Experiment No. 7

Title: Implementation of Spanning Trees

Batch:A2   **Roll No: 16010421059**   **Experiment No.:3**
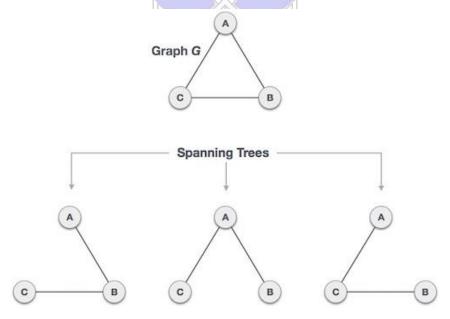
Aim: To study Spanning Tree for implementation of problem statement that is based on Minimum spanning tree and verify given test cases.

Resources needed: Text Editor, C/C++ IDE

Theory:

A spanning tree is a subset of Graph G, which has all the vertices covered with minimum possible number of edges. Hence, a spanning tree does not have cycles and it cannot be disconnected.

By this definition, we can draw a conclusion that every connected and undirected Graph G has at least one spanning tree. A disconnected graph does not have any spanning tree, as it cannot be spanned to all its vertices.

We found three spanning trees off one complete graph. A complete undirected graph can have maximum n^{n-2} number of spanning trees, where **n** is the number of nodes. In the above addressed example, **n is 3,** hence $3^{3-2} = 3$ spanning trees are possible.

General Properties of Spanning Tree

We now understand that one graph can have more than one spanning tree. Following are a few properties of the spanning tree connected to graph G −

- A connected graph G can have more than one spanning tree.

- All possible spanning trees of graph G, have the same number of edges and vertices.

- The spanning tree does not have any cycle (loops).
- Removing one edge from the spanning tree will make the graph disconnected, i.e. the spanning tree is **minimally connected**.
- Adding one edge to the spanning tree will create a circuit or loop, i.e. the spanning tree is **maximally acyclic**.

Mathematical Properties of Spanning Tree

- Spanning tree has **n-1** edges, where **n** is the number of nodes (vertices).
- From a complete graph, by removing maximum **e - n + 1** edges, we can construct a spanning tree.
- A complete graph can have maximum n^{n-2} number of spanning trees.

Thus, we can conclude that spanning trees are a subset of connected Graph G and disconnected graphs do not have spanning tree.

Application of Spanning Tree

Spanning tree is basically used to find a minimum path to connect all nodes in a graph. Common application of spanning trees are −

- Civil Network Planning
- Computer Network Routing Protocol
- Cluster Analysis

Let us understand this through a small example. Consider, city network as a huge graph and now plans to deploy telephone lines in such a way that in minimum lines we can connect to all city nodes. This is where the spanning tree comes into picture.

Minimum Spanning Tree (MST)

In a weighted graph, a minimum spanning tree is a spanning tree that has minimum weight than all other spanning trees of the same graph. In real-world situations, this weight can be measured as distance, congestion, traffic load or any arbitrary value denoted to the edges.

Minimum Spanning-Tree Algorithm

- Kruskal's Algorithm
- Prim's Algorithm

Both are greedy algorithms.

Activity:

Consider the following problem statement and other information provided along with it:

Problem Statement: Friendless Dr. Sheldon Cooper

Leonard has decided to quit living with Dr. Sheldon Cooper and has started to live with Penny. Yes, you read it right. (And you read it here for the first time!) He is fed up of Sheldon, after all. Since, Sheldon no more has Leonard to drive him all around the city for various things, he's feeling a lot uneasy so he decides to set up a network of drivers all around the city to drive him to various places.

But, not every driver wants to go every place in the city for various personal reasons, so Sheldon needs to trust many different cab drivers. (This is a very serious issue for him, by the way) The problem occurs mainly when Sheldon needs to go to - for example, the Comic book store - and there's no cab driver who goes directly to that place. So, he has to take a cab till another place, and then take a cab from there - making him more scared!

Sheldon wants to limit his trust issues. Really once and for all

Let's say that you're given the schedule of all the cabs from the major points where he travels to and from - can you help Sheldon figure out the least number of cab drivers he needs to trust, in order to go to all the places he wants to?

Input format

The first line contains a number with the number of test cases. Every test case has the following input:

- Two integers a, b.	a - number of places he needs to go. b - Number of cab drivers.

Output format

Print the minimum number of cab drivers he needs to have faith in to travel between places in the city.
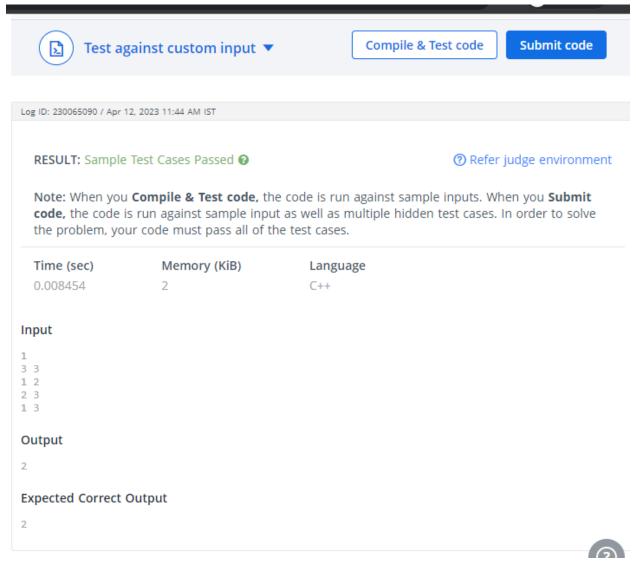
Constraints:
1<=t<=100
2<=a<=1000 | 1<=b<=1000
m NOT equal to n | 1<=m | n<=b
The graph is connected.

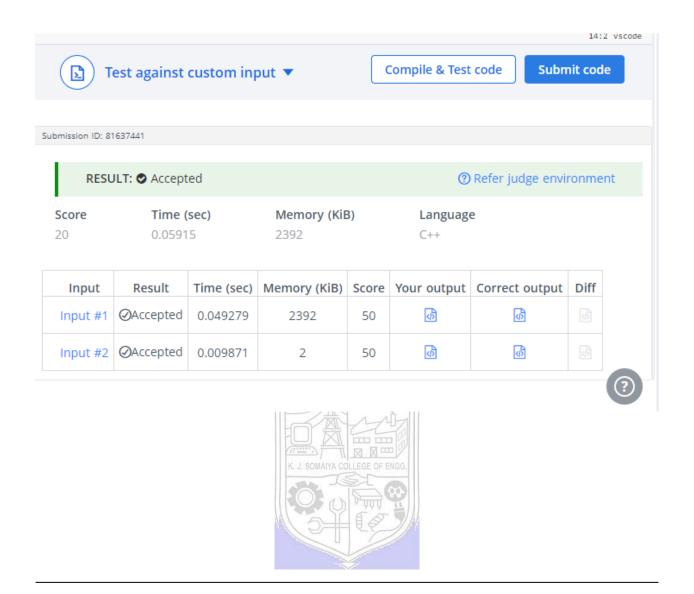| Sample Input	Sample Output
1	2
3 3	
1 2	
2 3	
1 3	

Program:

```cpp
#include<iostream>
#include<bits/stdc++.h>
using namespace std;
vector<int> adj[999];
bool visited[999];
void dfs(int s)
{
    visited[s]=true;
    for(int i=0;i<adj[s].size();i++)
    {
        if(visited[adj[s][i]]==false)
            dfs(adj[s][i]);
    }
}
int main()
{
    int t; cin>>t;
    while(t--)
    {
        int a,b,x,y; cin>>a>>b;
        int count=0;
        for(int i=1;i<=b;i++)
        {
            cin>>x>>y;
            adj[x].push_back(y);
            adj[y].push_back(x);
        }
        for(int i=1;i<=a;i++)
        {
            if(visited[i]==false)
            {
                dfs(visited[i]);
                count++;
            }
        }
        cout<<(count-1)<<endl;

    }

}
```

Output:

Test Result:

Outcomes:

  The outcome of this experiment is- CO3 . Understand the Graphs, related algorithms, efficient implementation of those algorithms and applications.

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)

We studied the spanning tree for implementation of problem statement that is based on minimum spanning tree to verify given test cases.

References:

  1. https://www.hackerearth.com/practice/algorithms/graphs/minimum-spanning-

tree/practice-problems/algorithm/friendless-dr-sheldon-cooper-14/

2. T.H. Coreman ,C.E. Leiserson,R.L. Rivest, and C. Stein, " Introduction to algorithms", 3rd Edition 2009, Prentice Hall India Publication
3. Antti Laaksonen, "Guide to Competitive Programming",Springer,2018
4. Gayle Laakmann McDowell," Cracking the Coding Interview",CareerCup LLC,2015
5. Steven S. Skiena Miguel A. Revilla,"Programming challenges, The Programming Contest Training Manual", Springer, 2006
6. Antti Laaksonen, "Competitive Programmer's Handbook", Hand book, 2018
7. Steven Halim and Felix Halim, "Competitive Programming 3: The Lower Bounds of Programming Contests", Handbook for ACM ICPC