Tên: Huỳnh Công Tính

MSSV: 3124411315

Lớp DCT124C6

Github: congtinh06

Câu 1:

- Ý tưởng: Đếm có bao nhiêu đỉnh trong cùng thành phần liên thông với đỉnh bắt đầu rổi in ra các đỉnh có thể đi tới được từ đỉnh bắt đầu.

- Code:

#include <iostream>

using namespace std;

const int MAX = 100005;

int n, start;

int adj[MAX][MAX];

int deg[MAX];

bool visited[MAX];

void DFS(int *u*) {

visited[u] = true;

for (int i = 0; i < deg[u]; i++) {

int v = adj[u][i];

if (!visited[v]) {

DFS(v);

}

}

}

int main() {

cin >> n >> start;

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

int v;

while (cin >> v && v != -1) {

adj[i][deg[i]++] = v;

}

}

DFS(start);

int count = 0;

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

if (visited[i]) count++;

}

cout << count << "\n";

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

if (visited[i]) cout << i << " ";

}

cout << "\n";

return 0;

}

Câu 2:

- Ý tưởng: Dùng bfs từ đỉnh bắt đầu. Ghi nhớ parent của mỗi đỉnh khi duyệt qua, nếu target không được thì in 0, nguọc lại thì về đỉnh bắt đầu để theo parent tim đường. In ra số đỉnh trên đường đi và các đỉnh theo thứ tự

- Code:

#include <iostream>

#include <queue>

using namespace std;

const int MAX = 1005;

int n, start, target;

int adj[MAX][MAX];

int deg[MAX];

bool visited[MAX];

int parent[MAX];

void BFS(int *s*) {

queue<int> q;

q.push(*s*);

visited[*s*] = true;

parent[*s*] = -1; // gốc không có cha

while (!q.empty()) {

int u = q.front(); q.pop();

for (int i = 0; i < deg[u]; i++) {

int v = adj[u][i];

if (!visited[v]) {

visited[v] = true;

parent[v] = u;

q.push(v);

}

}

}

}

int main() {

cin >> n >> start >> target;

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

int v;

while (cin.peek() != '\n' && cin >> v) {

adj[i][deg[i]++] = v;

}

cin.ignore();

}

BFS(start);

if (!visited[target]) {

cout << "0\n";

return 0;

}

int path[MAX], len = 0;

for (int v = target; v != -1; v = parent[v]) {

path[len++] = v;

}

// In kết quả

cout << len << "\n";

for (int i = len - 1; i >= 0; i--) {

cout << path[i] << " ";

}

cout << "\n";

return 0;

}

Câu 3:

- Ý tưởng: tô màu các đỉnh bằng hai màu khác nhau(0,1). Nếu hai đỉnh kề nhau có cùng mày thì không phân đôi và ngược lại.

- Code:

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

int main() {

int n;

cin >> n;

vector<vector<int>> graph(n + 1);

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

while (true) {

int x;

cin >> x;

if (x == -1) break;

graph[i].push\_back(x);

}

}

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

for (int x : graph[i]) {

graph[x].push\_back(i);

}

}

vector<int> color(n + 1, -1);

bool isBipartite = true;

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

if (color[i] != -1) continue;

queue<int> q;

q.push(i);

color[i] = 0;

while (!q.empty() && isBipartite) {

int u = q.front();

q.pop();

for (int v : graph[u]) {

if (color[v] == -1) {

color[v] = 1 - color[u];

q.push(v);

} else if (color[v] == color[u]) {

isBipartite = false;

break;

}

}

}

}

if (isBipartite)

cout << "Do thi phan doi\n";

else

cout << "Do thi khong phan doi\n";

return 0;

}

Câu 3(2/2):

-Ý tưởng: Đọc danh sách kề để đựng đồ thị, dùng bfs đi từ 1 đỉnh chưa thăm -> duyệt hết các đỉnh trong thành phần liên thông đó. Lưu các đỉnh thuộc mỗi phần. Sau khi duyệt hết ta có danh sách các thành phần-> sắp xếp rồi in ra

- Code:

#include <iostream>

#include <queue>

#include <vector>

#include <algorithm>

using namespace std;

int n;

vector<int> adj[1005];

bool visited[1005];

void BFS(int *start*, vector<int> &*component*) {

queue<int> q;

q.push(*start*);

visited[*start*] = true;

while (!q.empty()) {

int u = q.front(); q.pop();

*component*.push\_back(u);

// duyệt các đỉnh kề

for (int v : adj[u]) {

if (!visited[v]) {

visited[v] = true;

q.push(v);

}

}

}

}

int main() {

cin >> n;

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

int v;

while (cin >> v && v != -1) {

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

}

}

vector<vector<int>> components;

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

if (!visited[i]) {

vector<int> comp;

BFS(i, comp);

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

components.push\_back(comp);

}

}

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

//in kết quả

cout << components.size() << "\n";

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

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

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

}

cout << "\n";

}

return 0;

}

Câu 4(2/2):

- Ý tưởng: +cạnh cầu: tạm bỏ cạnh x,-> rồi chạy bfs rừ x. Nếu không còn đến được y -> cạnh đó là cầu.

. + đỉnh khớp: tạm bỏ đinh z, rồi chạy bfs từ 1 đỉnh khác . Nếu còn đỉnh nào khônh thăm được -> z là đỉnh khớp

- Code:

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

int main() {

int n, x, y, z;

cin >> n >> x >> y >> z;

vector<vector<int>> graph(n + 1);

//đọc danh sách kề

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

while (true) {

int v;

cin >> v;

if (v == -1) break;

graph[i].push\_back(v);

}

}

//kiểm tra cạnh cầu

vector<bool> visited(n + 1, false);

queue<int> q;

visited[x] = true;

q.push(x);

while (!q.empty()) {

int u = q.front();

q.pop();

for (int v : graph[u]) {

if ((u == x && v == y) || (u == y && v == x)) continue;

if (!visited[v]) {

visited[v] = true;

q.push(v);

}

}

}

if (visited[y] == false)

cout << "canh cau\n";

else

cout << "khong la canh cau\n";

//kiểm tra đỉnh khớp

int start = 1;

if (start == z) start = 2;

visited.assign(n + 1, false);

visited[z] = true;

q.push(start);

visited[start] = true;

while (!q.empty()) {

int u = q.front();

q.pop();

for (int v : graph[u]) {

if (!visited[v]) {

visited[v] = true;

q.push(v);

}

}

}

bool articulation = false;

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

if (!visited[i]) {

articulation = true;

break;

}

}

if (articulation)

cout << "dinh khop\n";

else

cout << "khong la dinh khop\n";

return 0;

}