

【华为OD机考 统一考试机试C卷】 欢乐的周末 (C++ Java JavaScript Python C语言)

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

目前在考C卷，经过两个月的收集整理，**C卷真题已基本整理完毕**

抽到原题的概率为2/3到3/3，**也就是最少抽到两道原题。请注意：大家刷完C卷真题，最好要把B卷的真题刷一下，因为C卷的部分真题来自B卷。**

另外订阅专栏还可以联系笔者开通在线 **OJ** 进行刷题，提高刷题效率。

真题目录：华为OD机考机试 真题目录 (C卷 + D卷 + B卷 + A卷) + 考点说明

专栏：2023华为OD机试(B卷+C卷+D卷) (C++JavaJSPy)

华为OD面试真题精选：华为OD面试真题精选

在线OJ：点击立即刷题，模拟真实机考环境

题目描述

小华和小为是很要好的朋友，他们约定周末一起吃饭。

通过手机交流，他们在地图上选择了多个聚餐地点（由于自然地形等原因，部分聚餐地点不可达），求小华和小为都能到达的聚餐地点有多少个？

输入描述

第一行输入m和n，m代表地图的长度，n代表地图的宽度。

第二行开始具体输入地图信息，地图信息包含：

0 为通畅的道路

1 为障碍物（且仅1为障碍物）

2 为小华或者小为，地图中必定有且仅有2个（非障碍物）

3 为被选中的聚餐地点（非障碍物）

输出描述

可以被两方都到达的聚餐地点数量，行末无空格。

用例

输入	4 4 2 1 0 3 0 1 2 1 0 3 0 0 0 0 0 0
输出	2
说明	第一行输入地图的长宽为3和4。 第二行开始为具体的地图，其中：3代表小华和小明选择的聚餐地点；2代表小华或者小明（确保有2个）；0代表可以通行的位置；1代表不可以通行的位置。 此时两者能都能到达的聚餐位置有2处。

输入	4 4 2 1 2 3 0 1 0 0 0 1 0 0 0 1 0 0
输出	0
说明	第一行输入地图的长宽为4和4。 第二行开始为具体的地图，其中：3代表小华和小明选择的聚餐地点；2代表小华或者小明（确保有2个）；0代表可以通行的位置；1代表不可以通行的位置。 由于图中小华和小为之间有个阻隔，此时，没有两人都能到达的聚餐地址，故而返回0。

备注：

地图的长宽为m和n，其中：

4 <= m <= 100

4 <= n <= 100

聚餐的地点数量为 k , 则

$1 < k \leq 100$

C++

```
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  // 定义四个方向的偏移量 (上、下、左、右)
7  const int dirs[4][2] = {{-1, 0}, {1, 0}, {0, 1}, {0, -1}};
8
9  // 深度优先搜索函数
10 bool dfs(int currX, int currY, int targetX, int targetY, vector<vector<int>>& map, vector<vector<vector<bool>>>& visited, int person) {
11     // 如果当前位置就是目标位置, 返回true
12     if (currX == targetX && currY == targetY) {
13         return true;
14     }
15
16     // 遍历四个方向
17     for (int i = 0; i < 4; ++i) {
18         int nextX = currX + dirs[i][0], nextY = currY + dirs[i][1];
19         // 如果下一个位置超出地图范围, 或者是障碍物, 或者已经访问过, 跳过
20         if (nextX < 0 || nextY < 0 || nextX >= map.size() || nextY >= map[0].size() || map[nextX][nextY] == 1 || visited[nextX][nextY][person]) {
21             continue;
22         }
23
24         // 标记下一个位置为已访问
25         visited[nextX][nextY][person] = true;
26         // 递归搜索下一个位置
27         if (dfs(nextX, nextY, targetX, targetY, map, visited, person)) {
28             return true;
29         }
30     }
31
32     return false;
33 }
34
35
```

```
35 int main() {
36     // 输入初始化
37     int m, n;
38     cin >> m >> n;
39     vector<vector<int>> map(m, vector<int>(n));
40     // 使用三维数组visited来记录每个人访问过的位置
41     vector<vector<vector<bool>>> visited(m, vector<vector<bool>>(n, vector<bool>(2)));
42     vector<pair<int, int>> persons, targets;
43
44     // 读取地图信息, 并记录小华和小为的位置以及聚餐地点
45     for (int i = 0; i < m; ++i) {
46         for (int j = 0; j < n; ++j) {
47             cin >> map[i][j];
48             if (map[i][j] == 2) {
49                 persons.emplace_back(i, j);
50             } else if (map[i][j] == 3) {
51                 targets.emplace_back(i, j);
52             }
53         }
54     }
55
56     // 获取小华和小为的位置
57     auto xiaohua = persons[0];
58     auto xiaowei = persons[1];
59     int res = 0;
60
61     // 遍历所有聚餐地点
62     for (const auto& target : targets) {
63         // 重置visited数组
64         visited = vector<vector<vector<bool>>>(m, vector<vector<bool>>(n, vector<bool>(2)));
65         // 判断小华是否能到达目标位置
66         if (dfs(xiaohua.first, xiaohua.second, target.first, target.second, map, visited, 0)) {
67             // 重置visited数组
68             visited = vector<vector<vector<bool>>>(m, vector<vector<bool>>(n, vector<bool>(2)));
69             // 判断小为是否能到达目标位置
70             if (dfs(xiaowei.first, xiaowei.second, target.first, target.second, map, visited, 1)) {
71                 // 如果两个人都能到达目标位置, 结果加1
72                 res++;
73             }
74         }
75     }
76 }
```

```
76     }
77
78     // 输出可以被两人都到达的聚餐地点数量
79     cout << res << endl;
80
81     return 0;
82 }
```

JavaScript

```
1  const readline = require('readline');
2
3  // 定义四个方向的偏移量 (上、下、左、右)
4  const dirs = [[-1, 0], [1, 0], [0, 1], [0, -1]];
5
6  // 深度优先搜索函数
7  function dfs(currX, currY, targetX, targetY, map, visited, person) {
8      // 如果当前位置就是目标位置, 返回true
9      if (currX === targetX && currY === targetY) {
10         return true;
11     }
12
13     // 遍历四个方向
14     for (const dir of dirs) {
15         const nextX = currX + dir[0], nextY = currY + dir[1];
16         // 如果下一个位置超出地图范围, 或者是障碍物, 或者已经访问过, 跳过
17         if (nextX < 0 || nextY < 0 || nextX >= map.length || nextY >= map[0].length || map[nextX][nextY] === 1 || visited[nextX][nextY][person]) {
18             continue;
19         }
20
21         // 标记下一个位置为已访问
22         visited[nextX][nextY][person] = true;
23         // 递归搜索下一个位置
24         if (dfs(nextX, nextY, targetX, targetY, map, visited, person)) {
25             return true;
26         }
27     }
28
29     return false;
30 }
```

```
30 }
31
32 const rl = readline.createInterface({
33   input: process.stdin,
34   output: process.stdout
35 });
36
37 const input = [];
38
39 rl.on('line', (line) => {
40   input.push(line);
41 }).on('close', () => {
42   // 输入初始化
43   const [m, n] = input.shift().split(' ').map(Number);
44   const map = input.splice(0, m).map(row => row.split(' ').map(Number));
45   const visited = Array.from({ length: m }, () => Array.from({ length: n }, () => Array(2).fill(false)));
46   const persons = [];
47   const targets = [];
48
49   // 读取地图信息, 并记录小华和小为的位置以及聚餐地点
50   for (let i = 0; i < m; i++) {
51     for (let j = 0; j < n; j++) {
52       if (map[i][j] === 2) {
53         persons.push([i, j]);
54       } else if (map[i][j] === 3) {
55         targets.push([i, j]);
56       }
57     }
58   }
59
60   // 获取小华和小为的位置
61   const xiaohua = persons[0];
62   const xiaowei = persons[1];
63   let res = 0;
64
65   // 遍历所有聚餐地点
66   for (const target of targets) {
67     // 重置visited数组
68     visited.forEach(row => row.forEach(cell => cell.fill(false)));
69     // 判断小华是否能到达目标位置
70   }
71 }
```

```
71     if (dfs(xiaohua[0], xiaohua[1], target[0], target[1], map, visited, 0)) {
72         // 重置visited数组
73         visited.forEach(row => row.forEach(cell => cell.fill(false)));
74         // 判断小为是否能到达目标位置
75         if (dfs(xiaowei[0], xiaowei[1], target[0], target[1], map, visited, 1)) {
76             // 如果两个人都能到达目标位置, 结果加1
77             res++;
78         }
79     }
80 }
81
82 // 输出可以被两人都到达的聚餐地点数量
83 console.log(res);
84 };
```

Java

```
1 import java.util.Scanner;
2 import java.util.ArrayList;
3 import java.util.List;
4
5 public class Main {
6     // 定义四个方向的偏移量 (上、下、左、右)
7     private static int[][] dirs = {{-1, 0}, {1, 0}, {0, 1}, {0, -1}};
8
9     // 深度优先搜索函数
10    private static boolean dfs(int currX, int currY, int targetX, int targetY, int[][] map, boolean[][][] visited, int person) {
11        // 如果当前位置就是目标位置, 返回true
12        if (currX == targetX && currY == targetY) {
13            return true;
14        }
15
16        // 遍历四个方向
17        for (int[] dir : dirs) {
18            int nextX = currX + dir[0], nextY = currY + dir[1];
19            // 如果下一个位置超出地图范围, 或者是障碍物, 或者已经访问过, 跳过
20            if (nextX < 0 || nextY < 0 || nextX >= map.length || nextY >= map[0].length || map[nextX][nextY] == 1 || visited[nextX][nextY][person]) {
21                continue;
22            }
23        }
24    }
```



```
23
24     // 标记下一个位置为已访问
25     visited[nextX][nextY][person] = true;
26     // 递归搜索下一个位置
27     if (dfs(nextX, nextY, targetX, targetY, map, visited, person)) {
28         return true;
29     }
30 }
31
32 return false;
33 }
34
35 public static void main(String[] args) {
36     // 输入初始化
37     Scanner scanner = new Scanner(System.in);
38     int m = scanner.nextInt();
39     int n = scanner.nextInt();
40     int[][] map = new int[m][n];
41     // 使用三维数组visited来记录每个人访问过的位置
42     boolean[][][] visited = new boolean[m][n][2];
43     List<int[]> persons = new ArrayList<>();
44     List<int[]> targets = new ArrayList<>();
45     // 读取地图信息, 并记录小华和小为的位置以及聚餐地点
46     for (int i = 0; i < m; i++) {
47         for (int j = 0; j < n; j++) {
48             map[i][j] = scanner.nextInt();
49             if (map[i][j] == 2) {
50                 persons.add(new int[]{i, j});
51             } else if (map[i][j] == 3) {
52                 targets.add(new int[]{i, j});
53             }
54         }
55     }
56
57     // 获取小华和小为的位置
58     int[] xiaohua = persons.get(0);
59     int[] xiaowei = persons.get(1);
60     int res = 0;
61     // 遍历所有聚餐地点
62     for (int[] target : targets) {
63
```

```

64         // 重置visited数组
65         visited = new boolean[m][n][2];
66         // 判断小华是否能到达目标位置
67         if (dfs(xiaohua[0], xiaohua[1], target[0], target[1], map, visited, 0)) {
68             // 重置visited数组
69             visited = new boolean[m][n][2];
70             // 判断小为是否能到达目标位置
71             if (dfs(xiaowei[0], xiaowei[1], target[0], target[1], map, visited, 1)) {
72                 // 如果两个人都能到达目标位置, 结果加1
73                 res++;
74             }
75         }
76     }
77     // 输出可以被两人都到达的聚餐地点数量
78     System.out.println(res);
79
80     scanner.close();
81 }
82 }

```

Python

```

1  # 定义四个方向的偏移量 (上、下、左、右)
2  dirs = [[-1, 0], [1, 0], [0, 1], [0, -1]]
3
4  # 深度优先搜索函数
5  def dfs(currX, currY, targetX, targetY, map, visited, person):
6      # 如果当前位置就是目标位置, 返回True
7      if currX == targetX and currY == targetY:
8          return True
9
10     # 遍历四个方向
11     for dir in dirs:
12         nextX, nextY = currX + dir[0], currY + dir[1]
13         # 如果下一个位置超出地图范围, 或者是障碍物, 或者已经访问过, 跳过
14         if nextX < 0 or nextY < 0 or nextX >= len(map) or nextY >= len(map[0]) or map[nextX][nextY] == 1 or visited[nextX][nextY][person]:
15             continue
16
17     # 标记下一个位置为已访问
18
19

```

```
18     visited[nextX][nextY][person] = True
19     # 递归搜索下一个位置
20     if dfs(nextX, nextY, targetX, targetY, map, visited, person):
21         return True
22
23     return False
24
25 # 输入初始化
26 m, n = map(int, input().split())
27 map = [list(map(int, input().split())) for _ in range(m)]
28 # 使用三维数组visited来记录每个人访问过的位置
29 visited = [[[False, False] for _ in range(n)] for _ in range(m)]
30 persons = []
31 targets = []
32
33 # 读取地图信息, 并记录小华和小为的位置以及聚餐地点
34 for i in range(m):
35     for j in range(n):
36         if map[i][j] == 2:
37             persons.append([i, j])
38         elif map[i][j] == 3:
39             targets.append([i, j])
40
41 # 获取小华和小为的位置
42 xiaohua = persons[0]
43 xiaowei = persons[1]
44 res = 0
45
46 # 遍历所有聚餐地点
47 for target in targets:
48     # 重置visited数组
49     visited = [[[False, False] for _ in range(n)] for _ in range(m)]
50     # 判断小华是否能到达目标位置
51     if dfs(xiaohua[0], xiaohua[1], target[0], target[1], map, visited, 0):
52         # 重置visited数组
53         visited = [[[False, False] for _ in range(n)] for _ in range(m)]
54         # 判断小为是否能到达目标位置
55         if dfs(xiaowei[0], xiaowei[1], target[0], target[1], map, visited, 1):
56             # 如果两个人都能到达目标位置, 结果加1
57             res += 1
58
59 --
```

```
59 |
60 | # 输出可以被两人都到达的聚餐地点数量
61 | print(res)
```

C语言

```
1 | #include <stdio.h>
2 | #include <stdbool.h>
3 |
4 | #define MAX_SIZE 100 // 定义地图的最大尺寸
5 |
6 | // 定义四个方向的偏移量 (上、下、左、右)
7 | const int dirs[4][2] = {{-1, 0}, {1, 0}, {0, 1}, {0, -1}};
8 |
9 | // 深度优先搜索函数
10 | bool dfs(int currX, int currY, int targetX, int targetY, int m, int n, int map[MAX_SIZE][MAX_SIZE], bool visited[MAX_SIZE][MAX_SIZE], int person) {
11 |     // 如果当前位置就是目标位置, 返回true
12 |     if (currX == targetX && currY == targetY) {
13 |         return true;
14 |     }
15 |
16 |     // 遍历四个方向
17 |     for (int i = 0; i < 4; ++i) {
18 |         int nextX = currX + dirs[i][0], nextY = currY + dirs[i][1];
19 |         // 如果下一个位置超出地图范围, 或者是障碍物, 或者已经访问过, 跳过
20 |         if (nextX < 0 || nextY < 0 || nextX >= m || nextY >= n || map[nextX][nextY] == 1 || visited[nextX][nextY]) {
21 |             continue;
22 |         }
23 |
24 |         // 标记下一个位置为已访问
25 |         visited[nextX][nextY] = true;
26 |         // 递归搜索下一个位置
27 |         if (dfs(nextX, nextY, targetX, targetY, m, n, map, visited, person)) {
28 |             return true;
29 |         }
30 |     }
31 |
32 |     return false;
33 | }
```

```
34
35 int main() {
36     int m, n;
37     scanf("%d %d", &m, &n); // 读取地图的长和宽
38     int map[MAX_SIZE][MAX_SIZE]; // 地图数组
39     bool visited[MAX_SIZE][MAX_SIZE]; // 访问标记数组
40     int persons[2][2], targets[MAX_SIZE][2]; // 人物和目标位置
41     int personsCount = 0, targetsCount = 0;
42
43     // 读取地图信息, 并记录小华和小为的位置以及聚餐地点
44     for (int i = 0; i < m; ++i) {
45         for (int j = 0; j < n; ++j) {
46             scanf("%d", &map[i][j]);
47             if (map[i][j] == 2) {
48                 persons[personsCount][0] = i;
49                 persons[personsCount][1] = j;
50                 personsCount++;
51             } else if (map[i][j] == 3) {
52                 targets[targetsCount][0] = i;
53                 targets[targetsCount][1] = j;
54                 targetsCount++;
55             }
56         }
57     }
58
59     int res = 0; // 可以被两人都到达的聚餐地点数量
60
61     // 遍历所有聚餐地点
62     for (int i = 0; i < targetsCount; ++i) {
63         bool canReach = true; // 标记是否都可以到达
64         for (int p = 0; p < 2; ++p) { // 对于小华和小为
65             // 重置visited数组
66             memset(visited, 0, sizeof(visited));
67             // 判断是否能到达目标位置
68             if (!dfs(persons[p][0], persons[p][1], targets[i][0], targets[i][1], m, n, map, visited, p)) {
69                 canReach = false;
70                 break;
71             }
72         }
73         if (canReach) {
74             res++;
75         }
76     }
77     printf("%d", res);
78     return 0;
79 }
```

```
75         res++;  
76     }  
77 }  
78  
79 printf("%d\n", res); // 输出结果  
80  
81 return 0;  
}
```

完整用例

用例1

```
4 4  
2 1 0 3  
0 1 2 1  
0 3 0 0  
0 0 0 0
```

用例2

```
4 4  
2 1 2 3  
0 1 0 0  
0 1 0 0  
0 1 0 0
```

用例3

```
3 3  
2 1 0  
0 1 3  
0 0 2
```

用例4

```
3 3  
2 0 0
```

1 1 0

0 0 2

用例5

5 5

2 1 0 0 3

0 1 2 1 0

0 3 0 0 0

0 0 0 0 0

0 0 0 0 0

用例6

3 4

2 1 0 3

0 1 2 1

0 3 0 0

用例7

5 4

2 1 0 0

0 1 2 1

0 3 0 0

0 0 0 0

3 0 0 0

用例8

5 4

2 1 0 0

0 1 2 1

0 3 0 0

1 1 1 1

3 0 0 0

用例9

5 5
2 1 0 0 3
0 1 2 1 0
0 3 0 0 0
0 0 0 0 0
0 0 0 0 0

用例10

6 6
2 1 0 0 0 3
0 1 2 1 0 0
0 3 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
3 0 0 0 0 0

文章目录

- 华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷
 - 题目描述
 - 输入描述
 - 输出描述
 - 用例
 - C++
 - JavaScript
 - Java
 - Python
 - C语言
 - 完整用例
 - 用例1
 - 用例2
 - 用例3
 - 用例4

用例5

用例6

用例7

用例8

用例9

用例10

机考真题 华为OD



CSDN @算法大师