# LC 547. Friend Circles

## Question

There are **N** students in a class. Some of them are friends, while some are not. Their friendship is transitive in nature. For example, if A is a **direct** friend of B, and B is a **direct** friend of C, then A is an **indirect** friend of C. And we defined a friend circle is a group of students who are direct or indirect friends.

Given a **N*N** matrix **M** representing the friend relationship between students in the class. If M[i][j] = 1, then the $i_{th}$ and $j_{th}$ students are **direct** friends with each other, otherwise not. And you have to output the total number of friend circles among all the students.

**Example 1:**

```
Input:
[[1,1,0],
 [1,1,0],
 [0,0,1]]
Output: 2
Explanation:The 0th and 1st students are direct friends, so they are in a friend circle.
The 2nd student himself is in a friend circle. So return 2.
```

**Example 2:**

```
Input:
[[1,1,0],
 [1,1,1],
 [0,1,1]]
Output: 1
Explanation:The 0th and 1st students are direct friends, the 1st and 2nd students are direct friends,
so the 0th and 2nd students are indirect friends. All of them are in the same friend circle, so return 1.
```

## Solution

```python
class Solution:
    def findCircleNum(self, M):
        """
        :type M: List[List[int]]
        :rtype: int
        """
        #Solution 3 - DFS
        if not M:
            return 0

        visited = set()
        counter, n = 0, len(M)
        for i in range(n):
            if i not in visited:
                self.dfs(M, i, visited)
```

```
                counter += 1
        return counter

    def dfs(self, M, i, visited):
        visited.add(i)
        for idx, val in enumerate(M[i]):
            if val == 1 and idx not in visited:
                self.dfs(M, idx, visited)

        #Solution
        #这道题可以用dfs做，也可以用并查集做，dfs的速度居然比并查集快是怎么回事?
        #dfs解法，遍历每一个结点，遍历到这个结点的时候，
        #会把它所有的直接朋友和间接朋友都遍历到，而且会将其加入到visited中，
        #所以如果遍历到的一个结点不在visited中，则证明它不在之前的朋友圈中，
        #可以开启一个新的朋友圈了，在对其进行dfs。
        def dfs(node):
            visited.add(node)
            for friend in range(len(M)):
                if M[node][friend] and friend not in visited:
                    dfs(friend)

        circle = 0
        visited = set()
        for node in range(len(M)):
            if node not in visited:
                dfs(node)
                circle += 1
        return circle


        #Solution 2 - 并查集
        #没懂。。。
        def find(x):
            if x != parents[x]:
                parents[x] = find(parents[x])
            return parents[x]
        def union(x, y):
            parents[find(x)] = find(y)
        n = len(M)
        parents = list(range(n))
        for i in range(len(M)):
            for j in range(len(M[i])):
                if M[i][j]:
                    union(i, j)
        circle = set(find(i) for i in range(n))
        return len(circle)
```