MONK AND ISLANDS

**\*Problem:**

The monk visited the land of the Islands. There are a total of N islands numbered from 1 to N . Several pairs of islands are connected by two-way bridges run on water. Đat hates going over these bridges because they require a lot of effort. He is stands on Island No. 1 and wants to go to Island No. Find the minimum number of bridges that he will must go through, if he takes the optimal path. Assume that there is always a path from vertex 1 to n.

* **Input:** The first line contains T . Number of test tests . The first line of each test set contains two integers N , M. Each of the M lines contains two integers X and Y , indicating that there is a bridge between Island X and Island Y.
* **Output:** Print the answer for each test case in a new line.

Example:

|  |  |
| --- | --- |
| **INPUT**  2  3 2  1 2  2 3  4 4  1 2  2 3  3 4  4 2 | **OUTPUT**  2  2 |

**\*Analysis:**

We use BFS algorithm because here the requirement is to find the shortest path between 1 to N but if using Dijktra, it is too complicated and here if using Dijktra, we have to assign weight to each edge is 1, and if using BFS we won't need.

+We use array di] to store the number of vertices from vertex 1 to node I with d[1]=1, so the final output will be d[n]-1, we have -1 because the number of edges is equal to the number of vertices minus 1.

void BFS(vector < int > ke[1000],int n,int m,int d[1000])

{

queue < int > qu;

qu.push(1);

d[1]=1;

while (!qu.empty())

{

int u=qu.front();

qu.pop();

for(int i=0;i<ke[u].size();i++)

{

int v=ke[u][i];

if(d[v]==0)

{

d[v]=d[u]+1;

qu.push(v);

}

if(v==n)

return;

}

}

}

int main()

{

int q;

cin>>q;

for(int i=0;i<q;i++)

{

int m,n,d[1000];

bool F[1000];

vector < int > ke[1000];

memset(d, 0, sizeof(d));

cin>>n>>m;

for(int j=0;j<m;j++)

{

int x,y;

cin>>x>>y;

/\*We use vector to store the adjacent edge of vertex i as j and vice versa\*/

ke[x].push\_back(y);

ke[y].push\_back(x);

}

BFS(ke,n,m,d);

cout<<d[n]-1<<endl;

}

return 0;

}