Practice Compete Jobs Rank Leaderboard gadhiya

Dashboard > Algorithms > Graph Theory > Even Tree

Badge Progress ★ ★ ★ ★

Points: 4727.88 Rank: 491

Even Tree

by HackerRank

Problem Submissions Leaderboard Discussions Editorial

You are given a tree (a simple connected graph with no cycles). The tree has N nodes numbered from 1 to N and is rooted at node 1.

Find the maximum number of edges you can remove from the tree to get a forest such that each connected component of the forest contains an even number of nodes.

Input Format

The first line of input contains two integers N and M. N is the number of nodes, and M is the number of edges.
The next M lines contain two integers u_i and v_i which specifies an edge of the tree.

Constraints

- $2 \leq N \leq 100$

Note: The tree in the input will be such that it can always be decomposed into components containing an even number of nodes.

Output Format

Print the number of removed edges.

Sample Input

```
10 9
2 1
3 1
4 3
5 2
6 1
7 2
8 6
9 8
10 8
```

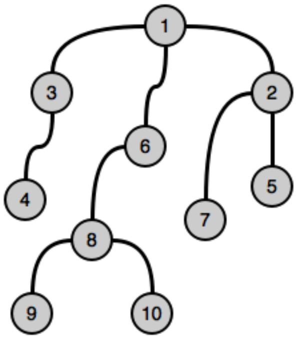
Sample Output

```
2
```

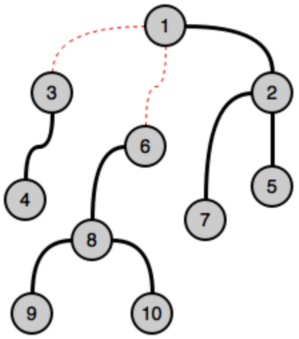
Explanation

On removing edges $(1, 3)$ and $(1, 6)$, we can get the desired result.

Original tree:



Decomposed tree:



Submissions: 22680
Max Score: 50
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

More

Current Buffer (saved locally, editable)

Java 8

```
import java.io.*;
import java.util.*;

class Graph{

     private int V;
     private LinkedList<Integer> adj[];
     private Map<Integer,Integer> mapCNT;

    public Graph(int V){
        this.V = V;
        adj = new LinkedList[V];
        mapCNT = new HashMap<Integer,Integer>();

        for(int i = 0 ; i < V ; i++){
            adj[i] = new LinkedList<Integer>();
        }

    }

    public void addEdge(int destinationVertex, int sourceVertex){
        adj[sourceVertex - 1].add(destinationVertex - 1);
    }

    public Map<Integer,Integer> getMAPCNT(){
        return mapCNT;
    }

    public Map<String, Integer> doDFS(int startingNode, int callingFrom, Map<String,Integer> map){

```

```java
        Iterator<Integer> i = adj[startingNode].listIterator();

        int cnt = 0;

        if(i.hasNext()){
            while (i.hasNext())
            {
                int n = i.next();
                cnt++;
                doDFS(n,startingNode,map);

                if(n != startingNode){

                    int newCNT = mapCNT.get(n);

                    if(mapCNT.containsKey(startingNode)){
                        int oldCNT = mapCNT.get(startingNode);
                        mapCNT.put(startingNode,(oldCNT + newCNT));
                    }
                    else{
                        mapCNT.put(startingNode,newCNT + 1);
                    }

                }

            }
        }
        else{
            mapCNT.put(startingNode,1);
        }

        if(startingNode != callingFrom){
            map.put(callingFrom + "-" + startingNode,mapCNT.get(startingNode));
        }

        return map;
    }
}

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);
        int V = scan.nextInt();
        int E = scan.nextInt();

        Graph graph = new Graph(V);

        for(int i = 0 ; i < E ; i++){

            int dest = scan.nextInt();
            int src = scan.nextInt();
            graph.addEdge(dest,src);
        }

        Map<String,Integer> map = graph.doDFS(0,0,new HashMap<String,Integer>());

        int output = 0;

        for(int i : map.values()){

            if(i % 2 == 0){
                output++;
            }

        }

        System.out.println(output);
    }
}
```

Line: 1 Col: 1

Upload Code as File □ Test against custom input

Run Code Submit Code

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature