

**Graph Theory - S001 (E017)**

Solved Challenges 0/2

[Back To Challenges List](#)**Path Exists from Source to Destination Cell****ID:11090    Solved By 655 Users**

The program must accept a matrix of size **R\*C** and the indices of two cells (Source and Destination) in the matrix as the input. The matrix contains only **1's** and **0's**. The cell value **1** indicates the presence of a path. The cell value **0** indicates the presence of a stone (i.e., no path). The movement from one cell to another can be in the **left, right, bottom** and **top** directions. The program must print **yes** if there is a path from the given source cell to the destination cell. Else the program must print **no** as the output.

**Boundary Condition(s):** $2 \leq R, C \leq 50$ **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.

The (R+2)<sup>nd</sup> line contains two integers representing the indices of the source cell.

The (R+3)<sup>rd</sup> line contains two integers representing the indices of the destination cell.

**Output Format:**

The first line contains yes or no.

**Example Input/Output 1:**

Input:

```
4 5
1 0 1 1 0
0 1 0 1 1
1 1 0 1 0
1 1 1 1 1
1 1
1 4
```

Output:

```
yes
```

Explanation:

One of the possible paths from the source cell to the destination cell in the matrix is highlighted below.

1 0 1 1 0

0 1 0 1 1

1 1 0 1 0

1 1 1 1 1

**Example Input/Output 2:**

Input:

3 3

1 0 1

0 1 1

1 0 1

0 2

2 0

Output:

no

**Max Execution Time Limit: 500 millisecs**

Ambiance

C ( gcc 8.x)



Reset

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  int R,C,sourceR,sourceC,destR,destC,found=0;
5
6  void traverse(int matrix[R][C], int row,int col)
7  {
8      if(row>=0 && row<R && col>=0 && col<C)
9      {
10         if(row == destR && col == destC)
11         {
12             found = 1;
13             return;
14         }
15
16         if(matrix[row][col]==0 || matrix[row][col]==2)
17             return;
18
19         matrix[row][col] = 2;
20
21
22         traverse(matrix,row,col+1);
23         if(!found)
24             traverse(matrix,row,col-1);
25         if(!found)
26             traverse(matrix,row-1,col);
27         if(!found)
28             traverse(matrix,row+1,col);
29     }
30 }
31
32 int main()
33 {
34     scanf("%d%d", &R,&C);
35     int matrix[R][C];
36     for(int row=0;row<R;row++)
37         for(int col=0;col<C;col++)
38             scanf("%d",&matrix[row][col]);
39
40
41     scanf("%d%d%d%d",&sourceR,&sourceC,&destR,&destC);
42
43     if(matrix[sourceR][sourceC]==1 && matrix[destR][destC]==1)
44     {
45         traverse(matrix,sourceR,sourceC);
46     }
47
48     printf(found==1?"yes":"no");
49 }
```

Code did not pass the execution



TestCase ID: 63478

Input:

```
4 5
1 0 1 1 0
0 1 0 1 1
1 1 0 1 0
1 1 1 1 1
1 1
1 4
```

Expected Output:

yes

Your Program Output:

**Segmentation fault (core dumped)**

Save

Run

☐ Run with a custom test case (Input/Output)