**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

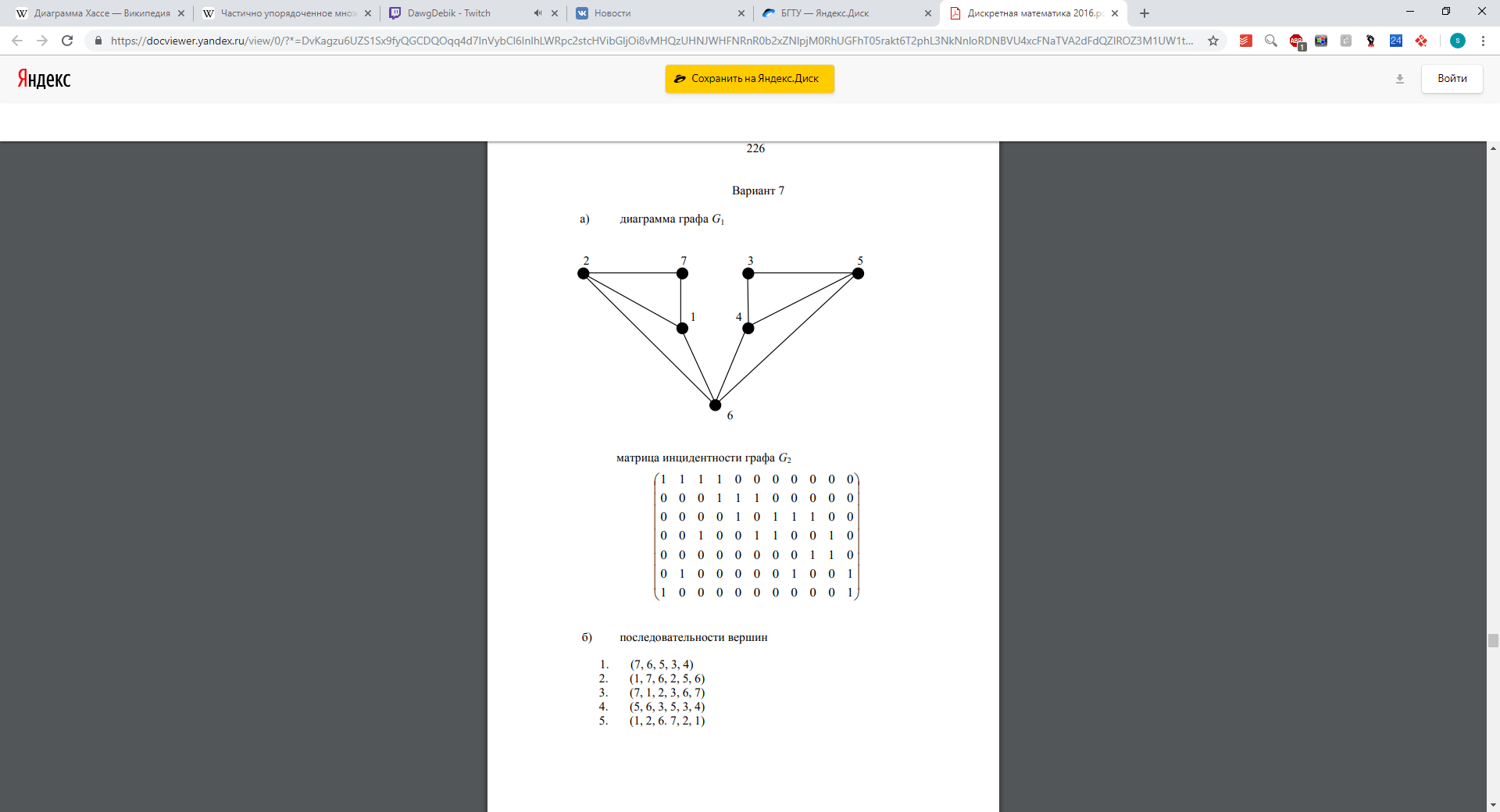
Лабораторная работа № 4.2

тема: «Циклы»

|  |  |
| --- | --- |
|  | Выполнил: ст. группы ВT-22  Макаров Даниил Сергеевич  Проверил: Рязанов Ю.Д. |

Белгород

2018

**Цель занятия:** изучить основные понятия теории графов, способы задания графов, научиться программно реализовывать алгоритмы получения и анализа маршрутов в графах.

**Задания**

1. Представить графы G1 и G2 (см. п.а) матрицей смежности, матрицей инцидентности, диаграммой.

|  |  |  |
| --- | --- | --- |
| G1 | | |
| Матрица смежности | Матрица инцидентности | Диаграмма |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| G2 | | |
| Матрица смежности | Матрица инцидентности | Диаграмма |
|  |  |  |

2. Определить, являются ли последовательности вершин (см. п.б) маршрутом, цепью, простой цепью, циклом, простым циклом в графах G1 и G2 (см. п.а).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| G1 | | | | | |
|  | Маршрут | Цепь | Простая цепь | Цикл | Простой цикл |
| (7,6,5,3,4) | - | - | - | - | - |
| (1,7,6,2,5,6) | - | - | - | - | - |
| (7,1,2,3,6,7) | - | - | - | - | - |
| (5,6,3,4,3,4) | - | - | - | - | - |
| (1,2,6,7,2,1) | - | - | - | - | - |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| G2 | | | | | |
|  | Маршрут | Цепь | Простая цепь | Цикл | Простой цикл |
| (7,6,5,3,4) | - | - | - | - | - |
| (1,7,6,2,5,6) | - | - | - | - | - |
| (7,1,2,3,6,7) | - | - | - | - | - |
| (5,6,3,4,3,4) | - | - | - | - | - |
| (1,2,6,7,2,1) | - | - | - | - | - |

3. Написать программу, определяющую, является ли заданная последовательность вершин (см. п.б) маршрутом, цепью, простой цепью, циклом, простым циклом в графах G1 и G2 (см. п.а).

4. Написать программу, получающую все маршруты заданной длины, выходящие из заданной вершины. Использовать программу для получения всех маршрутов заданной длины в графах G1 и G2 (см. п.а).

5. Написать программу, определяющую количество маршрутов заданной длины между каждой парой вершин графа. Использовать программу для определения количества маршрутов заданной длины между каждой парой вершин в графах G1 и G2 (см., п.а).

6. Написать программу, определяющую все маршруты заданной длины между заданной парой вершин графа. Использовать программу для определения всех маршрутов заданной длины между заданной парой вершин в гра­фах G1 и G2 (см., п.а).

#include <stdio.h>

#include <stdlib.h>

int IsRoute(int \*seq, int len, int \*\*graph, int n){

int i=0, x, y;

while (i<len-1){

x = seq[i];

y = seq[i+1];

if (!graph[x][y]){

break;

}

i++;

}

return i == len-1;

}

int IsChain(int \*seq, int len, int \*\*graph, int n){

int \*\*E = malloc(n \* sizeof(int\*));

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

E[i] = malloc(n \* sizeof(int));

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

E[i][j] = 0;

}

}

int x, y, i = 0;

while (i < len-1){

x = seq[i];

y = seq[i+1];

if (!graph[x][y] || E[x][y] || E[y][x]){

break;

}

E[x][y] = E[y][x] = 1;

i++;

}

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

free(E[i]);

}

free(E);

return i==len-1;

}

int IsSimpleChain(int \*seq, int len, int \*\*graph, int n){

int i=0, j, flag=1, x, y;

while (i<len-1 && flag){

x = seq[i];

y = seq[i+1];

if (!graph[x][y]){

flag = 0;

}

j=0;

while (j<i+1 && flag){

flag = (seq[i+1] != seq[j]);

j++;

}

i++;

}

return flag;

}

int IsCycle(int \*seq, int len, int \*\*graph, int n){

return seq[0] == seq[len-1] && IsChain(seq, len, graph, n);

}

int IsSimpleCycle(int \*seq, int len, int \*\*graph, int n){

int last = seq[len-1];

int pre\_last = seq[len-2];

return seq[0] == last && graph[last][pre\_last]

&& IsSimpleChain(seq, len-1, graph, n);

}

void PrintRoute(int \*W, int l){

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

printf("%i ", W[i]+1);

}

printf("\n");

}

void MakeRoutes(int \*\*G, int n, int l, int i, int v, int \*W){

W[i] = v;

i++;

if (i==l+1){

PrintRoute(W, l+1);

}else{

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

if (G[v][j]){

MakeRoutes(G, n, l, i, j, W);

}

}

}

}

int \*\*MultMatr(int\*\* a, int\*\* b, int n){

int \*\*c=(int\*\*)calloc(n, sizeof(int\*));

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

c[i]=(int\*)calloc(n, sizeof(int));

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

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

c[i][j]=0;

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

c[i][j]+=(a[i][k]\*b[k][j]);

}

c[i][j] -= 1;

}

}

return c;

}

int \*\*CountRoutes(int \*\*G, int n, int l){

// t - копия G

int \*\*t=malloc(n\*sizeof(int \*));

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

t[i]=malloc(n\*sizeof(int));

t[i]=G[i];

}

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

t=MultMatr(t, G, n);

}

return t;

}

void MakeRoutes2 (int \*\*G, int n, int l, int i, int x, int y, int \*W){

W[i] = x;

i++;

if (i==l+1){

if (W[i-1] == y){

PrintRoute(W, l+1);

}

}else{

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

if (G[x][j]){

MakeRoutes2(G, n, l, i, j, y, W);

}

}

}

}

void MakeMaxChain(int \*\*G, int n, int i, int v, int \*W, int \*visited){

W[i-1] = v;

int flag = 0;

if (i==n) return;

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

if (G[v][x] && !visited[x]){

flag = 1;

W[i] = x;

visited[x] = 1;

MakeMaxChain(G, n, i+1, x, W, visited);

visited[x] = 0;

}

}

if (flag == 0){

PrintRoute(W, i);

}

}

int main(){

int n;

FILE \*f = fopen("graph2.txt", "r");

if (f == NULL){

printf("NULL\n");

}

fscanf(f, "%i", &n);

printf("n = %i\n", n);

int \*\*G = malloc(n \* sizeof(int\*));

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

G[i] = malloc(n \* sizeof(int));

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

fscanf(f, "%i", &G[i][j]);

}

}

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

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

printf("%i ", G[i][j]);

}

printf("\n");

}

int \*W = malloc(sizeof(int)\*n);

int \*visited = malloc(sizeof(int)\*n);

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

visited[i] = 0;

}

printf("\n");

MakeMaxChain(G, n, 1, 0, W, visited);

fclose(f);

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

free(G[i]);

}

free(G);

return 0;

}