CHUYỂN DANH SÁCH CẠNH SANG DANH SÁCH KỀ

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09001/top)

Cho đồ thị vô hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy viết chương trình thực hiện chuyển đổi biểu diễn đồ thị dưới dạng danh sách kề.

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm |E| +1 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh và số cạnh của đồ thị; |E| dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤200; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra danh sách kề của các đỉnh tương ứng theo khuôn dạng của ví dụ dưới đây. Các đỉnh trong danh sách in ra theo thứ tự tăng dần.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  6  9  1  2  1  3  2  3  2  5  3  4  3  5  4  5  4  6  5  6 | 1: 2 3  2: 1 3 5  3: 1 2 4 5  4: 3 5 6  5: 2 3 4 6  6: 4 5 |

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

main(){

int t;

cin >> t;

while(t--){

vector<ll> ke[1001];

ll V, E, u, v;

cin >> V >> E;

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

cin >> u >> v;

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

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

}

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

sort(ke[i].begin(), ke[i].end());

}

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

cout << i << ": ";

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

cout << ke[i][j] << " ";

}

cout << endl;

}

}

}

Cho đơn đồ thị G vô hướng liên thông được mô tả bởi danh sách kề. Hãy in ra danh sách cạnh tương ứng của G.

**Input**

* Dòng đầu tiên ghi số N là số đỉnh (1<N<50)
* N dòng tiếp theo mỗi dòng ghi 1 danh sách kề lần lượt theo thứ tự từ đỉnh 1 đến đỉnh N

**Output:**Ghi ra lần lượt từng cạnh của đồ thị theo thứ tự tăng dần.

**Ví dụ**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  2 3  1 3  1 2 | 1 2  1 3  2 3 |

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

main(){

int n;

cin >> n;

cin.ignore();

int a[51][51];

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

string s;

getline(cin,s);

for(int i = 0; i < s.size(); i++){

int u = 0;

while(i < s.size() && s[i] != ' '){

u = u\*10 + (s[i]-'0'); //TH dinh >= 2 cs

i++;

}

a[t][u] = 1;

a[u][t] = 1;

}

}

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

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

if(a[i][j] == 1){

cout << i << " " << j << endl;

a[i][j] = a[j][i] = 0;

}

}

}

}

 BIỂU DIỄN ĐỒ THỊ CÓ HƯỚNG.

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09003/top)

Cho đồ thị có hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy viết chương trình thực hiện chuyển đổi biểu diễn đồ thị dưới dạng danh sách kề.

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm |E| +1 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh và số cạnh của đồ thị; |E| dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤200; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra danh sách kề của các đỉnh tương ứng theo khuôn dạng của ví dụ dưới đây. Các đỉnh trong danh sách in ra theo thứ tự tăng dần.

**Ví dụ:**

|  |  |
| --- | --- |
| Input: | Output: |
| 1  6  9  1  2  2  5  3  1  3  2  3  5  4  3  5  4  5  6  6  4 | 1: 2  2: 5  3: 1 2 5  4: 3  5: 4 6  6: 4 |

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

main(){

int t;

cin >> t;

while(t--){

vector<ll> ke[1001];

ll V, E, u, v;

cin >> V >> E;

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

cin >> u >> v;

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

}

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

sort(ke[i].begin(), ke[i].end());

}

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

cout << i << ": ";

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

cout << ke[i][j] << " ";

}

cout << endl;

}

}

}

DFS TRÊN ĐỒ THỊ VÔ HƯỚNG

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09004/top)

Cho đồ thị vô hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy viết thuật toán duyệt theo chiều sâu bắt đầu tại đỉnh uÎV (DFS(u)=?)

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm |E| +1 dòng: dòng đầu tiên đưa vào ba số |V|, |E| tương ứng với số đỉnh và số cạnh của đồ thị, và u là đỉnh xuất phát; |E| dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤200; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra danh sách các đỉnh được duyệt theo thuật toán DFS(u) của mỗi test theo khuôn dạng của ví dụ dưới đây.

**Ví dụ:**

|  |  |
| --- | --- |
| Input: | Output: |
| 1  6 9 5  1 2  1 3  2 3  2 4  3 4  3 5  4 5  4 6  5 6 | 5 3 1 2 4 6 |

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

ll vs[1001], e[1001];

vector<int> ke[1001];

void DFS(int u){

cout << u << " ";

vs[u] = 1;

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

if(vs[ke[u][i]] == 0){

vs[ke[u][i]] = 1;

DFS(ke[u][i]);

}

}

}

main(){

int t;

cin >> t;

while(t--){

memset(ke,0,sizeof(ke));

ll V, E, u, v, x;

cin >> V >> E >> x;

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

cin >> u >> v;

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

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

}

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

sort(ke[i].begin(), ke[i].end());

}

memset(vs,0,sizeof(vs));

DFS(x);

cout << endl;

}

}

BFS TRÊN ĐỒ THỊ VÔ HƯỚNG

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09005/top)

Cho đồ thị vô hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy viết thuật toán duyệt theo chiều rộng bắt đầu tại đỉnh uÎV (BFS(u)=?)

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào ba số |V|, |E|, uÎV tương ứng với số đỉnh, số cạnh và đỉnh bắt đầu duyệt; Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤200; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra danh sách các đỉnh được duyệt theo thuật toán BFS(u) của mỗi test theo khuôn dạng của ví dụ dưới đây.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  6 9 1  1 2 1 3 2 3 2 5 3 4 3 5 4 5 4 6 5 6 | 1 2 3 5 4 6 |

#include<bits/stdc++.h>

using namespace std;

int A[1005][1005], n, m ,u;

bool used[1005];

void init(){

cin >> n >> m >> u;

memset(A, 0, sizeof(A));

memset(used, false, sizeof(used));

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

int x, y; cin >> x >> y;

A[x][y] = 1;

A[y][x] = 1;

}

}

void BFS(){

queue<int> Q;

Q.push(u); used[u] = true;

while(!Q.empty()){

int s = Q.front();

cout << s << " ";

Q.pop();

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

if (!used[t] && A[s][t]){

Q.push(t);

used[t] = true;

}

}

}

}

int main(){

int t; cin >> t;

while (t--){

init();

BFS();

cout << endl;

}

}

LIỆT KÊ CẠNH CẦU

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09013/top)

Cho đồ thị vô hướng liên thông G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy đưa ra tất cả các cạnh cầu của đồ thị?

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm |E| + 1 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh và số cạnh; |E| dòng tiếp theo đưa vào các bộ đôi u, v tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra danh sách các cạch cầu của mỗi test theo từng dòng. In ra đáp án theo thứ tự từ điển, theo dạng “a b …” với a < b.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  5 5  1 2  1 3  2 3  2 5  3 4 | 2 5 3 4 |

#include<bits/stdc++.h>

using namespace std;

int n, m;

bool used[1005];

vector<int> vt[1005];

vector<pair<int,int> > ans;

void Reinit(){

for (int i=0; i<1005; i++)

vt[i].clear();

memset(used, false, sizeof(used));

}

void BFS(int u){

queue<int> Q;

Q.push(u);

used[u] = true;

while(!Q.empty()){

int s = Q.front();

Q.pop();

for (int t=0; t<vt[s].size(); t++){

if (!used[vt[s][t]]){

Q.push(vt[s][t]);

used[vt[s][t]] = true;

}

}

}

}

int TPLT(){

int res = 0;

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

if (!used[i]){

res++;

BFS(i);

}

}

return res;

}

void Canhcau(int c, int res, vector<pair<int,int> > tmp){

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

if (i != c){

vt[tmp[i].first].push\_back(tmp[i].second);

vt[tmp[i].second].push\_back(tmp[i].first);

}

}

int d = TPLT();

if (d > res){

if(tmp[c].first < tmp[c].second)

ans.push\_back({tmp[c].first, tmp[c].second});

else

ans.push\_back({tmp[c].second, tmp[c].first});

}

}

int main(){

int T; cin >> T;

while (T--){

Reinit();

ans.clear();

cin >> n >> m;

vector<pair<int,int> > tmp;

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

int x, y; cin >> x >> y;

tmp.push\_back({x, y});

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

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

}

int res = TPLT();

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

Reinit();

Canhcau(i, res, tmp);

}

sort(ans.begin(), ans.end());

for (int i=0; i<ans.size(); i++)

cout << ans[i].first << " " << ans[i].second << " ";

cout << endl;

}

}

ĐƯỜNG ĐI VÀ CHU TRÌNH EULER VỚI ĐỒ THỊ VÔ HƯỚNG

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA10004/top)

Cho đồ thị vô hướng liên thông G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy kiểm tra xem đồ thị có đường đi Euler hay chu trình Euler hay không?

Đường đi Euler bắt đầu tại một đỉnh, và kết thúc tại một đỉnh khác.

Chu trình Euler bắt đầu tại một đỉnh, và kết thúc chu trình tại chính đỉnh đó.

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh,  số cạnh của đồ thị; Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra 1, 2, 0 kết quả mỗi test theo từng dòng tương ứng với đồ thị có đường đi Euler, chu trình Euler và trường hợp không tồn tại.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 2  6  10  1 2 1 3 2 3 2 4 2 5 3 4 3 5 4 5 4 6 5 6  6 9  1  2 1  3 2  3 2  4 2  5 3  4 3  5 4  5 4  6 | 2  1 |

#include<bits/stdc++.h>

using namespace std;

int n, m;

vector<int> adj[1005];

int isEulerian(){

int odd = 0;

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

if (adj[i].size()%2 != 0)

odd++;

}

if (odd > 2)

return 0;

return (odd == 2)? 1 : 2;

}

int main(){

int T; cin >> T;

while (T--){

cin >> n >> m;

for (int i=0; i<1005; i++)

adj[i].clear();

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

int x, y; cin >> x >> y;

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

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

}

cout << isEulerian() << endl;

}

}

CHU TRÌNH EULER TRONG ĐỒ THỊ CÓ HƯỚNG

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA10005/top)

Cho đồ thị có hướng liên thông yếu G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy kiểm tra xem đồ thị có chu trình Euler hay không?

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh,  số cạnh của đồ thị; Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra 1, 0 kết quả mỗi test theo từng dòng tương ứng với đồ thị có chu trình Euler và trường hợp không tồn tại đáp án.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 2  6  10  1 2 2 4 2 5 3 1 3 2 4 3 4 5 5 3 5 6 6 4  3 3  1 2 2 3 1 3 | 1  0 |

#include<bits/stdc++.h>

using namespace std;

int n, m;

vector<int> adj[1005];

bool isEulerCycle(){

vector<int> start(1005, 0), end(1005, 0);

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

int sum = 0;

for (int j=0; j<adj[i].size(); j++){

start[adj[i][j]]++;

sum++;

}

end[i] = sum;

}

if (start == end)

return true;

return false;

}

int main(){

int T; cin >> T;

while (T--){

cin >> n >> m;

for (int i=0; i<1005; i++)

adj[i].clear();

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

int x, y; cin >> x >> y;

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

}

if (isEulerCycle()) cout << 1;

else cout << 0;

cout << endl;

}

}

CHUYỂN MA TRẬN KỀ SANG DANH SÁCH KỀ

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09021/top)

Ma trận kề A của một đồ thị vô hướng là một ma trận chỉ có các số 0 hoặc 1 trong đó A[i][j] = 1 có ý nghĩa là đỉnh i kề với đỉnh j (chỉ số tính từ 1).

Danh sách kề thì liệt kê các đỉnh kề với đỉnh đó theo thứ tự tăng dần.

Hãy chuyển biểu diễn đồ thị từ dạng ma trận kề sang dạng danh sách kề.

**Input:** Dòng đầu tiên chứa số nguyên n – số đỉnh của đồ thị (1 < n ≤ 1000). n dòng tiếp theo, mỗi dòng có n số nguyên có giá trị 0 và 1 mô tả ma trận kề của đồ thị.

**Output:** Gồm n dòng, dòng thứ i chứa các số nguyên là đỉnh có nối với đỉnh i và được sắp xếp tăng dần. Dữ liệu đảm bảo mỗi đỉnh có kết nối với ít nhất 1 đỉnh khác.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  0 1 1  1 0 1  1 1 0 | 2 3  1 3  1 2 |

#include<bits/stdc++.h>

using namespace std;

int matrix[1005][1005];

int main(){

vector<int> res;

int n; cin >> n;

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

for (int j = 1; j <= n; j++)

cin >> matrix[i][j];

}

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

res.clear();

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

if (matrix[i][j])

res.push\_back(j);

}

for (int k = 0; k < res.size(); k++)

cout << res[k] << " ";

cout << endl;

}

}

CHUYỂN DANH SÁCH KỀ SANG MA TRẬN KỀ

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09020/top)

Cho đơn đồ thị vô hướng có n đỉnh dưới dạng danh sách kề.

Hãy biểu diễn đồ thị bằng ma trận kề.

**Input:** Dòng đầu tiên chứa số nguyên n – số đỉnh của đồ thị (1 ≤  n ≤  1000). n dòng tiếp theo, dòng thứ i chứa các số nguyên là các đỉnh kề với đỉnh i.

**Output:** Ma trận kề của đồ thị.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  2 3  1 3  1 2 | 0 1 1  1 0 1  1 1 0 |

#include<bits/stdc++.h>

using namespace std;

int n;

int matrix[1005][1005];

int strToNum(string s){

int res=0;

for (int i=0; i<s.length(); i++)

res = res\*10+(s[i]-'0');

return res;

}

int main(){

int n; cin >> n;

cin.ignore();

memset(matrix, 0 , sizeof(matrix));

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

string s; getline(cin, s);

stringstream ss;

ss<<s;

while (ss>>s){

int tmp=strToNum(s);

matrix[i][tmp] = 1;

}

}

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

for (int j=1; j<=n; j++)

cout<<matrix[i][j]<<" ";

cout << endl;

}

}

DFS TRÊN ĐỒ THỊ CÓ HƯỚNG

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09022/top)

Cho đồ thị có hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy viết thuật toán duyệt theo chiều sâu bắt đầu tại đỉnh uÎV (DFS(u)=?)

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào ba số |V|, |E|, uÎV tương ứng với số đỉnh, số cạnh và đỉnh bắt đầu duyệt; Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤200; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra danh sách các đỉnh được duyệt theo thuật toán DFS(u) của mỗi test theo khuôn dạng của ví dụ dưới đây.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  6 9 5  1 2 2 5 3 1 3 2 3 5 4 3 5 4 5 6 6 3 | 5 4 3 1 2 6 |

#include<bits/stdc++.h>

int A[1005][1005], n, m, u;

bool used[1005];

using namespace std;

void init(){

cin >> n >> m >> u;

memset(A, 0, sizeof(A));

memset(used, false, sizeof(used));

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

int x, y; cin >> x >> y;

A[x][y] = 1;

}

}

void DFS(){

stack<int> St;

St.push(u); used[u] = true;

cout << u << " ";

while(!St.empty()){

int s = St.top();

St.pop();

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

if (!used[t] && A[s][t]){

cout << t << " ";

used[t] = true;

St.push(s);

St.push(t);

break;

}

}

}

}

int main(){

int t; cin >> t;

while (t--){

init();

DFS();

cout << endl;

}

}

BFS TRÊN ĐỒ THỊ CÓ HƯỚNG

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09024/top)

Cho đồ thị có hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy viết thuật toán duyệt theo chiều rộng bắt đầu tại đỉnh uÎV (BFS(u)=?)

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào ba số |V|, |E|, uÎV tương ứng với số đỉnh, số cạnh và đỉnh bắt đầu duyệt; Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤200; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra danh sách các đỉnh được duyệt theo thuật toán BFS(u) của mỗi test theo khuôn dạng của ví dụ dưới đây.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  6  9  1  1 2 2 5 3 1 3 2 3 5 4 3 5 4 5 6 6 4 | 1 2 5 4 6 3 |

#include<bits/stdc++.h>

using namespace std;

int A[1005][1005], n, m ,u;

bool used[1005];

void init(){

cin >> n >> m >> u;

memset(A, 0, sizeof(A));

memset(used, false, sizeof(used));

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

int x, y; cin >> x >> y;

A[x][y] = 1;

}

}

void BFS(){

queue<int> Q;

Q.push(u); used[u] = true;

while(!Q.empty()){

int s = Q.front();

cout << s << " ";

Q.pop();

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

if (!used[t] && A[s][t]){

Q.push(t);

used[t] = true;

}

}

}

}

int main(){

int t; cin >> t;

while (t--){

init();

BFS();

cout << endl;

}

}

ĐƯỜNG ĐI THEO DFS VỚI ĐỒ THỊ CÓ HƯỚNG

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09025/top)

Cho đồ thị có hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy tìm đường đi từ đỉnh sÎV đến đỉnh tÎV trên đồ thị bằng thuật toán DFS.

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào bốn số |V|, |E|, sÎV, tÎV tương ứng với số đỉnh, số cạnh,  đỉnh u, đỉnh v; Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra đường đi từ đỉnh s đến đỉnh t của mỗi test theo thuật toán DFS của mỗi test theo khuôn dạng của ví dụ dưới đây. Nếu không có đáp án, in ra -1.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  6  9  1  6  1 2 2 5 3 1 3 2 3 5 4 3 5 4 5 6 6 4 | 1 2 5 6 |

#include<bits/stdc++.h>

using namespace std;

int n, m, s, e;

int matrix[1005][1005];

bool visited[1005];

int forward\_point[1005];

void init(){

cin >> n >> m >> s >> e;

memset(matrix, 0, sizeof(matrix));

memset(forward\_point, 0, sizeof(forward\_point));

memset(visited, false, sizeof(visited));

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

int x, y; cin >> x >> y;

matrix[x][y] = 1;

}

}

void DFS(int u)

{

stack<int> st;

st.push(u);

visited[u] = true;

while(!st.empty())

{

int s = st.top();

st.pop();

for(int t = 1; t <= n; t++)

{

if(visited[t] == false && matrix[s][t] == 1)

{

forward\_point[t] = s;

visited[t] = true;

st.push(s);

st.push(t);

break;

}

}

}

}

void PrintPath(int s, int e)

{

if(forward\_point[e] == 0)

cout << "-1" ;

else

{

vector<int> res;

res.push\_back(e);

int u = forward\_point[e];

while(u != s)

{

res.push\_back(u);

u = forward\_point[u];

}

res.push\_back(s);

for (int i = res.size()-1; i >= 0; i--)

cout << res[i] << " ";

}

}

int main(){

int T; cin >> T;

while (T--){

init();

DFS(s);

PrintPath(s, e);

cout << endl;

}

}

ĐƯỜNG THI THEO BFS TRÊN ĐỒ THỊ CÓ HƯỚNG

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09026/top)

Cho đồ thị có hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy tìm đường đi từ đỉnh uÎV đến đỉnh vÎV trên đồ thị bằng thuật toán BFS.

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào bốn số |V|, |E|, sÎV, tÎV tương ứng với số đỉnh, số cạnh,  đỉnh u, đỉnh v; |E| Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra đường đi từ đỉnh s đến đỉnh t của mỗi test theo thuật toán BFS của mỗi test theo khuôn dạng của ví dụ dưới đây. Nếu không có đáp án, in ra -1.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  6 9 1 6  1 2 2 5 3 1 3 2 3 5 4 3 5 4 5 6 6 4 | 1 2 5 6 |

#include<bits/stdc++.h>

using namespace std;

int n, m, s, e, matrix[1005][1005], f[1005];

bool used[1005];

void init(){

cin >> n >> m >> s >> e;

memset(matrix, 0, sizeof(matrix));

memset(f, 0, sizeof(f));

memset(used, false, sizeof(used));

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

int x, y; cin >> x >> y;

matrix[x][y] = 1;

}

}

void BFS(int u){

queue<int> Q;

Q.push(u); used[u] = true;

while(!Q.empty()){

int s = Q.front();

Q.pop();

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

if (!used[t] && matrix[s][t]){

f[t] = s;

used[t] = true;

Q.push(t);

}

}

}

}

void BFS\_Path(int s, int t){

if(f[t] == 0)

cout << "-1" << endl;

else

{

vector<int> res;

res.push\_back(t);

int u = f[t];

while(u != s)

{

res.push\_back(u);

u = f[u];

}

res.push\_back(s);

for (int i = res.size()-1; i >= 0; i--)

cout << res[i] << " ";

cout << endl;

}

}

int main(){

int T; cin >> T;

while (T--){

init();

BFS(s);

BFS\_Path(s, e);

}

}

Cho đồ thị vô hướng có N đỉnh và M cạnh. Có Q truy vấn, mỗi truy vấn yêu cầu trả lời câu hỏi giữa 2 đỉnh x và y có tồn tại đường đi tới nhau hay không?

**Input:**

* Dòng đầu tiên là số lượng bộ test T (T ≤ 20).
* Mỗi test gồm 2 số nguyên N, M (1 ≤ N, M ≤ 1000).
* M dòng tiếp theo, mỗi dòng gồm 2 số nguyên u, v cho biết có cạnh nối giữa đỉnh u và v.
* Dòng tiếp là số lượng truy vấn Q (1 ≤ Q ≤ 1000).
* Q dòng tiếp theo, mỗi dòng gồm 2 số nguyên x và y.

**Output:**Với mỗi truy vấn, in ra “YES” nếu có đường đi từ x tới y, in ra “NO” nếu ngược lại.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output** |
| 1  6 5  1 2  2 3  3 4  1 4  5 6  2  1 5  2 4 | NO  YES |

#include<bits/stdc++.h>

using namespace std;

int n, m;

vector<int> vt[1005];

bool used[1005];

int f[1005];

void BFS(int u, int v){

queue<int> Q;

Q.push(u); used[u] = true;

while(!Q.empty()){

int s = Q.front();

Q.pop();

if (s == v){

cout << "YES";

return;

}

for (int t=0; t<vt[s].size(); t++){

if (!used[vt[s][t]]){

f[vt[s][t]] = s;

used[vt[s][t]] = true;

Q.push(vt[s][t]);

}

}

}

cout << "NO";

}

void solve(int s, int e){

if(f[e] == 0)

cout << "NO" << endl;

else

cout << "YES" << endl;

}

int main(){

int T; cin >> T;

while (T--){

cin >> n >> m;

for (int i=0; i<1005; i++)

vt[i].clear();

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

int x, y; cin >> x >> y;

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

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

}

int q; cin >> q;

while (q--){

memset(used, false, sizeof(used));

int s, e;

cin >> s >> e;

BFS(s, e);

cout << endl;

}

}

}

 ĐƯỜNG ĐI THEO DFS VỚI ĐỒ THỊ VÔ HƯỚNG

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09006/top)

Cho đồ thị vô hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy tìm đường đi từ đỉnh sÎV đến đỉnh tÎV trên đồ thị bằng thuật toán DFS.

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào bốn số |V|, |E|, sÎV, tÎV tương ứng với số đỉnh, số cạnh,  đỉnh u, đỉnh v; Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra đường đi từ đỉnh s đến đỉnh t của mỗi test theo thuật toán DFS của mỗi test theo khuôn dạng của ví dụ dưới đây. Nếu không có đáp án, in ra -1.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  6  9 1 6  1 2 1 3 2 3 2 5 3 4 3 5 4 5 4 6 5 6 | 1 2 3 4 5 6 |

#include<bits/stdc++.h>

using namespace std;

int n, m, s, e;

int matrix[1005][1005];

bool visited[1005];

int forward\_point[1005];

void init(){

cin >> n >> m >> s >> e;

memset(matrix, 0, sizeof(matrix));

memset(forward\_point, 0, sizeof(forward\_point));

memset(visited, false, sizeof(visited));

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

int x, y; cin >> x >> y;

matrix[x][y] = 1;

matrix[y][x] = 1;

}

}

void DFS(int u)

{

stack<int> st;

st.push(u);

visited[u] = true;

while(!st.empty())

{

int s = st.top();

st.pop();

for(int t = 1; t <= n; t++)

{

if(visited[t] == false && matrix[s][t] == 1)

{

forward\_point[t] = s;

visited[t] = true;

st.push(s);

st.push(t);

break;

}

}

}

}

void PrintPath(int s, int e)

{

if(forward\_point[e] == 0)

cout << "-1" ;

else

{

vector<int> res;

res.push\_back(e);

int u = forward\_point[e];

while(u != s)

{

res.push\_back(u);

u = forward\_point[u];

}

res.push\_back(s);

for (int i = res.size()-1; i >= 0; i--)

cout << res[i] << " ";

}

}

int main(){

int T; cin >> T;

while (T--){

init();

DFS(s);

PrintPath(s, e);

cout << endl;

}

}

ĐƯỜNG ĐI THEO BFS TRÊN ĐỒ THỊ VÔ HƯỚNG

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09007/top)

Cho đồ thị vô hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy tìm đường đi từ đỉnh sÎV đến đỉnh tÎV trên đồ thị bằng thuật toán BFS.

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào bốn số |V|, |E|, sÎV, tÎV tương ứng với số đỉnh, số cạnh,  đỉnh u, đỉnh v; Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra đường đi từ đỉnh s đến đỉnh t của mỗi test theo thuật toán BFS của mỗi test theo khuôn dạng của ví dụ dưới đây. Nếu không có đáp án, in ra -1.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  6 9 1 6  1 2 1 3 2 3 2 5 3 4 3 5 4 5 4 6 5 6 | 1 2 5 6 |

#include<bits/stdc++.h>

using namespace std;

int n, m, s, e, matrix[1005][1005], f[1005];

bool used[1005];

void init(){

cin >> n >> m >> s >> e;

memset(matrix, 0, sizeof(matrix));

memset(f, 0, sizeof(f));

memset(used, false, sizeof(used));

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

int x, y; cin >> x >> y;

matrix[x][y] = 1;

matrix[y][x] = 1;

}

}

void BFS(int u){

queue<int> Q;

Q.push(u); used[u] = true;

while(!Q.empty()){

int s = Q.front();

Q.pop();

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

if (!used[t] && matrix[s][t]){

f[t] = s;

used[t] = true;

Q.push(t);

}

}

}

}

void BFS\_Path(int s, int t){

if(f[t] == 0)

cout << "-1" << endl;

else

{

vector<int> res;

res.push\_back(t);

int u = f[t];

while(u != s)

{

res.push\_back(u);

u = f[u];

}

res.push\_back(s);

for (int i = res.size()-1; i >= 0; i--)

cout << res[i] << " ";

cout << endl;

}

}

int main(){

int T; cin >> T;

while (T--){

init();

BFS(s);

BFS\_Path(s, e);

}

}

 ĐẾM SỐ THÀNH PHẦN LIÊN THÔNG

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09008/top)

Cho đồ thị vô hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy tìm số thành phần liên thông của đồ thị.

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh và số cạnh; Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra số thành phần liên thông của đồ thị.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  6 6  1 2 1 3 2 3 3 4 3 5 4 5 | 2 |

#include<bits/stdc++.h>

using namespace std;

int n, m, sol;

int matrix[1005][1005];

bool used[1005];

void init(){

cin >> n >> m;

sol = 0;

memset(matrix, 0, sizeof(matrix));

memset(used, false, sizeof(used));

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

int x, y; cin >> x >> y;

matrix[x][y] = 1;

matrix[y][x] = 1;

}

}

void DFS(int u){

used[u] = true;

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

if (matrix[u][v] && !used[v])

DFS(v);

}

}

void TPLT\_DFS(){

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

if (!used[u]){

sol++;

DFS(u);

}

}

cout << sol << endl;

}

int main(){

int t; cin >> t;

while (t--){

init();

TPLT\_DFS();

}

}

TÌM SỐ THÀNH PHẦN LIÊN THÔNG VỚI BFS

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09009/top)

Cho đồ thị vô hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy tìm số thành phần liên thông của đồ thị bằng thuật toán BFS.

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh và số cạnh; Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra số thành phần liên thông của đồ thị bằng thuật toán BFS.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  6  6  1 2 1 3 2 3 3 4 3 5 4 5 | 2 |

#include<bits/stdc++.h>

using namespace std;

int n, m, sol;

int matrix[1005][1005];

bool used[1005];

void init(){

cin >> n >> m;

sol = 0;

memset(matrix, 0, sizeof(matrix));

memset(used, false, sizeof(used));

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

int x, y; cin >> x >> y;

matrix[x][y] = 1;

matrix[y][x] = 1;

}

}

void BFS(int u){

queue<int> Q;

Q.push(u); used[u] = true;

while(!Q.empty()){

int s = Q.front();

Q.pop();

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

if (!used[t] && matrix[s][t]){

Q.push(t);

used[t] = true;

}

}

}

}

void TPLT\_BFS(){

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

if (!used[u]){

sol++;

BFS(u);

}

}

cout << sol << endl;

}

int main(){

int t; cin >> t;

while (t--){

init();

TPLT\_BFS();

}

}

KIỂM TRA TÍNH LIÊN THÔNG MẠNH

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09010/top)

Cho đồ thị có hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy kiểm tra xem đồ thị có liên thông mạnh hay không?

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh và số cạnh; Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra “YES”, hoặc “NO” theo từng dòng tương ứng với test là liên thông mạnh hoặc không liên thông mạnh.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  6 9  1 2 2 4 3 1 3 2 3 5 4 3 5 4 5 6 6  3 | YES |

#include<bits/stdc++.h>

using namespace std;

int n, m;

bool used[1005];

vector<int> vt[1005];

void BFS(int u){

queue<int> Q;

Q.push(u);

used[u] = true;

while(!Q.empty()){

int s = Q.front();

Q.pop();

for (int t=0; t<vt[s].size(); t++){

if (!used[vt[s][t]]){

Q.push(vt[s][t]);

used[vt[s][t]] = true;

}

}

}

}

int main(){

int T; cin >> T;

while (T--){

cin >> n >> m;

for (int i=0; i<1005; i++)

vt[i].clear();

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

int x, y; cin >> x >> y;

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

}

memset(used, false, sizeof(used));

int res = 0;

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

if (!used[i]){

res++;

BFS(i);

}

}

if (res >= 2) cout << "NO";

else cout << "YES";

cout << endl;

}

}

 LIỆT KÊ ĐỈNH TRỤ

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09012/top)

Cho đồ thị vô hướng liên thông G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy đưa ra tất cả các đỉnh trụ của đồ thị?

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh và số cạnh; Dòng tiếp theo đưa vào các bộ đôi u, v tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra danh sách các đỉnh trụ của mỗi test  theo từng dòng.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  5 5  1 2 1 3 2 3 2 5 3 4 | 2 3 |

#include<bits/stdc++.h>

using namespace std;

int n, m;

int matrix[1005][1005];

bool used[1005];

void init(){

cin >> n >> m;

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

for (int j=1; j<=n; j++)

matrix[i][j] = 0;

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

used[i] = false;

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

int x, y; cin >> x >> y;

matrix[x][y] = 1;

matrix[y][x] = 1;

}

}

int BFS(int u){

int c = 0;

queue<int> Q;

Q.push(u);

used[u] = true;

while(!Q.empty()){

int s = Q.front();

c++;

Q.pop();

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

if (!used[t] && matrix[s][t]){

Q.push(t);

used[t] = true;

}

}

}

return c;

}

void Reinit(){

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

used[i] = false;

}

void Dinh\_tru\_BFS(){

vector<int> res;

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

used[u] = true;

if (u == 1){

if (BFS(2) != n-1)

res.push\_back(u);

}

else{

if (BFS(1) != n-1)

res.push\_back(u);

}

Reinit();

}

for (int i = 0; i < res.size(); i++)

cout << res[i] << " ";

cout << endl;

}

int main(){

int T; cin >> T;

while (T--){

init();

Dinh\_tru\_BFS();

}

}

KIỂM TRA CHU TRÌNH TRÊN ĐỒ THỊ VÔ HƯỚNG

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09014/top)

Cho đồ thị vô hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy kiểm tra xem đồ thị có tồn tại chu trình hay không?

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh,  số cạnh của đồ thị; Dòng tiếp theo đưa vào các bộ đôi u, v tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra YES hoặc “NO” kết quả test theo từng dòng tương ứng với đồ thị tồn tại hoặc không tồn tại chu trình.

**Ví dụ:**

|  |  |
| --- | --- |
| Input: | Output: |
| 1  6  9  1 2 1 3 2 3 2 5 3 4 3 5 4 5 4 6 5 6 | YES |

#include<bits/stdc++.h>

using namespace std;

int n, m;

bool used[1005];

int f[1005];

vector<int> vt[1005];

bool isCycle(int u){

queue<int> Q;

Q.push(u);

used[u] = true;

while(!Q.empty()){

int s = Q.front();

Q.pop();

for (int t=0; t<vt[s].size(); t++){

if (!used[vt[s][t]]){

f[vt[s][t]] = s;

Q.push(vt[s][t]);

used[vt[s][t]] = true;

}

else if (f[s] != vt[s][t])

return true;

}

}

return false;

}

bool check(){

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

if (!used[i] && isCycle(i))

return true;

}

return false;

}

int main(){

int T; cin >> T;

while (T--){

cin >> n >> m;

for (int i=0; i<1005; i++)

vt[i].clear();

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

int x, y; cin >> x >> y;

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

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

}

memset(used, false, sizeof(used));

if (check()) cout << "YES";

else cout << "NO";

cout << endl;

}

}

KIỂM TRA CHU TRÌNH TRÊN ĐỒ THỊ CÓ HƯỚNG VỚI DFS

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09016/top)

Cho đồ thị có hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Sử dụng thuật toán DFS, hãy kiểm tra xem đồ thị có tồn tại chu trình hay không?

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh,  số cạnh của đồ thị; Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra YES hoặc “NO” kết quả test theo từng dòng tương ứng với đồ thị tồn tại hoặc không tồn tại chu trình.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  6 9  1 2 2 4 3 1 3 2 3 5 4 3 5 4 5 6 6 4 | YES |

#include<bits/stdc++.h>

using namespace std;

int n, m;

bool used[1005];

bool recStack[1005];

vector<int> vt[1005];

bool isCycle(int u){

if (!used[u]){

used[u] = true;

recStack[u] = true;

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

if (!used[vt[u][i]] && isCycle(vt[u][i]))

return true;

else if(recStack[vt[u][i]])

return true;

}

}

recStack[u] = false;

return false;

}

bool check(){

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

if (isCycle(i))

return true;

return false;

}

int main(){

int T; cin >> T;

while (T--){

cin >> n >> m;

for (int i=0; i<1005; i++)

vt[i].clear();

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

int x, y; cin >> x >> y;

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

}

memset(used, false, sizeof(used));

memset(recStack, false, sizeof(recStack));

if (check()) cout << "YES";

else cout << "NO";

cout << endl;

}

}

 KIỂM TRA CHU TRÌNH TRÊN ĐỒ THỊ CÓ HƯỚNG

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09015/top)

Cho đồ thị có hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy kiểm tra xem đồ thị có tồn tại chu trình hay không?

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh,  số cạnh của đồ thị; Dòng tiếp theo đưa vào các bộ đôi u, v tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra YES hoặc “NO” kết quả test theo từng dòng tương ứng với đồ thị tồn tại hoặc không tồn tại chu trình.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  6 9  1 2 2 4 3 1 3 2 3 5 4 3 5 4 5 6 6 4 | YES |

#include<bits/stdc++.h>

using namespace std;

int n, m;

bool used[1005];

bool recStack[1005];

vector<int> vt[1005];

bool isCycle(int u){

if (!used[u]){

used[u] = true;

recStack[u] = true;

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

if (!used[vt[u][i]] && isCycle(vt[u][i]))

return true;

else if(recStack[vt[u][i]])

return true;

}

}

recStack[u] = false;

return false;

}

bool check(){

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

if (isCycle(i))

return true;

return false;

}

int main(){

int T; cin >> T;

while (T--){

cin >> n >> m;

for (int i=0; i<1005; i++)

vt[i].clear();

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

int x, y; cin >> x >> y;

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

}

memset(used, false, sizeof(used));

memset(recStack, false, sizeof(recStack));

if (check()) cout << "YES";

else cout << "NO";

cout << endl;

}

}

KIỂM TRA ĐỒ THỊ CÓ PHẢI LÀ CÂY HAY KHÔNG

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09017/top)

Một đồ thị N đỉnh là một cây, nếu như nó có đúng N-1 cạnh và giữa 2 đỉnh bất kì, chỉ tồn tại duy nhất 1 đường đi giữa chúng.

Cho một đồ thị N đỉnh và N-1 cạnh, hãy kiểm tra đồ thị đã cho có phải là một cây hay không?

**Input:**

* Dòng đầu tiên là số lượng bộ test T (T ≤ 20).
* Mỗi test bắt đầu bởi số nguyên N (1 ≤ N ≤ 1000).
* N-1 dòng tiếp theo, mỗi dòng gồm 2 số nguyên u, v cho biết có cạnh nối giữa đỉnh u và v.

**Output:**

* Với mỗi test, in ra “YES” nếu đồ thị đã cho là một cây, in ra “NO” trong trường hợp ngược lại.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  4  1 2  1 3  2 4  4  1 2  1 3  2 3 | YES  NO |

#include<bits/stdc++.h>

using namespace std;

int n;

bool used[1005];

vector<int> adj[1005];

void Tree\_DFS(int u){

vector<vector<int> > vt;

stack<int> S;

S.push(u);

used[u] = true;

while (!S.empty()){

int s = S.top();

S.pop();

for (int t=0; t<adj[s].size(); t++){

if (!used[adj[s][t]]){

vector<int> v;

v.push\_back(s); v.push\_back(adj[s][t]);

vt.push\_back(v);

used[adj[s][t]] = true;

S.push(s); S.push(adj[s][t]);

break;

}

}

}

if (vt.size() < n-1)

cout << "NO" << endl;

else cout << "YES" << endl;

}

int main(){

int T; cin >> T;

while (T--){

cin >> n;

for (int i=0; i<1005; i++)

adj[i].clear();

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

int x, y; cin >> x >> y;

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

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

}

memset(used, false, sizeof(used));

Tree\_DFS(1);

}

}

Cho một bản đồ kích thước N x M được mô tả bằng ma trận A[][].A[i][j] = 1 có nghĩa vị trí (i, j) là nổi trên biển. 2 vị trí (i, j) và (x, y) được coi là liền nhau nếu như nó có chung đỉnh hoặc chung cạnh. Một hòn đảo là một tập hợp các điểm (i, j) mà A[i][j] = 1 và có thể di chuyển giữa hai điểm bất kì trong đó.

Nhiệm vụ của bạn là hãy đếm số lượng đảo xuất hiện trên bản đồ.

**Input:** Dòng đầu tiên là số lượng bộ test T (T ≤ 20).

Mỗi test bắt đầu bởi 2 số nguyên N và M (1 ≤ N, M ≤ 500).

N dòng tiếp theo, mỗi dòng gồm M số nguyên A[i][j].

**Output:** Với mỗi test, in ra số lượng hòn đảo tìm được.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output** |
| 1  5 5  1 1 0 0 0  0 1 0 0 1  1 0 0 1 1  0 0 0 0 0  1 0 1 0 1 | 5 |

#include <bits/stdc++.h>

using namespace std;

int isSafe(int n, int m, int M[][505], int row, int col, bool visited[][505]){

return (row >= 0) && (row < n) && (col >= 0) && (col < m) && (M[row][col] && !visited[row][col]);

}

void DFS(int n, int m, int M[][505], int row, int col, bool visited[][505]){

static int rowNbr[] = { -1, -1, -1, 0, 0, 1, 1, 1 };

static int colNbr[] = { -1, 0, 1, -1, 1, -1, 0, 1 };

visited[row][col] = true;

for (int k = 0; k < 8; ++k)

if (isSafe(n, m, M, row + rowNbr[k], col + colNbr[k], visited))

DFS(n, m, M, row + rowNbr[k], col + colNbr[k], visited);

}

int countIslands(int n, int m, int M[][505]){

bool visited[505][505];

memset(visited, 0, sizeof(visited));

int count = 0;

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

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

if (M[i][j] && !visited[i][j]) {

DFS(n, m, M, i, j, visited);

++count;

}

return count;

}

int main(){

int t; cin >> t;

while (t--){

int n, m, matrix[505][505];

cin >> n >> m;

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

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

cin >> matrix[i][j];

cout << countIslands(n , m, matrix) << endl;

}

}

TÔ MÀU ĐỒ THỊ

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09028/top)

Một trong những bài toán kinh điển của lý thuyết đồ thị là bài toán Tô màu đồ thị. Bài toán được phát biểu như sau: Cho đồ thị vô hướng G =<V, E> được biểu diễn dưới dạng danh sách cạnh và số M. Nhiệm vụ của bạn là kiểm tra xem đồ thị có thể tô màu các đỉnh bằng nhiều nhất M màu sao cho hai đỉnh kề nhau đều có màu khác nhau hay không?

Chart

Description automatically generated

**Input:**

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào ba số V, E, M tương ứng với số đỉnh, số cạnh và số màu; phần thứ hai đưa vào các cạnh của đồ thị.
* T, V, E, M thỏa mãn ràng buộc: 1≤T ≤100; 1≤V≤10; 1≤ E ≤N(N-1), 1≤V≤N.

**Output:**

* Đưa ra kết quả mỗi test theo từng dòng.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 4  5  3 1 2  2 3  3 4  4 1  1 3 3 3 2 1 2  2 3  1 3 | YES NO |

#include<bits/stdc++.h>

using namespace std;

int n, m, p;

int color[1005];

int matrix[1005][1005];

bool isSafe(int v, int c){

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

if (matrix[v][i] && c==color[i])

return false;

return true;

}

bool Try(int v){

if (v == n+1)

return true;

for (int c=1; c<=p; c++){

if (isSafe(v, c)){

color[v] = c;

if (Try(v+1) == true)

return true;

color[v] = 0;

}

}

return false;

}

int main(){

int t; cin >> t;

while (t--){

memset(matrix, 0 , sizeof(matrix));

memset(color, 0, sizeof(color));

cin >> n >> m >> p;

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

int x, y; cin >> x >> y;

matrix[x][y] = 1;

matrix[y][x] = 1;

}

if (Try(1) == false)

cout << "NO" << endl;

else

cout << "YES" << endl;

}

}

 ĐƯỜNG ĐI HAMILTON

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09029/top)

Đường đi đơn trên đồ thị có hướng hoặc vô hướng đi qua tất cả các đỉnh của đồ thị mỗi đỉnh đúng một lần được gọi là đường đi Hamilton. Cho đồ thị vô hướng G = <V, E>, hãy kiểm tra xem đồ thị có đường đi Hamilton hay không?

**Input:**

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào hai số V, E tương ứng với số đỉnh, số cạnh của đồ thị; phần thứ hai đưa vào các cạnh của đồ thị.
* T, V, E thỏa mãn ràng buộc: 1≤T ≤100; 1≤V≤10; 1≤ E ≤15.

**Output:**

* Đưa ra 1 hoặc 0 tương ứng với test có hoặc không có đường đi Hamilton theo từng dòng.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 4 4 1 2 2 3 3 4 2 4 4 3 1 2 2 3 2 4 | 1  0 |

#include<bits/stdc++.h>

using namespace std;

int n, m;

vector<int> adj[1005];

bool DFS(int u, int c, bool used[]){

if (c == n)

return true;

used[u] = true;

for (int v=0; v<adj[u].size(); v++){

if (!used[adj[u][v]]){

if (DFS(adj[u][v], c+1, used)){

return true;

}

}

}

used[u] = false;

return false;

}

bool isHamilton(){

bool used[1005];

memset(used, false, sizeof(used));

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

int c = 1;

if (DFS(i, c, used))

return true;

}

return false;

}

int main(){

int t; cin >> t;

while (t--){

cin >> n >> m;

for (int i=0; i<1005; i++)

adj[i].clear();

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

int x, y; cin >> x >> y;

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

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

}

if (isHamilton()) cout << 1;

else cout << 0;

cout << endl;

}

}

ĐỒ THỊ HAI PHÍA

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09030/top)

Đồ thị hai phía là một đồ thị đặc biệt, trong đó tập các đỉnh có thể được chia thành hai tập không giao nhau thỏa mãn điều kiện không có cạnh nối hai đỉnh bất kỳ thuộc cùng một tập. Cho đồ thị N đỉnh và M cạnh, bạn hãy kiểm tra đồ thị đã cho có phải là một đồ thị hai phía hay không?

Venn diagram

Description automatically generated

**Input:**

* Dòng đầu tiên là số lượng bộ test T (T ≤  20).
* Mỗi test bắt đầu bởi số nguyên N và M (1 ≤  N, M ≤  1000).
* M dòng tiếp theo, mỗi dòng gồm 2 số nguyên u, v cho biết có cạnh nối giữa đỉnh u và v.

**Output:**

* Với mỗi test, in ra “YES” nếu đồ thị đã cho là một đồ thị hai phía, in ra “NO” trong trường hợp ngược lại.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output** |
| 2  5 4  1 5  1 3  2 3  4 5  3 3  1 2  1 3  2 3 | YES  NO |

#include<bits/stdc++.h>

using namespace std;

int n, m;

vector<int> adj[1005];

int mark[1005];

bool bfs(int u){

queue<int> q;

q.push(u);

mark[u] = 1;

while (!q.empty()){

int s = q.front();

q.pop();

for (int t: adj[s]){

if (mark[s] == mark[t])

return false;

else if (!mark[t]){

q.push(t);

mark[t] = 3-mark[s];

}

}

}

return true;

}

void solve(){

memset(mark, 0, sizeof(mark));

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

if (mark[i]) continue;

if (!adj[i].size()){

mark[i] = 1;

continue;

}

if (!bfs(i)){

cout << "NO" << endl;

return;

}

}

cout << "YES" << endl;

}

int main(){

int t; cin >> t;

while (t--){

cin >> n >> m;

for (int i=0; i<1005; i++)

adj[i].clear();

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

int x, y; cin >> x >> y;

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

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

}

solve();

}

}

 LIỆT KÊ ĐỈNH TRỤ

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09018/top)

Cho đồ thị vô hướng liên thông G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy đưa ra tất cả các đỉnh trụ của đồ thị?

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh và số cạnh; Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra danh sách các đỉnh trụ của mỗi test  theo từng dòng.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  5 5  1 2 1 3 2 3 2 5 3 4 | 2 3 |

#include<bits/stdc++.h>

using namespace std;

int n, m;

int matrix[1005][1005];

bool used[1005];

void init(){

cin >> n >> m;

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

for (int j=1; j<=n; j++)

matrix[i][j] = 0;

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

used[i] = false;

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

int x, y; cin >> x >> y;

matrix[x][y] = 1;

matrix[y][x] = 1;

}

}

int BFS(int u){

int c = 0;

queue<int> Q;

Q.push(u);

used[u] = true;

while(!Q.empty()){

int s = Q.front();

c++;

Q.pop();

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

if (!used[t] && matrix[s][t]){

Q.push(t);

used[t] = true;

}

}

}

return c;

}

void Reinit(){

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

used[i] = false;

}

void Dinh\_tru\_BFS(){

vector<int> res;

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

used[u] = true;

if (u == 1){

if (BFS(2) != n-1)

res.push\_back(u);

}

else{

if (BFS(1) != n-1)

res.push\_back(u);

}

Reinit();

}

for (int i = 0; i < res.size(); i++)

cout << res[i] << " ";

cout << endl;

}

int main(){

int T; cin >> T;

while (T--){

init();

Dinh\_tru\_BFS();

}

}

KIỂM TRA CHU TRÌNH

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA09019/top)

Cho đồ thị vô hướng G=<V, E> được biểu diễn dưới dạng danh sách cạnh. Hãy kiểm tra xem đồ thị có tồn tại chu trình hay không?

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh,  số cạnh của đồ thị; Dòng tiếp theo đưa vào các bộ đôi uÎV, vÎV tương ứng với một cạnh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra YES hoặc “NO” kết quả test theo từng dòng tương ứng với đồ thị tồn tại hoặc không tồn tại chu trình.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  6  9  1 2 1 3 2 3 2 5 3 4 3 5 4 5 4 6 5 6 | YES |

#include<bits/stdc++.h>

using namespace std;

int n, m;

bool used[1005];

int f[1005];

vector<int> vt[1005];

bool isCycle(int u){

queue<int> Q;

Q.push(u);

used[u] = true;

while(!Q.empty()){

int s = Q.front();

Q.pop();

for (int t=0; t<vt[s].size(); t++){

if (!used[vt[s][t]]){

f[vt[s][t]] = s;

Q.push(vt[s][t]);

used[vt[s][t]] = true;

}

else if (f[s] != vt[s][t])

return true;

}

}

return false;

}

bool check(){

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

if (!used[i] && isCycle(i))

return true;

}

return false;

}

int main(){

int T; cin >> T;

while (T--){

cin >> n >> m;

for (int i=0; i<1005; i++)

vt[i].clear();

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

int x, y; cin >> x >> y;

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

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

}

memset(used, false, sizeof(used));

if (check()) cout << "YES";

else cout << "NO";

cout << endl;

}

}

 CÂY KHUNG CỦA ĐỒ THỊ THEO THUẬT TOÁN DFS

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA10006/top)

Cho đồ thị vô hướng G=(V, E). Hãy xây dựng một cây khung của đồ thị G với đỉnh u ∈ V là gốc của cây bằng thuật toán DFS.

**Input**

Dòng đầu tiên gồm một số nguyên T (1 ≤ T ≤ 20) là số lượng bộ test.

Tiếp theo là T bộ test, mỗi bộ test có dạng sau:

* Dòng đầu tiên gồm 3 số nguyên N=|V|, M=|E|, u (1 ≤ N ≤ 103, 1 ≤ M ≤ 105, 1 ≤ u ≤ N).
* M dòng tiếp theo, mỗi dòng gồm 2 số nguyên a, b (1 ≤ a, b ≤ N, a ≠ b) tương ứng cạnh nối hai chiều từ a tới b.
* Dữ liệu đảm bảo giữa hai đỉnh chỉ tồn tại nhiều nhất một cạnh nối.

**Output**

Với mỗi bộ test, nếu tồn tại cây khung thì in ra N – 1 cạnh của cây khung với gốc là đỉnh u trên N – 1 dòng theo thứ tự duyệt của thuật toán DFS. Ngược lại nếu không tồn tại cây khung thì in ra -1.

**Ví dụ**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  4 3 2  1 2  1 3  2 4  3 4  4 2 2  1 2  3 4 | 2 1  1 3  3 4  -1 |

#include<bits/stdc++.h>

using namespace std;

int n, m, u;

vector<int> adj[1005];

bool used[1005];

void Tree\_DFS(){

vector<vector<int> > vt;

stack<int> S;

S.push(u);

used[u] = true;

while (!S.empty()){

int s = S.top();

S.pop();

for (int t=0; t<adj[s].size(); t++){

if (!used[adj[s][t]]){

vector<int> v;

v.push\_back(s); v.push\_back(adj[s][t]);

vt.push\_back(v);

used[adj[s][t]] = true;

S.push(s); S.push(adj[s][t]);

break;

}

}

}

if (vt.size() < n-1)

cout << "-1" << endl;

else{

for (int i=0; i<vt.size(); i++){

for (int j=0; j<vt[i].size(); j++)

cout << vt[i][j] << " ";

cout << endl;

}

}

}

int main(){

int t; cin >> t;

while (t--){

cin >> n >> m >> u;

memset(used, false, sizeof(used));

for (int i=0; i<1005; i++)

adj[i].clear();

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

int x, y; cin >> x >> y;

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

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

}

Tree\_DFS();

}

}

CÂY KHUNG CỦA ĐỒ THỊ THEO THUẬT TOÁN BFS

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA10007/top)

Cho đồ thị vô hướng G=(V, E). Hãy xây dựng một cây khung của đồ thị G với đỉnh u ∈ V là gốc của cây bằng thuật toán BFS.

**Input**

Dòng đầu tiên gồm một số nguyên T (1 ≤ T ≤ 20) là số lượng bộ test.

Tiếp theo là T bộ test, mỗi bộ test có dạng sau:

* Dòng đầu tiên gồm 3 số nguyên N=|V|, M=|E|, u (1 ≤ N ≤ 103, 1 ≤ M ≤ 105, 1 ≤ u ≤ N).
* M dòng tiếp theo, mỗi dòng gồm 2 số nguyên a, b (1 ≤ a, b ≤ N, a ≠ b) tương ứng cạnh nối hai chiều từ a tới b.
* Dữ liệu đảm bảo giữa hai đỉnh chỉ tồn tại nhiều nhất một cạnh nối.

**Output**

Với mỗi bộ test, nếu tồn tại cây khung thì in ra N – 1 cạnh của cây khung với gốc là đỉnh u trên N – 1 dòng theo thứ tự duyệt của thuật toán BFS. Ngược lại nếu không tồn tại cây khung thì in ra -1.

**Ví dụ**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  4 4 2  1 2  1 3  2 4  3 4  4 2 2  1 2  3 4 | 2 1  2 4  1 3  -1 |

#include<bits/stdc++.h>

using namespace std;

int n, m, u;

vector<int> adj[1005];

bool used[1005];

void Tree\_BFS(){

vector<vector<int> > vt;

queue<int> Q;

Q.push(u);

used[u] = true;

while (!Q.empty()){

int s = Q.front();

Q.pop();

for (int t=0; t<adj[s].size(); t++){

if (!used[adj[s][t]]){

vector<int> v;

v.push\_back(s);

v.push\_back(adj[s][t]);

vt.push\_back(v);

Q.push(adj[s][t]);

used[adj[s][t]] = true;

}

}

}

if (vt.size() < n-1)

cout << "-1" << endl;

else{

for (int i = 0; i < vt.size(); i++){

for (int j = 0; j < vt[i].size(); j++)

cout << vt[i][j] << " ";

cout << endl;

}

}

}

int main(){

int t; cin >> t;

while (t--){

cin >> n >> m >> u;

memset(used, false, sizeof(used));

for (int i=0; i<1005; i++)

adj[i].clear();

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

int x, y; cin >> x >> y;

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

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

}

Tree\_BFS();

}

}

 DIJKSTRA

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA10008/top)

Cho đồ thị có trọng số không âm G=<V, E> được biểu diễn dưới dạng danh sách cạnh trọng số. Hãy viết chương trình tìm đường đi ngắn nhất từ đỉnh uÎV đến tất cả các đỉnh còn lại trên đồ thị.

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm |E|+1 dòng: dòng đầu tiên đưa vào hai ba số |V|, |E| tương ứng với số đỉnh và uÎV là đỉnh bắt đầu; |E| dòng tiếp theo mỗi dòng đưa vào bộ ba uÎV, vÎV, w tương ứng với một cạnh cùng với trọng số canh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra kết quả của mỗi test theo từng dòng. Kết quả mỗi test là trọng số đường đi ngắn nhất từ đỉnh u đến các đỉnh còn lại của đồ thị theo thứ tự tăng dần các đỉnh.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  9  12 1  1  2   4  1  8   8  2  3   8  2  8   11  3  4   7  3  6   4  3  9   2  4  5   9  4  6  14  5  6  10  6  7  2  6  9  6 | 0 4 12 19 21 11 9 8 14 |

#include<bits/stdc++.h>

#define mp make\_pair

using namespace std;

typedef pair<int, int> iPair;

int n, m, s;

vector<iPair> adj[1005];

void dijkstra(){

priority\_queue<iPair, vector<iPair>, greater<iPair> > pq;

vector<int> dist(n+1, INT\_MAX);

pq.push(mp(0, s));

dist[s] = 0;

while (!pq.empty()){

int u = pq.top().second;

pq.pop();

for(auto x: adj[u]){

int v = x.first;

int w = x.second;

if (dist[v] > dist[u]+w){

dist[v] = dist[u]+w;

pq.push(mp(dist[v], v));

}

}

}

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

cout << dist[i] << " ";

cout << endl;

}

int main(){

int t; cin >> t;

while (t--){

cin >> n >> m >> s;

for (int i=0; i<1005; i++)

adj[i].clear();

while(m--){

int u, v, w; cin >> u >> v >> w;

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

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

}

dijkstra();

}

}

KRUSKAL

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA10015/top)

Cho đồ thị vô hướng có trọng số G=<V, E, W>. Nhiệm vụ của bạn là hãy xây dựng một cây khung nhỏ nhất của đồ thị bằng thuật toán Kruskal.

**Input:**

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào hai số V, E tương ứng với số đỉnh và số cạnh của đồ thị; phần thứ 2 đưa vào E cạnh của đồ thị, mỗi cạnh là một bộ 3: đỉnh đầu, đỉnh cuối và trọng số của cạnh.
* T, S, D thỏa mãn ràng buộc: 1≤T≤100;  1≤V≤100; 1≤E, W≤10000.

**Output:**

* Đưa ra kết quả mỗi test theo từng dòng.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 3 3 1 2 5  2 3 3  1 3 1 2 1 1 2 5 | 4 5 |

#include<bits/stdc++.h>

typedef long long ll;

using namespace std;

int n, m;

struct edge{

int u;

int v;

int w;

};

edge edges[10005];

int parent[10005];

int rankz[10005];

bool compare(edge x, edge y){

if(x.w < y.w) return true;

return false;

}

int find(int u){

if (parent[u] == u) return parent[u];

return find(parent[u]);

}

bool merge(int u, int v){

u = find(u); v = find(v);

if (u == v) return false;

if (rankz[u] == rankz[v]){

rankz[u]++;

}

if (rankz[u] < rankz[v])

parent[u] = v;

else parent[v] = u;

return true;

}

void solve(){

long long res = 0;

sort(edges, edges+m, compare);

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

if (merge(edges[i].u, edges[i].v))

res += edges[i].w;

}

cout << res << endl;

}

int main(){

int t; cin >> t;

while (t--){

cin >> n >> m;

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

cin >> edges[i].u >> edges[i].v >> edges[i].w;

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

parent[i] = i;

rankz[i] = 0;

}

solve();

}

}

CHU TRÌNH ÂM

[Bài làm tốt nhất](https://code.ptit.edu.vn/student/question/DSA10017/top)

Cho đồ thị có trọng số G=<V, E> được biểu diễn dưới dạng danh sách cạnh trọng số âm hoặc dương. Hãy viết chương trình xác định xem đồ thị có chu trình âm hay không.

**Input:**

* Dòng đầu tiên đưa vào T là số lượng bộ test.
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm |E|+1 dòng: dòng đầu tiên đưa vào hai số |V|, |E| tương ứng với số đỉnh và số cạnh của đồ thị; |E| dòng tiếp theo mỗi dòng đưa vào bộ ba uÎV, vÎV, w tương ứng với một cạnh cùng với trọng số canh của đồ thị.
* T, |V|, |E| thỏa mãn ràng buộc: 1≤T≤100; 1≤|V|≤103; 1≤|E|≤|V|(|V|-1)/2;

**Output:**

* Đưa ra 1 hoặc 0 theo từng dòng của mỗi test tương ứng với đồ thị có hoặc không có chu trình âm.

**Ví dụ:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 2  3  3  1  2 -1  2  3  4  3  1 -2  3  3  1  2 -1  2  3  2  3  1 -2 | 0  1 |

#include<bits/stdc++.h>

using namespace std;

int n, m;

struct edge{

int u;

int v;

int w;

};

edge edges[10005];

bool isNegCycle(int k, int dis[]){

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

dis[i] = INT\_MAX;

dis[k] = 0;

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

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

int u = edges[j].u;

int v = edges[j].v;

int w = edges[j].w;

if (dis[u] != INT\_MAX && dis[u]+w<dis[v])

dis[v] = dis[u]+w;

}

}

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

int u = edges[i].u;

int v = edges[i].v;

int w = edges[i].w;

if (dis[u] != INT\_MAX && dis[u]+w<dis[v]){

return true;

}

}

return false;

}

void solve(){

bool visited[10005];

memset(visited, 0, sizeof(visited));

int dis[10005];

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

if (!visited[i])

if(isNegCycle(i, dis)){

cout << 1 << endl;

return;

}

}

cout << 0 << endl;

}

int main(){

int t; cin >> t;

while (t--){

cin >> n >> m;

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

cin >> edges[i].u >> edges[i].v >> edges[i].w;

solve();

}

}