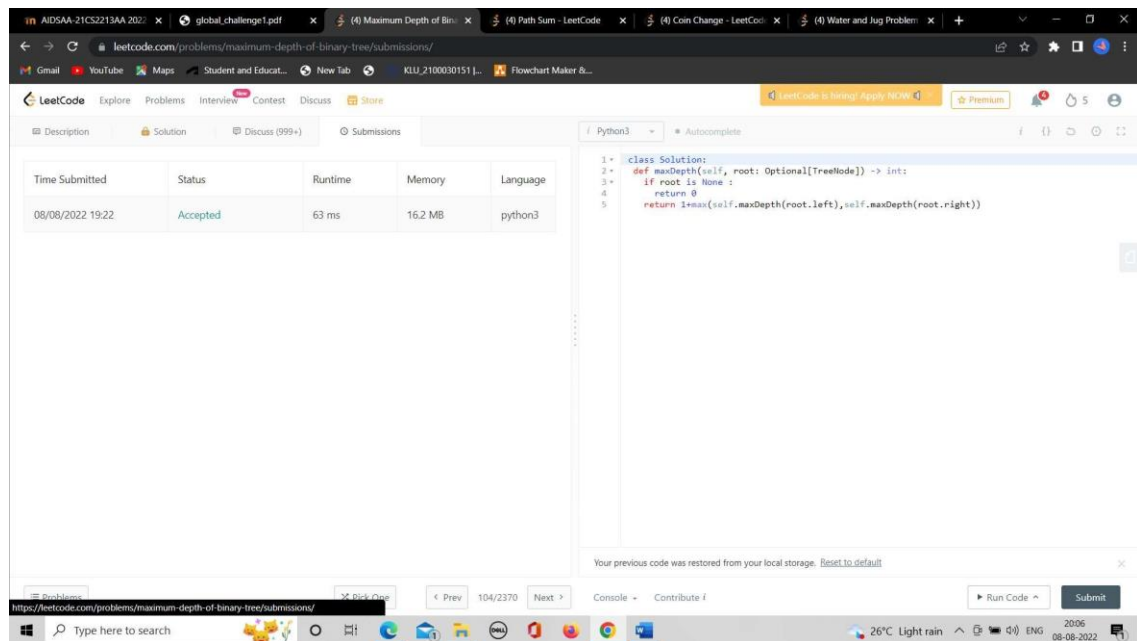
class Solution:

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

if root is None :

return 0 return

1+max(self.maxDepth(root.left),self.maxDepth(root.right))



2 class Solution: def hasPathSum(self, root: Optional[TreeNode],

targetSum: int) -> bool:

if root is None :

return 0 if not(root.left

or root.right) :

return 1 if root.val==targetSum else 0

targetSum = targetSum-root.val return self.hasPathSum(root.left,targetSum) or

self.hasPathSum(root.right,targetSum)

LeetCode.com/problems/path-sum/submissions/

Time Submitted	Status	Runtime	Memory	Language
08/08/2022 19:30	Accepted	83 ms	15.2 MB	python3

```

1 # Definition for a binary tree node.
2 # class TreeNode:
3 #     def __init__(self, val=0, left=None, right=None):
4 #         self.val = val
5 #         self.left = left
6 #         self.right = right
7
8 class Solution:
9     def hasPathSum(self, root: Optional[TreeNode], targetSum: int) -> bool:
10         if root is None:
11             return False
12         if not(root.left or root.right):
13             return True if root.val==targetSum else False
14         targetSum -= root.val
15         return self.hasPathSum(root.left,targetSum) or self.hasPathSum(root.right,targetSum)

```

4 class

Solution:

```
def coinChange(self,coins: List[int],amount: int)->int:
```

```
    req=[float('inf')]*(amount+1)
```

```
    req[0]=0
```

```
    for coin in coins:
```

```
        for x in range(coin,amount+1):
```

```
            req[x]=min(req[x],req[x-coin]+1)        return
```

```
    req[amount] if req[amount] !=float('inf')else-1
```

The screenshot shows the LeetCode website with the 'Coin Change' problem selected. The submission table is as follows:

Time Submitted	Status	Runtime	Memory	Language
08/08/2022 19:45	Accepted	2011 ms	14.3 MB	python3
08/08/2022 19:36	Wrong Answer	N/A	N/A	python3

The code editor shows the following Python solution:

```

1 class Solution:
2     def coinChange(self, coins: List[int], amount: int) -> int:
3         req = [float('inf')] * (amount + 1)
4         req[0] = 0
5
6         for coin in coins:
7             for x in range(coin, amount + 1):
8                 req[x] = min(req[x], req[x - coin] + 1)
9         return req[amount] if req[amount] != float('inf') else -1

```

class Solution:

def canMeasureWater(self, jug1Capacity: int, jug2Capacity: int, targetCapacity: int) -> bool:

def gcd(a,b) : while b:

a,b=b,a%b return a if targetCapacity >

jug1Capacity+jug2Capacity :

return False

return (targetCapacity %
gcd(jug1Capacity,jug2Capacity))==0

The screenshot shows the LeetCode website with the 'Water and Jug Problem' selected. The submission table is as follows:

Time Submitted	Status	Runtime	Memory	Language
08/08/2022 19:49	Accepted	46 ms	13.9 MB	python3
08/08/2022 19:48	Runtime Error	N/A	N/A	python3

The code editor shows the following Python solution:

```

1 class Solution:
2     def canMeasureWater(self, jug1Capacity: int, jug2Capacity: int, targetCapacity: int) ->
3     bool:
4         def gcd(a,b) :
5             while b:
6                 a,b=b,a%b
7             return a
8         if targetCapacity > jug1Capacity+jug2Capacity :
9             return False
10        return (targetCapacity % gcd(jug1Capacity,jug2Capacity))==0

```

HACKER RANK-KLU_2100030152

```
1 def
displayPathtoPrincess(n,grid):
#first, find where p is located
flag = True    n = n-1
while(flag):    if grid[0][n] ==
'p':
    flag = False
a = 0          b = n
if grid[n][0] == 'p':
    flag = False
a = n          b = 0
if grid[0][0] == 'p':
    flag =
False          a =
0              b = 0
    if grid[n][n] ==
'p':          flag =
False          a = n
b = n
    #else:
        #print("Something broke?") #Why does this execute?
        y = a - int(n/2)
x = b - int(n/2)
while 0 != y:    if
y < 0:          y =
y+1
print("UP")      if y
> 0:            y = y-1
print("DOWN")
```

```

while 0 != x:
    if
x < 0:
    x = x+1
print("LEFT")
if
x > 0:
    x = x-1
print("RIGHT")
m = int(input())
grid = []
for i in
range(0, m):
    grid.append(input().strip())

```

```
displayPathtoPrincess(m,grid)
```

1	klu_2100031226	13.90	14:53:22	?
1	h2100031107	13.90	16:20:28	🇮🇳
1	klu_2100031238	Compare	30:59:59	🇮🇳
1	h2100030834	13.90	32:05:11	🇮🇳
1	klu_2100030022	13.90	32:29:34	🇮🇳
1	pavankashyapvem1	13.90	32:54:13	🇮🇳
1	h2100032294	13.90	38:44:50	🇮🇳
1	klu_2100030152	13.90	100:29:22	🇮🇳

<< < 1 2 3 4 5 6 7 8 9 10 > >>

Items per page: 10

```
2 def
```

```
nextMove(n,r,c,grid):
```

```
    pos_col_m = c
```

```
    pos_row_m = r
```

```
    pos_col_p =
```

```
pos_row_p = 0
```

```
    for i in range(n):
```

```

        line = len(grid[i])
        for j in range(line):
            if grid[i][j] == 'p':
                pos_row_p = i
                pos_col_p = j

        # Verify the positions
of the bot with the
princess

        if pos_row_m <
pos_row_p:
            pos_row_m =
pos_row_m + 1
            return 'DOWN'

        elif pos_row_m >
pos_row_p:
            pos_row_m =
pos_row_m - 1
            return 'UP'

        if pos_col_m <
pos_col_p:
            pos_col_m =
pos_col_m + 1
            return 'RIGHT'

        elif pos_col_m >
pos_col_p:
            pos_col_m =
pos_col_m - 1
            return 'LEFT'

```

```

# Set the data
n = int(input())
r,c = [int(i) for i in
input().strip().split()]
grid = []
for i in range(0, n):
    grid.append(input())

# print the first move here
print(nextMove(n,r,c,grid
))

```

1	h2100031107	17.50	16:22:32	
1	klu_2100031238	17.50	31:33:19	
1	klu_2100030022	17.50	32:53:06	
1	h2100030719	17.50	40:25:34	
1	h2100030133	17.50	40:27:46	
1	h2100030867	17.50	40:28:29	
1	h2100030920	17.50	41:26:18	
1	KLU_2100031509	17.50	56:23:38	
1	klu_2100030152	17.50	396:54:38	

3 import

copy

```

pacman_a, pacman_b= list(map(int,
input().split())) food_a, food_b = list(map(int,
input().split())) h, m = list(map(int, input().split()))

grid = []

```

```

node_expanded = []
stack = []
answer_routes = None

for i in range(0, h):
    grid.append(list(map(str, input()))))

directions = [[-1, 0], [0, -1], [0, 1], [1, 0]]

stack.append([pacman_a, pacman_b, []])
while len(stack) > 0:    a, b, r =
stack.pop()    routes = copy.deepcopy(r)
routes.append([a, b])

    node_expanded.append([a, b])

    if a == food_a and b ==
food_b:        if answer_routes ==
None:            answer_routes =
routes
                break

    for direction in directions:
        next_a, next_b = a + direction[0], b + direction[1]        if
next_a < 0 or next_a >= h or next_b < 0 and next_b >= h:
            continue

        if grid[next_a][next_b] == "-" or grid[next_a][next_b] == ".":
            grid[next_a][next_b] = '='
stack.append([next_a,        next_b,        routes])

```



```

print(str(len(node_expanded)))    for    point    in
node_expanded:
    print(str(point[0]) + " " + str(point[1]))

```

```

print(str(len(answer_routes) - 1)) for
point    in    answer_routes:
print(str(point[0]) + " " + str(point[1]))

```

1	KLU_2100031509	15.00	56:26:20	
1	KLU_2100031334	15.00	65:37:32	
1	KLU_2100031395	15.00	100:45:03	
1	KLU_2100030802	15.00	102:07:37	
1	h2100030834	15.00	102:14:27	
1	KLU_2100030432	15.00	102:35:55	
1	h2100030473	15.00	102:55:23	
1	klu_2100030184	15.00	103:03:33	
1	klu_2100030152	15.00	396:59:48	

Navigation: 1 2 3 4 5 6 7 8 9 10 Items per page: 10

4 import copy

```

pacman_x, pacman_y = list(map(int, input().split()))
food_x, food_y = list(map(int, input().split()))
n, m = list(map(int, input().split()))
grid = []
queue = []
answer_routes = None

```

```

for i in range(0, n):
    grid.append(list(map(str, input())))

```

```

directions = [[-1, 0], [0, -1], [0, 1], [1, 0]]

```

```

queue.append([pacman_x, pacman_y, [], 0])
while len(queue) > 0:
    x, y, r, score = queue.pop(0)
    routes = copy.deepcopy(r)
    routes.append([x, y])

    if x == food_x and y == food_y:
        if answer_routes == None:
            answer_routes = routes
        break











    possible_moves = []
    for direction in directions:
        next_x, next_y = x + direction[0], y + direction[1]
        if next_x < 0 or next_x >= n or next_y < 0 or next_y >= n:
            continue

        if grid[next_x][next_y] == "-" or grid[next_x][next_y] == ".":
            grid[next_x][next_y] = '='
            possible_moves.append([next_x, next_y, score + abs(food_x - next_x) + abs(food_y -
next_y)])

    possible_moves.sort(key = lambda x: x[2])
    for move in possible_moves:
        queue.append([move[0], move[1], routes, score])

print(str(len(answer_routes) - 1))
for point in answer_routes:
    print(str(point[0]) + " " + str(point[1]))

```

1	KLU_2100031395	5.00	100:38:55	
1	h2100030834	5.00	102:36:57	
1	h2100032473	5.00	150:05:32	
1	KLU_2100031350	5.00	153:05:59	
1	h2100032404	5.00	155:04:21	
1	h21000324321	5.00	157:00:15	
1	klu_2100032420	5.00	157:02:16	
1	klu_2100031416	5.00	158:49:04	
1	h2100030103	5.00	159:00:37	
1	klu_2100030152	5.00	397:02:39	

<< < 1 2 3 4 5 6 7 8 9 10 > >>

Items per page: 10

5 import copy

```
pacman_x, pacman_y = list(map(int, input().split()))
```

```
food_x, food_y = list(map(int, input().split()))
```

```
n, m = list(map(int, input().split()))
```

```
grid = []
```

```
node_expanded = []
```

```
queue = []
```

```
answer_routes = None
```

```
for i in range(0, n):
```

```
    grid.append(list(map(str, input())))
```

```
directions = [[-1, 0], [0, -1], [0, 1], [1, 0]]
```

```
queue.append([pacman_x, pacman_y, []])
```

```
while len(queue) > 0:
```

```
    x, y, r = queue.pop(0)
```

```
    routes = copy.deepcopy(r)
```

```
    routes.append([x, y])
```

```
    node_expanded.append([x, y])
```

```

if x == food_x and y == food_y:
    if answer_routes == None:
        answer_routes = routes
    break

for direction in directions:
    next_x, next_y = x + direction[0], y + direction[1]
    if next_x < 0 or next_x >= n or next_y < 0 and next_y >= n:
        continue

    if grid[next_x][next_y] == "-" or grid[next_x][next_y] == ".":
        grid[next_x][next_y] = '='
        queue.append([next_x, next_y, routes])










print(str(len(node_expanded)))
for point in node_expanded:
    print(str(point[0]) + " " + str(point[1]))

print(str(len(answer_routes) - 1))
for point in answer_routes:
    print(str(point[0]) + " " + str(point[1]))

```

PacMan - BFS Leaderboard | AID: X +

hackerrank.com/contests/aids-globalchallenge1/challenges/pacman-bfs/leaderboard

1	h2100030834	10.00	102:22:25	
1	KLU_2100031509	10.00	111:48:18	
1	h2100032473	10.00	150:06:43	
1	KLU_2100031350	10.00	153:09:35	
1	h2100032404	10.00	155:16:04	
1	klu_2100031425	10.00	156:09:41	
1	klu_2100032420	10.00	157:07:06	
1	h21000324321	10.00	157:09:05	
1	klu_2100030152	10.00	397:04:35	

<< < 1 2 3 4 5 6 7 8 9 10 > >>

Items per page: 10

