Tema 1 Lab

Semigrupa 1, Grupa 231

Student: Tudose Alexandru-Stefan

Laborant: Narcis Lucian Trepteanu

Problema 1) <https://leetcode.com/problems/possible-bipartition/>

a)

class Solution {

public:

void dfs(int n, vector <vector<int>> &adjL, int viz[], int dist, int & ok, int tata){

if (viz[n] == 0){

if (dist % 2 == 0)

viz[n] = 2;

else

viz[n] = 1;

for (auto &it : adjL[n])

if (it != tata)

dfs(it, adjL, viz, dist + 1, ok, n);

}

else{

if (dist % 2 != viz[n] % 2)

ok = 0;

return;

}

}

bool possibleBipartition(int n, vector<vector<int>>& dislikes) {

vector<vector <int>> adjL;

adjL.assign(n, {});

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

adjL[dislikes[i][0]-1].push\_back(dislikes[i][1]-1);

adjL[dislikes[i][1]-1].push\_back(dislikes[i][0]-1);

}

int viz[n], ok = 1;

memset(viz, 0, n\*sizeof(int));

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

if (viz[i] == 0){

dfs(i, adjL, viz, 0, ok, i);

}

}

return ok;

}

};

b) Aceeasi functie dfs

vector <vector <int>> possibleBipartition(int n, vector<vector<int>>& dislikes) {

vector<vector <int>> adjL;

adjL.assign(n, {});

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

adjL[dislikes[i][0]-1].push\_back(dislikes[i][1]-1);

adjL[dislikes[i][1]-1].push\_back(dislikes[i][0]-1);

}

int viz[n], ok = 1;

memset(viz, 0, n\*sizeof(int));

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

if (viz[i] == 0){

dfs(i, adjL, viz, 0, ok, i);

}

}

vector <vector<int>>rez(2);

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

if (viz[i] == 1)

rez[0].push\_back(i+1);

if (viz[i] == 2)

rez[1].push\_back(i+1);

}

return rez;

}

};

2.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int n, m, i;

vector <int> perm, pozInPerm;

bool compFct(int a, int b){ ///aici trebuie sa sortez toate listele de adiacenta in functie de pozitia in permutare

return pozInPerm[b] > pozInPerm[a];

}

void dfs(int n, vector<int> & viz, vector<vector<int>> & adjL, vector <int> &rez){

viz[n] = 1;

rez.push\_back(n);

for (auto &it : adjL[n]){

if(viz[it] == 0)

dfs(it, viz, adjL, rez);

//--pozPerm;

}

}

int main()

{

cin >> n >> m;

perm.assign(n, 0);

pozInPerm.assign(n, 0);

vector <int> viz(n, 0);

vector <vector <int>> adjL(n);

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

cin >> perm[i];

--perm[i];

pozInPerm[perm[i]] = i;

}

int m1, m2;

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

cin >> m1 >> m2;

adjL[m1-1].push\_back(m2-1);

adjL[m2-1].push\_back(m1-1);

}

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

if (!adjL[i].empty()){

sort(adjL[i].begin(), adjL[i].end(), compFct);

}

}

vector <int> rez;

rez.reserve(n);

dfs(perm[0], viz, adjL, rez);

bool ok = 1;

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

if (rez[i] != perm[i])

ok = 0;

cout << ok;

return 0;

}

3. a)

class Solution {

public:

void dfs(vector <vector <int>> & lAdj, int start, vector <int> & viz, stack<int> & rez, bool & ciclu){

if (ciclu == 1)

return;

viz[start] = 1;

for (auto & i : lAdj[start]){

if (viz[i] == 0)

dfs(lAdj, i, viz, rez, ciclu);

else if (viz[i] == 1){

ciclu = 1;

return;

}

}

viz[start] = 2;

rez.push(start);

}

vector<int> findOrder(int numCourses, vector<vector<int>> & prerequisites) {

vector <vector <int>> lAdj(numCourses);

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

lAdj[prerequisites[i][1]].push\_back(prerequisites[i][0]);

}

vector <int> viz(numCourses, 0);

stack <int> rez;

bool ciclu = false;

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

if (viz[i] == 0)

dfs(lAdj, i, viz, rez, ciclu);

}

vector <int> rezz;

if (ciclu)

return rezz;

//rezz.push\_back(rez.size());

while (!rez.empty()){

rezz.push\_back(rez.top());

rez.pop();

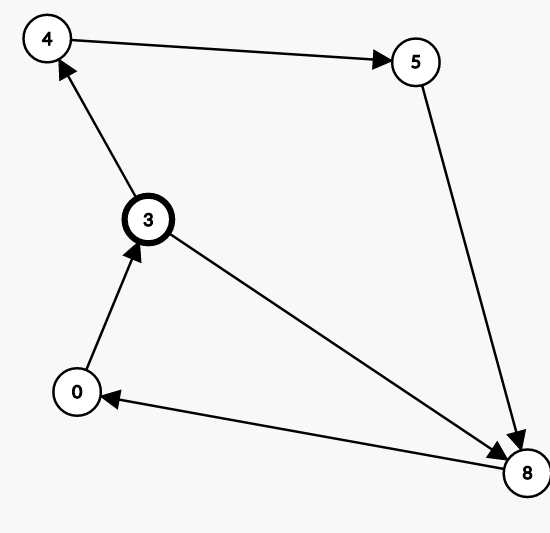
}

return rezz;

}

};

b)



Din cate am inteles, intr-o astfel de situatie, vectorul rezultat va fi [4, 5, 8, 0, 3, 4, 8]

Sau [4, 5, 8, 0, 3, 8, 4], asa ca am modificat functia ca atare

class Solution {

public:

void dfs(vector <vector <int>> & lAdj, int start, vector <int> & viz, stack<int> & rez, bool & ciclu){

~~if (ciclu == 1)~~

~~return;~~

viz[start] = 1;

for (auto & i : lAdj[start]){

if (viz[i] == 0)

dfs(lAdj, i, viz, rez, ciclu);

else if (viz[i] == 1){

ciclu = 1;

rez.push(i);

~~return;~~

}

}

viz[start] = 2;

rez.push(start);

}

vector<int> findOrder(int numCourses, vector<vector<int>> & prerequisites) {

vector <vector <int>> lAdj(numCourses);

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

lAdj[prerequisites[i][1]].push\_back(prerequisites[i][0]);

}

vector <int> viz(numCourses, 0);

stack <int> rez;

bool ciclu = false;

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

if (viz[i] == 0)

dfs(lAdj, i, viz, rez, ciclu);

}

vector <int> rezz;

if (!ciclu)

return rezz;

//rezz.push\_back(rez.size());

while (!rez.empty()){

rezz.push\_back(rez.top());

rez.pop();

}

return rezz;

}

};

4.

#include <iostream>

#include <fstream>

#include <vector>

#include <stack>

using namespace std;

ifstream fin("ctc.in");

ofstream fout("ctc.out");

int rez;

vector <vector <int>> rezz;

void ctc(int aici, vector <vector <int>> & adjL, vector <pair <int, int>> & viz, stack <int> & stv, int & pozStiva, vector <int> & inStiva){

for (auto &it : adjL[aici]){

if (viz[it].first == -1){

viz[it].first = pozStiva;

viz[it].second = pozStiva;

inStiva[it] = 1;

stv.push(it);

++pozStiva;

ctc(it, adjL, viz, stv, pozStiva, inStiva);

viz[aici].second = min(viz[aici].second, viz[it].second);

}

else if (inStiva[it]){

viz[aici].second = min(viz[aici].second, viz[it].second);

}

}

if (viz[aici].second == viz[aici].first){

rezz.push\_back({});

while (aici != stv.top()){

rezz[rez].push\_back(stv.top()+1);

inStiva[stv.top()] = 0;

stv.pop();

}

rezz[rez].push\_back(aici+1);

inStiva[aici] = 0;

stv.pop();

++rez;

}

}

int n, m, nr1, nr2, pozStiva;

int main()

{

fin >> n >> m;

vector <vector <int>> adjL(n);

vector <int> inStiva(n);

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

{

fin >> nr1 >> nr2;

adjL[nr1-1].push\_back(nr2-1);

}

vector <pair <int, int>> viz(n, make\_pair(-1, 0));

stack <int> stv;

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

if (viz[i].first == -1){

viz[i].first = 0;

viz[i].second = 0;

pozStiva = 1;

inStiva[i] = 1;

stv.push(i);

ctc(i, adjL, viz, stv, pozStiva, inStiva);

}

}

fout << rez << '\n';

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

for (auto j : rezz[i])

fout << j << ' ';

fout << '\n';

}

return 0;

}

5.

#include <iostream>

#include <fstream>

#include <vector>

#include <queue>

using namespace std;

ifstream fin("graf.in");

ofstream fout("graf.out");

int n, m, nr1, nr2;

int main()

{

fin >> n >> m;

vector <vector <int>> adjL(n);

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

fin >> nr1 >> nr2;

adjL[nr1-1].push\_back(nr2-1);

adjL[nr2-1].push\_back(nr1-1);

}

vector <int> dist(n, -1);

queue <int> q;

while (fin >> nr1){

q.push(nr1-1);

dist[nr1-1] = 0;

}

cout << "am ajuns1\n";

while (!q.empty()){

nr1 = q.front();

//cout << "am ajuns2\n";

for (auto & j : adjL[nr1]){

if (dist[j] == -1){

dist[j] = dist[nr1]+1;

q.push(j);

}

}

q.pop();

//cout << "am ajuns2\n";

}

for (auto & j : dist){

fout << j << ' ';

}

return 0;

}