547. Friend Circles

April 1, 2017 | 43.6K views

Average Rating: 4.45 (29 votes)

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 di
so the 0th and 2nd students are indirect friends. All of them are in the same friend ci
```

Note:

1. N is in range [1,200].
2. M[i][i] = 1 for all students.
3. If M[i][j] = 1, then M[j][i] = 1.

Solution

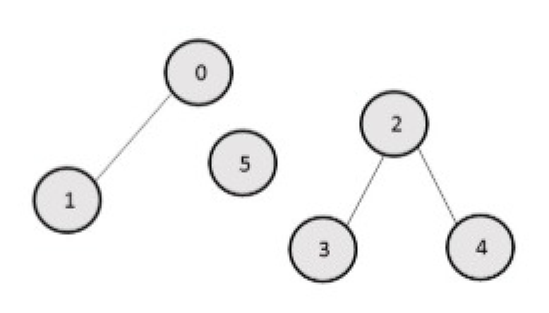
Approach #1 Using Depth First Search[Accepted]

Algorithm

The given matrix can be viewed as the Adjacency Matrix of a graph. By viewing the matrix in such a manner, our problem reduces to the problem of finding the number of connected components in an undirected graph. In order to understand the above statement, consider the example matrix below:

```
M= [1 1 0 0 0 0

    1 1 0 0 0 0

    0 0 1 1 1 0

    0 0 1 1 0 0

    0 0 1 0 1 0

    0 0 0 0 0 1]
```

If we view this matrix M as the adjancency matrix of a graph, the following graph is formed:

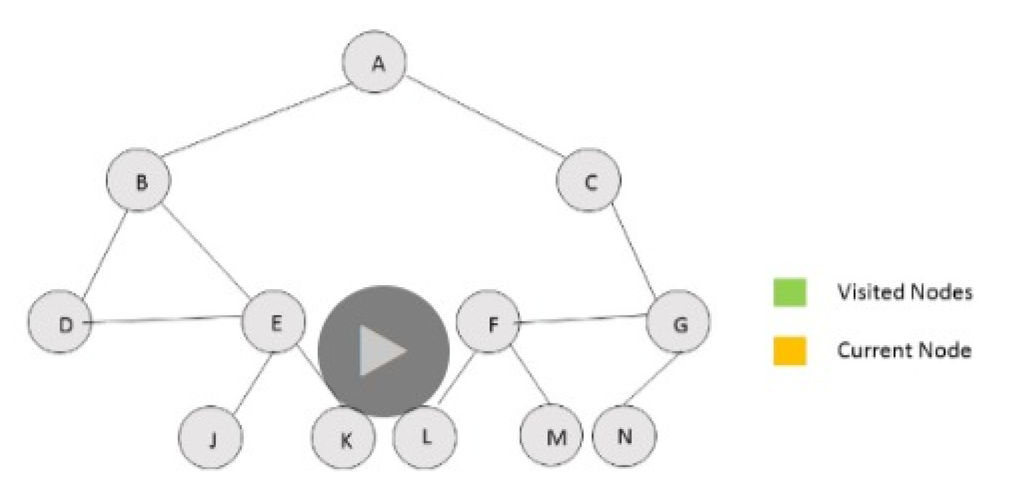


In this graph, the node numbers represent the indices in the matrix M and an edge exists between the nodes numbered i and j, if there is a 1 at the corresponding $M[i][j]$.

In order to find the number of connected components in an undirected graph, one of the simplest methods is to make use of Depth First Search starting from every node. We make use of $visited$ array of size $N(M$ is of size $NxN)$. This $visited[i]$ element is used to indicate that the i^{th} node has already been visited while undergoing a Depth First Search from some node.

To undergo DFS, we pick up a node and visit all its directly connected nodes. But, as soon as we visit any of those nodes, we recursively apply the same process to them as well. Thus, we try to go as deeper into the levels of the graph as possible starting from a current node first, leaving the other direct neighbour nodes to be visited later on.

The depth first search for an arbitrary graph is shown below:



1 / 14

From the graph, we can see that the components which are connected can be reached starting from any single node of the connected group. Thus, to find the number of connected components, we start from every node which isn't visited right now and apply DFS starting with it. We increment the $count$ of connected components for every new starting node.

Java

```java

public class Solution {
    public void dfs(int[][] M, int[] visited, int i) {
        for (int j = 0; j < M.length; j++) {
            if (M[i][j] == 1 && visited[j] == 0) {
                visited[j] = 1;
                dfs(M, visited, j);
            }
        }
    }
    public int findCircleNum(int[][] M) {
        int[] visited = new int[M.length];
        int count = 0;
        for (int i = 0; i < M.length; i++) {
            if (visited[i] == 0) {
                dfs(M, visited, i);
                count++;
            }
        }
        return count;
    }
}

```

Complexity Analysis

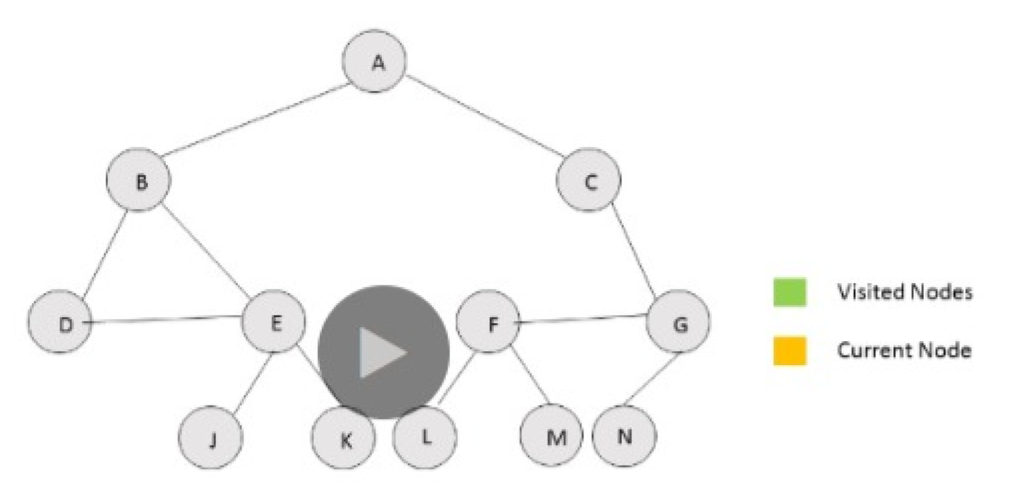
- Time complexity : $O(n^2)$. The complete matrix of size n^2 is traversed.
- Space complexity : $O(n)$. $visited$ array of size n is used.

Approach #2 Using Breadth First Search[Accepted]

Algorithm

As discussed in the above method, if we view the given matrix as an adjacency matrix of a graph, we can use graph algorithms easily to find the number of connected components. This approach makes use of Breadth First Search for a graph.

In case of Breadth First Search, we start from a particular node and visit all its directly connected nodes first. After all the direct neighbours have been visited, we apply the same process to the neighbour nodes as well. Thus, we exhaust the nodes of a graph on a level by level basis. An example of Breadth First Search is shown below:



1 / 13

In this case also, we apply BFS starting from one of the nodes. We make use of a $visited$ array to keep a track of the already visited nodes. We increment the $count$ of connected components whenever we need to start off with a new node as the root node for applying BFS which hasn't been already visited.

Java

```java
public class Solution {
    public int findCircleNum(int[][] M) {
        int[] visited = new int[M.length];
        int count = 0;
        Queue < Integer > queue = new LinkedList < > ();
        for (int i = 0; i < M.length; i++) {
            if (visited[i] == 0) {
                queue.add(i);
                while (!queue.isEmpty()) {
                    int s = queue.remove();
                    visited[s] = 1;
                    for (int j = 0; j < M.length; j++) {
                        if (M[s][j] == 1 && visited[j] == 0)
                            queue.add(j);
                    }
                }
                count++;
            }
        }
        return count;
    }
}
```

Complexity Analysis

- Time complexity : $O(n^2)$. The complete matrix of size n^2 is traversed.
- Space complexity : $O(n)$. A $queue$ and $visited$ array of size n is used.

Approach #3 Using Union-Find Method[Accepted]

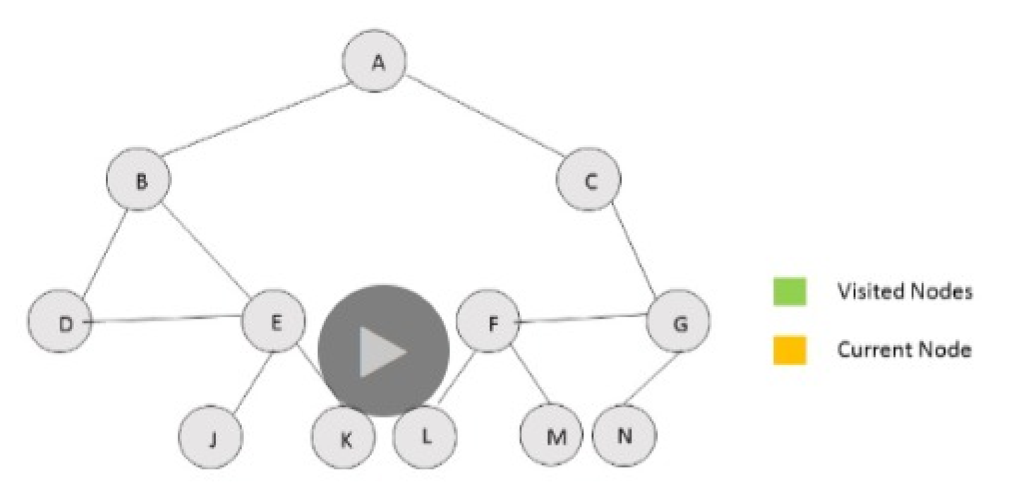
Algorithm

Another method that can be used to determine the number of connected components in a graph is the union find method. The method is simple.

We make use of a $parent$ array of size N. We traverse over all the nodes of the graph. For every node traversed, we traverse over all the nodes directly connected to it and assign them to a single group which is represented by their $parent$ node. This process is called forming a $union$. Every group has a single $parent$ node, whose own parent is given by -1.

For every new pair of nodes found, we look for the parents of both the nodes. If the parents nodes are the same, it indicates that they have already been united into the same group. If the parent nodes differ, it means they are yet to be united. Thus, for the pair of nodes (x, y), while forming the union, we assign $parent[parent[x]] = parent[y]$, which ultimately combines them into the same group.

The following animation depicts the process for a simple matrix:



1 / 14

At the end, we find the number of groups, or the number of parent nodes. Such nodes have their parents indicated by a -1. This gives us the required count.

Java

```java
1  public class Solution {
2      int find(int parent[], int i) {
3          if (parent[i] == -1)
4              return i;
5          return find(parent, parent[i]);
6      }
7  
8      void union(int parent[], int x, int y) {
9          int xset = find(parent, x);
10         int yset = find(parent, y);
11         if (xset != yset)
12             parent[xset] = yset;
13     }
14     public int findCircleNum(int[][] M) {
15         int[] parent = new int[M.length];
16         Arrays.fill(parent, -1);
17         for (int i = 0; i < M.length; i++) {
18             for (int j = 0; j < M.length; j++) {
19                 if (M[i][j] == 1 && i != j) {
20                     union(parent, i, j);
21                 }
22             }
23         }
24         int count = 0;
25         for (int i = 0; i < parent.length; i++) {
26             if (parent[i] == -1)
27                 count++;
28         }
```

Complexity Analysis

- Time complexity : $O(n^3)$. We traverse over the complete matrix once. Union and find operations take $O(n)$ time in the worst case.
- Space complexity : $O(n)$. $parent$ array of size n is used.

Rate this article: ★★★★★

Previous Next

Comments: 28 Sort By

Type comment here... (Markdown is supported)

Preview Post

ananlvjiao ★81 November 3, 2017 10:40 AM Report

I think for Union-Find approach, the complexity could be O(n^2) if path compression is used. Union-Find complexity with path compression is O(m), which m is operations (either union/find); in this case, since m[i][j] = m[j][i], the union operation is at most n*(n-1)/2, therefore it is O(n^2)

81 Share Reply

SHOW 3 REPLIES

spanlabs ★115 February 22, 2019 8:02 PM

If use `path compression` and `union by rank` to optimize union find solution, what time complexity will it be?
As big O of `union` operation is determined by `find` operation, and `find` has amortized time `O(a(n))` where a(n) < 5 in practical. So total time will be `O(n^2)` IMO.

16 Share Reply

SHOW 2 REPLIES

sean46 ★106 April 18, 2017 2:37 AM

For the union-find complexity analysis: If we traverse over the complete matrix O(n^2) and union/find operations take O(n), wouldn't the complexity be O(n^3)?

5 Share Reply

SHOW 2 REPLIES

aloginov ★196 June 25, 2019 7:20 AM

1. We can simplify/optimize code by using `boolean[] visited` instead of `int[] visited`

Read More

1 Share Reply

wangxlsb ★51 April 4, 2017 12:06 AM Report

Agree with tiny_code, time complexity should O(N^2).

1 Share Reply

vinod23 ★461 April 6, 2017 1:00 PM

Yes you are right. I have updated it. Thanks.

1 Share Reply

SHOW 1 REPLY

shchshhappy ★36 June 22, 2019 7:23 AM Report

change approach#3 line 19 from `if (M[i][j] == 1 && i != j)` to `if (M[i][j] == 1 && i < j)` saves 20ms(84ms->64ms) on Python3

0 Share Reply

laiyinlg ★140 January 25, 2019 5:57 AM

Simple Java solution:

```
class Solution {
    public int findCircleNum(int[][] M) {
        int[] result = new int[M.length];
```

Read More

0 Share Reply

alexishe ★236 October 5, 2018 8:38 AM

Can anyone explain Union-Find approach? I don't get why it works.

0 Share Reply

SHOW 3 REPLIES

vinod23 ★461 April 18, 2017 5:24 PM

I have updated the complexity of union-find approach.Thanks.

0 Share Reply

‹ 1 2 3 ›