All Contests > SLIIT Codefest 2022 Hackathon - First round > Time to be healthy

Time to be healthy

Problem Submissions Leaderboard

The interns at 99X Technology have come up with a game in which to make the employees of companies - in general - to be more physically engaging and active

Solved: 51
Attempted: 53

The game is conducted as follows:

Every morning the staff will receive an email with the number of floors they have to visit, points will be given for travelling each cubicle in the floor starting from the Start cubicle and the ending in the End cubicle. The person who gets the most number of points will win the game. If a person wins continuously three times, they will be rewarded.

Chillie is a software engineer and is interested in winning this game. He saw an interesting pattern in the game and thought of developing a program which will help him to identify exactly which cubicles he need to travel and the maximum points that can be gained in order to win the competition.

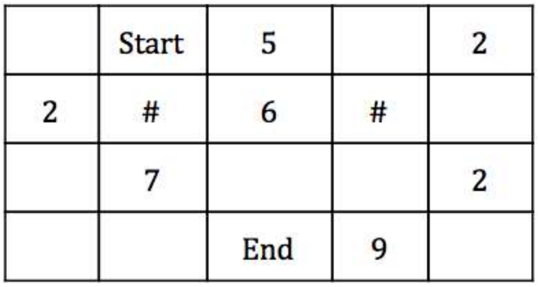
He received the first email with following content:

Number of floors: 4
Number of cubicles in each floor: 5
End cubicle: 4th floor 3rd cubicle
Points :
1st floor 3rd cubicle: 5
1st floor 5th cubicle: 2
2nd floor 1st cubicle: 2
2nd floor 3rd cubicle: 6
3rd floor 2nd cubicle: 7
3rd floor 5th cubicle: 2
4th floor 4th cubicle: 9

Cubicles that cannot travelled: 2nd floor 2nd cubicle 2nd floor 4th cubicle

Starting point will be your cubicle. Points will not be given for it. Cubicles with no points will be given 0 points.

After receiving the email he noted down the information in a grid as given below: (Chille is in the first floor 2nd cubicle and he noted down the cubicles that cannot be travelled in #)



	Start	5		2
2	#	6	#	
	7			2
		End	9	

Chillie's goal is to find a path to go from the start to end earning the maximum points. Also he can only travel either to **left, right or down** and **he cannot travel diagonally. (Each cubicle can be travelled only once and the end will always be in the last floor)**

Input Format

First line will consist **m**, **n** integers such that m is the number of floors and n is the number of cubicles
The rest of the **m** lines will consist of a table as shown above with the relevant information (S will be the start point,E will be the End point and # will represent the blocking points.)

Constraints

2 < **m , n** < 100

Output Format

The output should be a single line including the **maximum number of points** he can gain when going from the start to end.

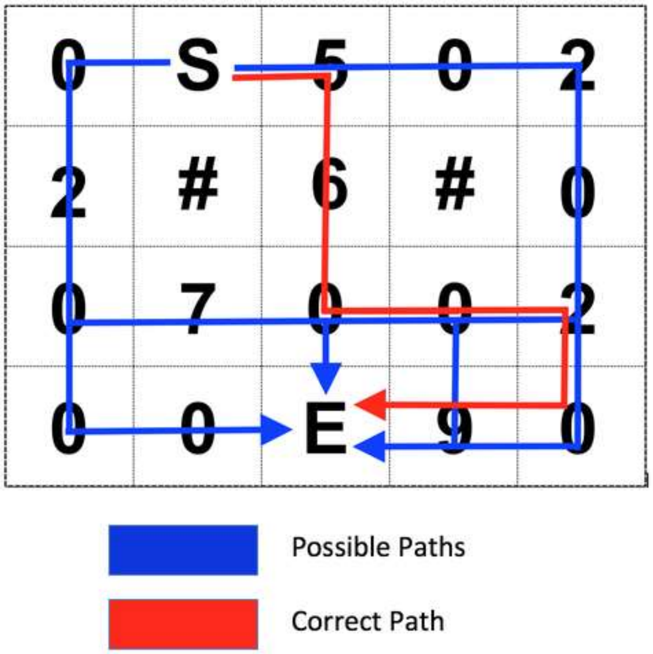
Sample Input 0

```
4,5
0S502
2#6#0
07002
00E90
```

Sample Output 0

```
22
```

Explanation 0



Sample Input 1

```
3,3
0S1
4#0
E32
```

Sample Output 1

```
6
```

Contest ends in a day

Submissions: 20
Max Score: 75

Rate This Challenge:
☆☆☆☆☆

More

Python 3

```
r,c = input().split(",")
r = int(r)
c = int(c)

matrix = []

sum_list = []

pointer_list = []

for i in range(r):
    row_input = input()
    temp = []
    for j in range(c):
        temp.append(row_input[j])
    matrix.append(temp)

start_end = []

for i in range(r):
    for j in range(c):
        if (matrix[i][j] == "S" or matrix[i][j] == "E"):
            start_end.append(i)
            start_end.append(j)
            break

s1 = start_end[0]
s2 = start_end[1]
e1 = start_end[2]
e2 = start_end[3]

def find_paths(matrix,path,paths,i,j,movement):

    if (i == e1 and j == e2):
        path = path+[matrix[i][j]]
        paths.append(path)
        return

    path.append(matrix[i][j])

    temp = movement

    # left (-1)
    if ((0<=i<=r-1) and (0<=j-1<=c-2)):
        if (movement != 1):
            movement = -1
            find_paths(matrix,path,paths,i,j-1,movement)
        else:
            pass

    movement = temp

    # right (1)
    if ((0<=i<=r-1) and (0<=j+1<=c-1)):
        if (movement != -1):
            movement = 1
            find_paths(matrix,path,paths,i,j+1,movement)
        else:
            pass

    movement = temp

    # down (0)
    if ((0<=i+1<=r-1) and (0<=j<=c-1)):
        find_paths(matrix,path,paths,i+1,j,0)

    path.pop()

path = []
paths = []

find_paths(matrix,path,paths,s1,s2,0)

sum_list = []

for list in paths:
    sum = 0
    if "#" not in list:
        for i in range(1,len(list)-1):
            sum += int(list[i])
        sum_list.append(sum)

print(max(sum_list))
```

Line: 1 Col: 1

Upload Code as File □ Test against custom input Run Code Submit Code

Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy |