LACS-Elite-Part007

Solved Challenges 0/1

Back To Challenges List

Corona Virus

ID:12920 **Solved By 254 Users**

An integer matrix of size **RxC** containing only the values **0**, **1** and **2** is given as the input to the program. The value **0** indicates an **empty** space, the value **1** indicates a person is **healthy** and the value **2** indicates a person is infected by the **corona virus**. Every day the virus is spread from infected person to other persons (all four adjacent persons). The program must print the minimum number of days required to spread the coronavirus to all individuals. If all the persons can not be affected by the corona virus, the program must print **-1** as the output.

Boundary Condition(s):
1 <= R, C <= 1000

Input Format:
The first line contains R and C separated by a space.
The next R lines, each containing C integers separated by a space.

Output Format:
The first line contains -1 or the minimum number of days required to spread the coronavirus to all individuals.

Example Input/Output 1:
Input:
3 5
2 1 0 2 1
1 0 1 2 1
1 0 0 2 1

Output:
2

Explanation:
After Day **1**:
2 2 0 2 2
2 0 2 2 2
1 0 0 2 2

After Day **2**:
2 2 0 2 2
2 0 2 2 2
2 0 0 2 2

Example Input/Output 2:
Input:
3 5
2 1 0 2 1
0 0 1 2 1
1 0 0 2 1

Output:
-1

Max Execution Time Limit: 1000 millisecs

Ambiance

Python3 (3.x)

Reset

```
string = input().split()
R = int(string[0])
C = int(string[1])

matrix = []
healthy = 0
days = 0
queue = []
for i in range(R):
    a = list(map(int, input().split()))
    matrix.append(a)

for rows in range(R):
    for cols in range(C):
        if(matrix[rows][cols] == 1):
            healthy = healthy +1;
        elif(matrix[rows][cols] == 2):
            queue.append(rows*C + cols)

queue.append(-1)

while(len(queue)!=0):
    node = queue.pop(0)

    if(node == -1):
        if(len(queue)!=0):
```

```
27              days = days +1
28              queue.append(-1)
29          continue
30
31      row = node//C
32      col = node%C
33
34      if(col!=0 and matrix[row][col-1] == 1):
35          matrix[row][col-1] = 2
36          queue.append(row*C+ col-1)
37          healthy = healthy -1
38
39      if(col!=C-1 and matrix[row][col+1] == 1):
40          matrix[row][col+1] = 2
41          queue.append(row*C + col+1)
42          healthy = healthy-1
43
44      if(row!=0 and matrix[row-1][col] == 1):
45          matrix[row-1][col] = 2
46          queue.append((row-1)*C + col)
47          healthy = healthy-1
48
49      if(row!=R-1 and matrix[row+1][col] == 1):
50          matrix[row+1][col] = 2
51          queue.append((row+1)*C + col)
52          healthy = healthy-1
53
54  if(healthy != 0):
55      print(-1)
56  else:
```

Close

SkillRack Message : Corona Virus

```
string = input().split()
R = int(string[0])
C = int(string[1])

matrix = []
healthy = 0
days = 0
queue = []
for i in range(R):
a = list(map(int, input().split()))
matrix.append(a)

for rows in range(R):
for cols in range(C):
if(matrix[rows][cols] == 1):
healthy = healthy +1;
elif(matrix[rows][cols] == 2):
```

```
queue.append(rows*C + cols)

queue.append(-1)

while(len(queue)!=0):
node = queue.pop(0)

if(node == -1):
if(len(queue)!=0):
days = days +1
queue.append(-1)
else:
continue

row = node//C;
col = node%C;

if(col!=0 and matrix[row][col-1] == 1):
matrix[row][col-1] = 2
queue.append(row*C+ col-1)
healthy = healthy -1

if(col!=C-1 and matrix[row][col+1] == 1):
matrix[row][col+1] = 2
queue.append(row*C + col+1)
healthy = healthy-1

if(row!=0 and matrix[row-1][col] == 1):
matrix[row-1][col] = 2
queue.append((row-1)*C + col)
healthy = healthy-1

if(row!=R-1 and matrix[row+1][col] == 1):
matrix[row+1][col] = 2
queue.append((row+1)*C + col)
healthy = healthy-1


print(days)
```

Code did not pass the execution

8 **Private (Hidden) Test Cases Failed.**

MEM: 0.09765625 MB **CPU: 0.01**

1 **Test Case(s) Failed Due To Timeout / Resource Limit**

This can be due to infinite loops or your program is not efficient in processing huge input data within the given time limit.

Error Message:

Your program did not execute successfully: SoftTimeLimitExceeded()

Save Run

Run with a custom test case (Input/Output)