**★★★☆☆**

**題組：Problem Set Archive** **with Online Judge**

**題號：315: Network**

**解題者：呂晉豪**

**解題日期：2024年5月2日**

**題意：**

給定一連通的無向圖，點的數量小於100，求出有幾個critical point，critical point是一個點，它的定義是：如果把這個點拿掉，其他點就不能連通。

**題意範例：**

輸入：

5 → 5個節點

5 1 2 3 4 → 5與1、5與2、5與3、5與4相連接

0 →第一筆測資結束

6 → 5個節點

2 1 3 → 2與1、2與3相連接

5 4 6 2 → 5與4、5與6、5與2相連接

0 →第二筆測資結束

0 →沒有測資了

輸出：

1 →第一筆測資有1個critical point

2 →第二筆測資有2個critical point

測資一：



測資二：

****

**解法：**

使用暴力法

1. 將圖建出來，這題我使用鄰接串列。

2. 用迴圈將每個點跑過一遍，每次把那一個點拿掉，再對剩下的隨便一個點執行BFS，看它們有沒有連通，如果有連通，代表它不是critical point；如果沒有連通，代表它是critical point。

3. 輸出critical point的數量。

**解法範例：**

原圖

拿掉1，其他點仍然連通

********

拿掉3，其他點仍然連通

拿掉2，其他點不連通

****

因此2、5是critical point

拿掉5，其他點不連通

拿掉6，其他點仍然連通

拿掉4，其他點仍然連通

**演算法分析:**

將每個點跑過一次需要O(n)的時間，對一個點做BFS需要O(|E|)，也就是O(n^2)的時間，因此，這個解法的時間複雜度是O(n^3)，因為n小於100，所以可以通過這題。

**討論：**

 這題因為測資很小，所以我用O(n^3)的演算法，那有更快的演算法嗎？ 有，如果能將圖先走訪一次，變為樹，變為樹後，每個節點有兩種可能：要嘛是根節點，要嘛不是根節點。 如果是根節點，並且它只有一棵子樹，那它不是critical point；如果它有多個子樹，那它就是critical point。 如果不是根節點，並且祖先到任何一棵子樹都有替代道路，則它不是critical point；如果祖先到其中一棵子樹沒有替代道路，則它是critical point。

**解法範例：**

5是根，它有多棵子樹，所以它是critical point；2不是根，它的祖先(5)到子節點(1、3)沒有替代道路，所以它是critical point。

****

**演算法分析:**

這種解法的時間複雜度是把圖搜尋一次的時間複雜度，為O(|E|)，也就是O(n^2)，比暴力法好。

**程式：**

#include <bits/stdc++.h>

using namespace std;

int bfs(vector <int> adj[] , int r , int x){

 int visited[101] = {};

 visited[x] = 1;

 queue <int> q;

 q.push(r);

 visited[r] = 1;

 int cnt = 1;

 while(!q.empty()){

 int v = q.front();

 q.pop();

 for(auto it = adj[v].begin() ; it != adj[v].end() ; it++){

 if(!visited[\*it]){

 visited[\*it] = 1;

 cnt++;

 q.push(\*it);

 }

 }

 }

 return cnt;

}

void func(int n){

 string str;

 stringstream ss;

 vector <int> adj[101];

 if(n == 1){

 cout << 0 << endl;

 } else {

 //graph

 int u , v;

 while(getline(cin , str)){

 if(str.length() == 1){

 break;

 }

 ss.str("");

 ss.clear();

 ss << str;

 ss >> u;

 while(ss >> v){

 adj[u].push\_back(v);

 adj[v].push\_back(u);

 }

 }

 int cnt = 0;

 for(int i = 1 ; i <= n ; i++){

 int num = bfs(adj , i % n + 1 , i);

 if(num != n - 1) cnt++;

 }

 cout << cnt << endl;

 }

}

int main()

{

 int n;

 while(cin >> n){

 if(n == 0) break;

 func(n);

 }

 return 0;

}