**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

дисциплина: Дискретная математика

Лабораторная работа №4.3

тема: «**Связность**»

|  |  |
| --- | --- |
|  | Выполнил: ст. группы ВТ-202  Проверил: Рязанов Ю.Д. |

Белгород

2021

Изображение выглядит как текст

Автоматически созданное описание

**Задание 1:**

1. Реализовать алгоритм Краскала построения покрывающего леса.

#include <iostream>

#include <vector>

using namespace::std;

//изменят элемнты класса а на класс b

void addToNewClass(vector<size\_t>& bouquets, size\_t a, size\_t b) {

for (size\_t i = 0; i < bouquets.size(); i++) {

if (bouquets[i] == a)

bouquets[i] = b;

}

}

//алгоритм Краскала

vector<vector<int>> kraskalAlg(const vector<vector<int>>& a, vector<size\_t>& bouquets) {

vector<vector<int>> res;

res.resize(a.size());

for (size\_t i = 0; i < a.size(); i++) {

res[i].resize(a[i].size(), 0);

}

for (size\_t i = 0; i < a.size(); i++)

for (size\_t j = i; j < a.size(); j++)

if (a[i][j] && bouquets[i] != bouquets[j]) {

res[i][j] = true;

res[j][i] = true;

addToNewClass(bouquets, bouquets[i], bouquets[j]);

}

return res;

}

int main() {

size\_t n;

cout << " Enter count of vertex :";

cin >> n;

vector<vector<int>> a, b;

a.resize(n);

for (size\_t i = 0; i < n; i++) {

a[i].resize(n, 0);

}

size\_t i, j;

while (cin >> i >> j) {

a[i - 1][j - 1] = 1;

}

vector<size\_t> bouquets;

for (size\_t i = 1; i <= n; i++) {

bouquets.push\_back(i);

}

b = kraskalAlg(a, bouquets);

cout << " result:\n";

for (size\_t i = 0; i < a.size(); i++) {

for (size\_t j = 0; j < a.size(); j++)

if (b[i][j])

cout << "( " << i + 1 << ", " << j + 1 << " ) ";

cout << endl;

}

}

Изображение выглядит как текст

Автоматически созданное описание

**Задание 2:**

1. Используя алгоритм Краскала, разработать и реализовать алгоритм решения задач.

**Вариант 3**

Найти минимальное множество вершин, удаление которых из связного графа разбивает его на три связные компоненты.

#include <iostream>

#include <set>

#include <vector>

#include <map>

using namespace std;

void addToNewClass(vector<size\_t>& bouquets, size\_t a, size\_t b) {

for (size\_t i = 0; i < bouquets.size(); i++) {

if (bouquets[i] == a)

bouquets[i] = b;

}

}

size\_t kraskalAlg(const vector<vector<int>>& a) {

vector<size\_t> bouquets;

for (size\_t i = 0; i < a.size(); i++) {

bouquets.push\_back(i);

}

bool f = 0;

pair<int, int> p;

for (size\_t j = 0; j < a[0].size(); j++) {

for (size\_t i = 0; i < a.size(); i++) {

if (a[i][j]) {

if (!f) {

p.first = i;

f = 1;

}

else {

p.second = i;

break;

}

}

}

f = 0;

if (bouquets[p.first] != bouquets[p.second]) {

addToNewClass(bouquets, bouquets[p.first], bouquets[p.second]);

}

p.first = p.second = 0;

}

set<int> s;

for (size\_t i = 0; i < bouquets.size(); i++) {

s.insert(bouquets[i]);

}

return s.size();

}

pair<int, int> takeFromIndent(vector<vector<int>> eOfOrigin, size\_t j) {

pair<int, int> a;

a.first;

a.second;

bool f = 0;

for (size\_t i = 0; i < eOfOrigin[0].size(); i++) {

if (eOfOrigin[i][j]) {

if (!f) {

a.first = i;

f = 1;

}

else {

a.second = i;

break;

}

}

}

return a;

}

void innerForMinSetVert(vector<vector<int>> eOfOrigin, set<int>& r) {

static int sup[100];

static int i = 0;

static map<int, int> reflectOfOriginE;

static vector<vector<int>> eOfSubset;

for (int x = 0; x != 2; x++) {

sup[i] = x;

if (i == eOfOrigin.size() - 1) {

for (int j = 0; j < eOfOrigin.size(); j++)

if (sup[j])

reflectOfOriginE[j] = reflectOfOriginE.size();

eOfSubset.resize(reflectOfOriginE.size());

int k = 0;

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

pair<int, int> p = takeFromIndent(eOfOrigin, j);

if (reflectOfOriginE.count(p.first) != 0

&& reflectOfOriginE.count(p.second) != 0) {

int size = eOfSubset[0].size();

for (int z = 0; z < eOfSubset.size(); z++)

eOfSubset[z].resize(size + 1);

eOfSubset[reflectOfOriginE[p.first]][k] =

eOfSubset[reflectOfOriginE[p.second]][k] = 1;

k++;

}

}

if (eOfSubset.size() != 0 && kraskalAlg(eOfSubset) == 3) {

if ((eOfOrigin.size() - reflectOfOriginE.size()) <

r.size() || r.size() == 0) {

r.clear();

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

if (reflectOfOriginE.count(i) == 0)

r.insert(i);

}

}

reflectOfOriginE.clear();

eOfSubset.clear();

}

else {

i++;

innerForMinSetVert(eOfOrigin, r);

i--;

}

}

}

set<int> MinSetVertToBreakGraph(vector<vector<int>> e) {

set<int> r;

innerForMinSetVert(e, r);

return r;

}

int main() {

vector<vector<int>> e;

size\_t m, n;

cout << " Enter count of vertex: ";

cin >> n;

cout << " Enter count of edjes: ";

cin >> m;

e.resize(n);

for (size\_t i = 0 ; i < n; i++)

e[i].resize(m, 0);

size\_t i, j;

while (cin >> i >> j) {

e[i - 1][j - 1] = 1;

}

set<int> r;

r = MinSetVertToBreakGraph(e);

for (size\_t i = 0; i < n; i++) {

if (r.count(i)) {

cout << i + 1 << " ";

}

}/\*

cout << kraskalAlg(e);\*/

return 0;

}

**Задание 3:**

1. Подобрать тестовые данные. Результат представить в виде диаграммы графа.

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Результат |
| 1 | 7  7  1 1  1 7  2 1  2 2  3 2  3 3  4 3  4 4  5 4  5 5  6 5  6 6  7 6  7 7 | 1 3 5 |
| 2 | 4  3  1 1  2 2  3 1  3 2  3 3  4 3 | 3 |
| 3 | 4  4  1 1  1 4  2 1  2 2  3 2  3 3  4 3  4 4 | deletion is not possible! |

**1.**

Изображение выглядит как текст

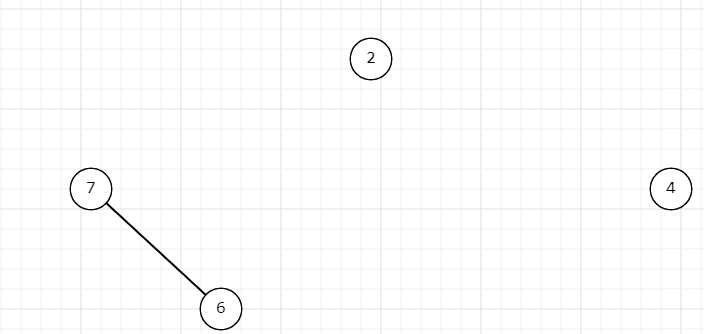
Автоматически созданное описание

Исходный граф:

Изображение выглядит как провод

Автоматически созданное описание

После удаления:



**2.**

Изображение выглядит как текст

Автоматически созданное описание

Исходный граф:

Изображение выглядит как стена, внутренний, стойка, с плиткой

Автоматически созданное описание

После удаления:

Изображение выглядит как стена, с плиткой

Автоматически созданное описание

**3.**

Изображение выглядит как текст

Автоматически созданное описание

Исходный граф:

Изображение выглядит как стена, внутренний, белый, стойка

Автоматически созданное описание