

# 【DFS/BFS】-广播服务器

## 题目描述与示例

### 题目描述

服务器连接方式包括直接相连，间接连接。A 和 B 直接连接，B 和 C 直接连接，则 A 和 C 间接连接。

直接连接和间接连接都可以发送广播。

给出一个大小为  $N \times N$  的二维矩阵 `matrix`，代表  $N$  个服务器。`matrix[i][j] = 1`，则代表  $i$  和  $j$  直接连接；`matrix[i][j] = 0` 时，代表  $i$  和  $j$  不直接连接。`matrix[i][j] == 1`，即自己和自己直接连接。

计算初始需要给几台服务器广播，才可以使每个服务器都收到广播。

### 输入

输入为  $N$  行，每行有  $N$  个数字，为 0 或 1，由空格分隔，构成  $N \times N$  的二维矩阵 `matrix`， $N$  的范围为  $1 \leq N \leq 40$ 。

### 输出

输出一个数字，为需要广播的服务器的数量。

### 示例一

#### 输入

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

#### 输出

```
1 3
```

### 示例二

## 输入

```
1 1 1
2 1 1
```

## 输出

```
1 1
```

## 解题思路

本题和 [LC547.省份数量](#) 不能说毫无联系，只能说一模一样。

## 代码

### 解法一：BFS

#### Python

```
1 # 题目：2024E-广播服务器
2 # 分值：200
3 # 作者：闭着眼睛学数理化
4 # 算法：BFS
5 # 代码看不懂的地方，请直接在群上提问
6
7
8 from collections import deque
9
10 isConnected = list()
11 # 先输入第一行
12 isConnected.append(list(map(int, input().split())))
13 # 根据第一行的长度，得到n
14 n = len(isConnected[0])
15 # 输入剩余的n-1行
16 for _ in range(n-1):
17     isConnected.append(list(map(int, input().split())))
18
```

```

19 ans = 0
20 checkList = [0] * n      # 构建检查数组checkList
21
22 # 遍历每一个服务器
23 for i in range(n):
24     if checkList[i] == 0:      # 若服务器i未检查过
25         q = deque([i])        # 把服务器i加入q中，作为BFS的起始位置
26         checkList[i] = 1      # 将服务器i标记为已检查过
27         # 从服务器i开始，进行BFS
28         while(q):
29             # 弹出q队头的服务器x，考虑所有与其相连的服务器y
30             x = q.popleft()
31             # 对于服务器x，遍历所有其他服务器y，若y未检查过，且与x相连
32             for y in range(n):
33                 if x != y and checkList[y] == 0 and isConnected[x][y] == 1:
34                     q.append(y)      # 则把服务器y加入队列中
35                     checkList[y] = 1 # 同时把服务器y标记为已检查过
36             # 完成本次BFS，连通分量+1，即ans+1
37             ans += 1
38
39 print(ans)

```

## Java

```

1 import java.util.*;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         List<List<Integer>> isConnected = new ArrayList<>();
7
8         String[] firstLine = scanner.nextLine().split(" ");
9         List<Integer> firstRow = new ArrayList<>();
10        for (String s : firstLine) {
11            firstRow.add(Integer.parseInt(s));
12        }
13        isConnected.add(firstRow);
14
15        int n = firstRow.size();
16        for (int i = 1; i < n; i++) {
17            String[] line = scanner.nextLine().split(" ");
18            List<Integer> row = new ArrayList<>();
19            for (String s : line) {
20                row.add(Integer.parseInt(s));
21            }

```

```

22         isConnected.add(row);
23     }
24
25     int ans = 0;
26     int[] checkList = new int[n];
27
28     for (int i = 0; i < n; i++) {
29         if (checkList[i] == 0) {
30             Queue<Integer> q = new LinkedList<>();
31             q.add(i);
32             checkList[i] = 1;
33
34             while (!q.isEmpty()) {
35                 int x = q.poll();
36                 for (int y = 0; y < n; y++) {
37                     if (x != y && checkList[y] == 0 &&
isConnected.get(x).get(y) == 1) {
38                         q.add(y);
39                         checkList[y] = 1;
40                     }
41                 }
42             }
43             ans++;
44         }
45     }
46
47     System.out.println(ans);
48 }
49 }
50

```

## C++

```

1  #include <iostream>
2  #include <sstream>
3  #include <vector>
4  #include <queue>
5  using namespace std;
6
7  int main() {
8      vector<vector<int>>> isConnected;
9      string line;
10     getline(cin, line);
11     istringstream iss(line);
12     int val;

```

```

13     vector<int> firstRow;
14     while (iss >> val) {
15         firstRow.push_back(val);
16     }
17     isConnected.push_back(firstRow);
18
19     int n = firstRow.size();
20     for (int i = 1; i < n; i++) {
21         getline(cin, line);
22         istringstream iss(line);
23         vector<int> row;
24         while (iss >> val) {
25             row.push_back(val);
26         }
27         isConnected.push_back(row);
28     }
29
30     int ans = 0;
31     vector<int> checkList(n, 0);
32
33     for (int i = 0; i < n; i++) {
34         if (checkList[i] == 0) {
35             queue<int> q;
36             q.push(i);
37             checkList[i] = 1;
38
39             while (!q.empty()) {
40                 int x = q.front();
41                 q.pop();
42                 for (int y = 0; y < n; y++) {
43                     if (x != y && checkList[y] == 0 && isConnected[x][y] == 1)
44 {
45                         q.push(y);
46                         checkList[y] = 1;
47                     }
48                 }
49                 ans++;
50             }
51         }
52
53         cout << ans << endl;
54         return 0;
55     }
56

```

## 解法二：DFS

### Python

```
1 # 题目：2024E-广播服务器
2 # 分值：200
3 # 作者：闭着眼睛学数理化
4 # 算法：DFS
5 # 代码看不懂的地方，请直接在群上提问
6
7
8 # dfs递归函数
9 def dfs(x, isConnected, checkList):
10     # 对于传入的服务器x，将其标记为已检查过
11     checkList[x] = 1
12     # 遍历其他服务器y，若服务器y未检查过，且与服务器x相连
13     for y in range(n):
14         if y != x and checkList[y] == 0 and isConnected[x][y] == 1:
15             dfs(y, isConnected, checkList) # 则对y进行DFS
16
17 isConnected = list()
18 # 先输入第一行
19 isConnected.append(list(map(int, input().split())))
20 # 根据第一行的长度，得到n
21 n = len(isConnected[0])
22 # 输入剩余的n-1行
23 for _ in range(n-1):
24     isConnected.append(list(map(int, input().split())))
25
26 ans = 0
27 checkList = [0] * n # 构建检查数组checkList
28
29 # 遍历每一个服务器i
30 for i in range(n):
31     # 如果该服务器没检查过，则以服务器i作为起始点，进行DFS
32     if checkList[i] == 0:
33         # 进行DFS搜索
34         dfs(i, isConnected, checkList)
35         # 完成本次DFS，连通分量+1，即ans+1
36         ans += 1
37
38 print(ans)
```

# Java

```
1 import java.util.*;
2
3 public class Main {
4     static void dfs(int x, List<List<Integer>> isConnected, int[] checkList) {
5         checkList[x] = 1;
6         int n = isConnected.size();
7         for (int y = 0; y < n; y++) {
8             if (y != x && checkList[y] == 0 && isConnected.get(x).get(y) == 1)
9         {
10             dfs(y, isConnected, checkList);
11         }
12     }
13
14     public static void main(String[] args) {
15         Scanner scanner = new Scanner(System.in);
16         List<List<Integer>> isConnected = new ArrayList<>();
17         String[] firstLine = scanner.nextLine().split(" ");
18         List<Integer> firstRow = new ArrayList<>();
19         for (String s : firstLine) {
20             firstRow.add(Integer.parseInt(s));
21         }
22         isConnected.add(firstRow);
23
24         int n = firstRow.size();
25         for (int i = 1; i < n; i++) {
26             String[] line = scanner.nextLine().split(" ");
27             List<Integer> row = new ArrayList<>();
28             for (String s : line) {
29                 row.add(Integer.parseInt(s));
30             }
31             isConnected.add(row);
32         }
33
34         int ans = 0;
35         int[] checkList = new int[n];
36
37         for (int i = 0; i < n; i++) {
38             if (checkList[i] == 0) {
39                 dfs(i, isConnected, checkList);
40                 ans++;
41             }
42         }
43     }
44 }
```

```

42     }
43
44     System.out.println(ans);
45 }
46 }
47

```

## C++

```

1  #include <iostream>
2  #include <sstream>
3  #include <vector>
4  using namespace std;
5
6  void dfs(int x, vector<vector<int>>& isConnected, vector<int>& checkList) {
7      checkList[x] = 1;
8      int n = isConnected.size();
9      for (int y = 0; y < n; y++) {
10         if (y != x && checkList[y] == 0 && isConnected[x][y] == 1) {
11             dfs(y, isConnected, checkList);
12         }
13     }
14 }
15
16 int main() {
17     vector<vector<int>> isConnected;
18     string line;
19     getline(cin, line);
20     istringstream iss(line);
21     int val;
22     vector<int> firstRow;
23     while (iss >> val) {
24         firstRow.push_back(val);
25     }
26     isConnected.push_back(firstRow);
27
28     int n = firstRow.size();
29     for (int i = 1; i < n; i++) {
30         getline(cin, line);
31         istringstream iss(line);
32         vector<int> row;
33         while (iss >> val) {
34             row.push_back(val);
35         }
36         isConnected.push_back(row);

```



```

37     }
38
39     int ans = 0;
40     vector<int> checkList(n, 0);
41
42     for (int i = 0; i < n; i++) {
43         if (checkList[i] == 0) {
44             dfs(i, isConnected, checkList);
45             ans++;
46         }
47     }
48
49     cout << ans << endl;
50     return 0;
51 }
52

```

## 时空复杂度

时间复杂度： $O(N^2)$ 。需要遍历整个关联矩阵 `isConnected`。

空间复杂度： $O(N)$ 。

## 相同问题不同描述

### 2023B-快递业务站

#### 题目

快递业务范围有  $N$  个站点， $A$  站点与  $B$  站点可以中转快递，则认为  $A-B$  站可达。

如果  $A-B$  可达， $B-C$  可达，则  $A-C$  可达。

现在给  $N$  个站点编号  $0, 1, \dots, n-1$ ，用  $s[i][j]$  表示  $i-j$  是否可达。

$s[i][j] = 1$  表示  $i-j$  可达， $s[i][j] = 0$  表示  $i-j$  不可达。

现用二维数组给定  $N$  个站点的可达关系，请计算至少选择从几个主站点出发，才能可达所有站点（覆盖所有站点业务）。说明： $s[i][j]$  与  $s[j][i]$  取值相同。

## 输入描述

第一行输入为  $N$ ， $N$  表示站点个数。  $1 < N < 10000$

之后  $N$  行表示站点之间的可达关系，第  $i$  行第  $j$  个数值表示编号为  $i$  和  $j$  之间是否可达。

## 输出描述

输出站点个数，表示至少需要多少个主站点。

## 示例一

### 输入

```
1 4
2 1 1 1 1
3 1 1 1 0
4 1 1 1 0
5 1 0 0 1
```

### 输出

```
1 1
```

## 说明

选择 0 号站点作为主站点，0 站点可达其他所有站点，所以至少选择 1 个站点作为主站才能覆盖所有站点业务

## 示例二

### 输入

```
1 4
2 1 1 0 0
3 1 1 0 0
4 0 0 1 0
5 0 0 0 1
```

### 输出

## 说明

选择 0 号站点可以覆盖 0、1 站点，  
选择 2 号站点可以覆盖 2 号站点，  
选择 3 号站点可以覆盖 3 号站点，  
所以至少选择 3 个站点作为主站才能覆盖所有站点业务