**Level of Nodes: -**

**Medium Accuracy: 50.95% Submissions: 30K+ Points: 4**

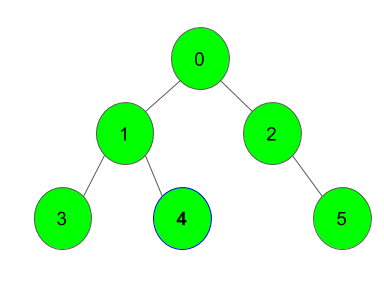
Given an integer **X** and an undirected acyclic graph with **V nodes**, labeled from **0**to **V-1**, and **E**edges, find the **level**of node labeled as **X**.

**Level**is the **minimum**number of edges you must travel from the node 0 to some target.

If there doesn't exists such a node that is labeled as **X**, **return -1**.

**Example 1:**

**Input:**

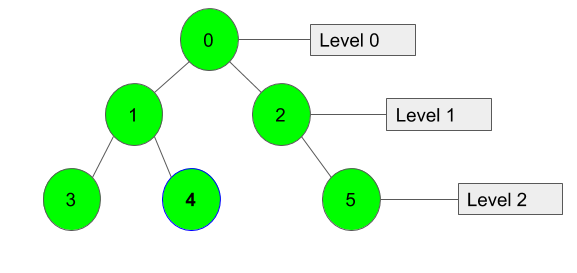


**X** = 4

**Output:**

2

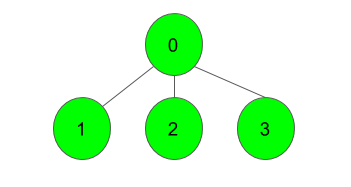
**Explanation**:



We can clearly see that Node 4 lies at Level 2.

**Example 2:**

**Input:**



**X** = 1

**Output:**

1

**Explanation**:

Node 1 lies at level 1, immediate after Node 0.

**Your task:**  
You dont need to read input or print anything. Your task is to complete the function **nodeLevel()** which takes two integers **V** and **X** denoting the number of vertices and the node, and another adjacency list **adj**as input parameters and returns the level of Node **X**. If node **X** isn't present it returns **-1**.

**Expected Time Complexity:**O(V)  
**Expected Auxiliary Space:**O(V)

**Constraints:**  
2 ≤ V ≤ 1041 ≤ E ≤ 104  
0 ≤ adji, j < V  
1 ≤ X < V  
Graph doesn't contain multiple edges and self loops.

**Code: -**

//{ Driver Code Starts

#include<bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution

{

public:

//Function to find the level of node X.

int nodeLevel(int V, vector<int> adj[], int X)

{

// code here

vector<bool> vis(V+1, false);

int level = 0;

queue<int> q;

q.push(0);

vis[q.front()] = true;

while(q.size()){

++level;

int count = q.size();

while(count--){

int front = q.front();

q.pop();

for(auto &i : adj[front]){

if(!vis[i]){

vis[i] = true;

if(i == X) return level;

q.push(i);

}

}

}

}

return -1;

}

};

//{ Driver Code Starts.

int main()

{

int t;

cin >> t;

while(t--)

{

int V, E, X;

cin >> V >> E;

vector<int> adj[V];

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

{

int u, v;

cin >> u >> v;

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

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

}

cin>>X;

Solution obj;

cout << obj.nodeLevel(V, adj, X) << "\n";

}

return 0;

}

// } Driver Code Ends

**T.C: - O(V)**

**S.C: - O(V)**