**LACS-Elite-Part006**

Solved Challenges 0/1

Back To Challenges List

## Street Travel Count

**ID:12884 Solved By 273 Users**

Mr.X has a bike and is travelling in a town which has **N** horizontal (West to East direction) and N vertical (North to South direction) streets. At the meeting junctions of these horizontal and vertical streets there may be a block. If there is a block Mr.X can take diversion to any other street and travel to his destination. A value of 1 indicates that a junction (meeting point of two roads) is NOT blocked and a value of 0 indicates that a junction is blocked. The streets are numbered from 0 to N-1 (similar to array indices). The source (starting junction of Mr.X and the destination junctions details are passed as the input. The program must print the number of streets through which Mr.X must travel to travel from the source to destination.

**Boundary Condition(s):**
1 <= N <= 100

**Input Format:**
The first line contains N.
The next N lines each containing N values 1 or 0 separated by a space.
The $(N+2)^{nd}$ line contains the source junction co-ordinates separated by a space.
The $(N+3)^{rd}$ line contains the destination junction co-ordinates separated by a space.

**Output Format:**
The number of streets Mr.X must travel to travel from source to destination.

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

Output:
3

Explanation:
The source is (0,0) indicated as S and the destination (0,2) by D.
**S** 0 **D**
1 0 1
1 1 1

0 implies block. Hence Mr.X must travel along 1s. Hence the path to travel is denoted by letter P from S to D.
**S** 0 **D**
**P** 0 **P**
**P P P**

Hence we can notice that Mr.X must travel through 3 streets to reach the destination.

**Example Input/Output 2:**
Input:
4
1 1 1 0
0 0 1 1
1 1 0 1
0 1 1 1
0 1
2 0

Output:
7

Explanation:
The path denoted by the letter P is
1 **S P** 0
0 0 **P P**
**D P** 0 **P**
0 **P P P**
Hence we can notice that Mr.X must travel through 7 streets to reach the destination.

**Example Input/Output 3:**
Input:
4
1 1 1 0
0 0 1 1
1 1 0 1
0 1 1 1
0 1
2 1

Output:
6

**Max Execution Time Limit: 2000 millisecs**

Ambiance

Python3 (3.x )

Reset

```
def getRelated(matrix, node, N):
    nodes = []
    nodeRow = node//N
    nodeCol = node%N
    for col in range(nodeCol-1,-1,-1):
        if(matrix[nodeRow][col] == 1):
            nodes.append(nodeRow * N + col)
```

```
        else:
            break

    for col in range(nodeCol+1,N):
        if(matrix[nodeRow][col] == 1):
            nodes.append(nodeCol *N + col)
        else:
            break

    for row in range(nodeRow-1 , -1,-1):
        if(matrix[row][nodeCol] == 1):
            nodes.append(row*N + nodeCol)
        else:
            break

    for row in range(nodeRow+1,N):
        if(matrix[row][nodeCol] == 1):
            nodes.append(row*N + nodeCol)
        else:
            break

    return nodes


N = int(input())
matrix = []

for rows in range(N):
    a = list(map(int, input().split()))
    matrix.append(a)

source_cord = [int(i) for i in input().split()]
source_r = source_cord[0]
source_c = source_cord[1]

dest_cord = [int(i) for i in input().split()]
dest_r = dest_cord[0]
dest_c = dest_cord[1]

source = (source_r)*N + source_c
destination = (dest_r)*N + dest_c

visited = [False for ctr in range(N*N)]
streets = [0 for ctr in range(N*N)]

queue = []
queue.append(source)
visited[source] = True
streets[source] = 0


```

```
found = False
while(len(queue)!=0):
    node = queue.pop(0)
    nodes = getRelated(matrix,node,N)
    for relNode in nodes:
        if(visited[relNode] == False):
            queue.append(relNode)
            visited[relNode] = True
            streets[relNode] = 1 + streets[node]

            if(relNode == destination):
                print(streets[relNode])
            else:
                found = True
                break
    if(found == True):
        break


```

Save Run

Run with a custom test case (Input/Output)