**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 № 4.1

тема: «Маршруты»

|  |  |
| --- | --- |
|  | Выполнил: ст. группы ПВ-21  Зановская Анна Ивановна  Проверил: Рязанов Ю.Д. |

Белгород

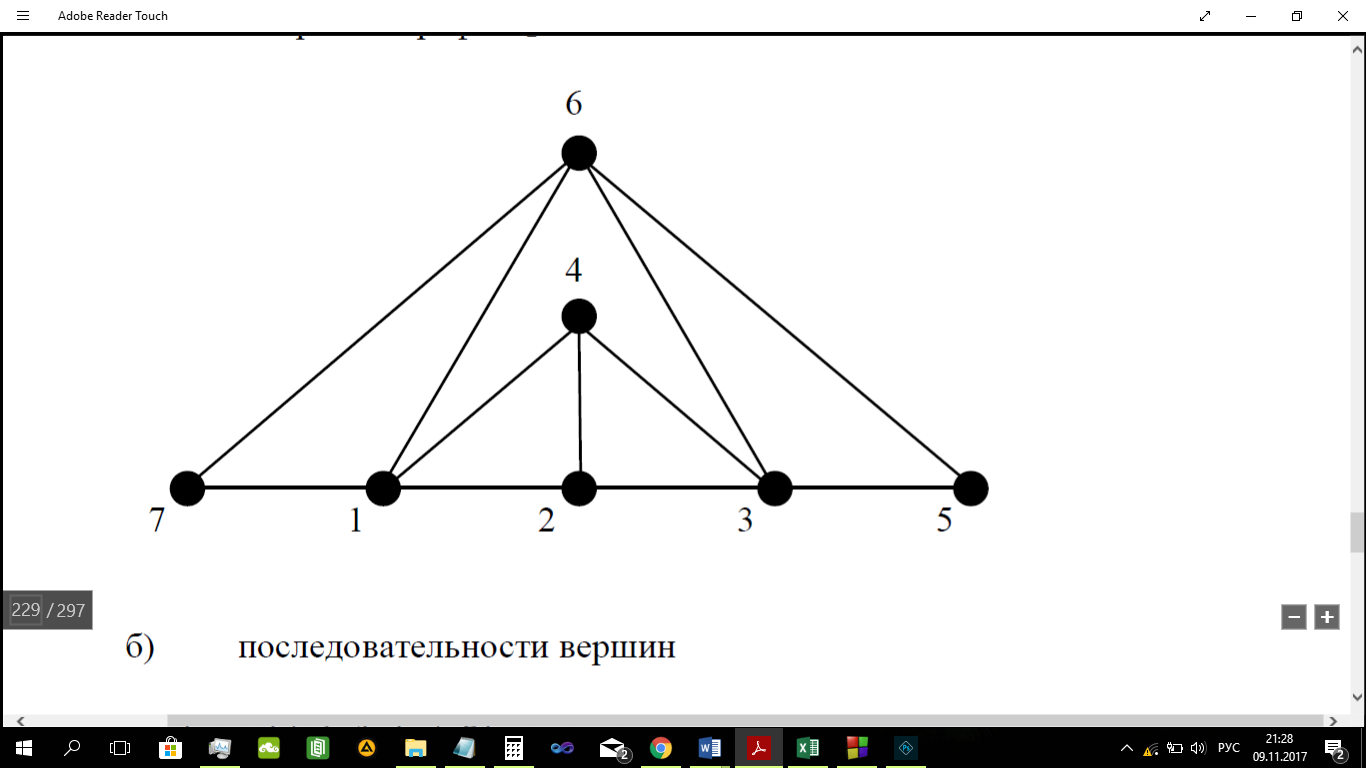
2017

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

**Вариант 10**

а) матрица инцидентности графа G1

диаграмма графа G2



б) последовательности вершин

1. (2, 3, 4, 1, 7, 6)

2. (4, 1, 6, 7. 1, 2)

3. (1, 4, 3, 2, 1)

4. (1, 2, 4, 3, 2, 1)

5. (2, 4. 1, 7, 6, 1, 2

**Задания**

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

|  |  |  |
| --- | --- | --- |
| G1 | | |
| Матрица смежности | Матрица инцидентности | Диаграмма |
|  |  | 7  6  3  2  1  4  5 |

|  |  |  |
| --- | --- | --- |
| 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 |
| (2, 3, 4, 1, 7, 6) | - | - | - | - | - |
| (4, 1, 6, 7. 1, 2) | - | - | - | - | - |
| (1, 4, 3, 2, 1) | - | - | - | - | - |
| (1, 2, 4, 3, 2, 1) | - | - | - | - | - |
| (2, 4. 1, 7, 6, 1, 2) | - | - | - | - | - |

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

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).

int IsMarsh (int \*\*a, size\_t n)

{

int i=0, l;

printf("Длина маршрута: ");

scanf("%i",&l);

int \*m=(int\*)calloc(l, sizeof(int));

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

scanf("%i",&m[i]);

i=0;

while (i<l-1 && (a[m[i]-1][m[i+1]-1]!=0))

i++;

return i==l-1 ? 1 : 0;

}

int IsChain (int \*\*a, size\_t n)//это цепь?

{

int i, l;

printf("Длина маршрута: ");

scanf("%i", &l);

int \*m=(int\*)calloc(l, sizeof(int));

for (i=0;i<l;i++) //тут хранится порядок вершин

scanf("%i", &m[i]);

int \*\*tmp = (int\*\*)calloc(n, sizeof(int\*)); //нулевая матрица для провкерки

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

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

int k=0, p=0;

i=0;

while (i<l-1 && (a[m[i]-1][m[i+1]-1]!=0))

{

k=m[i]-1; //получили индексы

p=m[i+1]-1; //для записи в tmp

if (tmp[k][p]==1 && tmp[p][k]==1)

return 0;

tmp[k][p]=1;

tmp[p][k]=1;

i++;

}

return i==l-1 ? 1 : 0;

}

int IsSimpleChain (int \*\*a, size\_t n)

{

int i=0, l, j, flag=1;

printf("Длина маршрута: ");

scanf("%i",&l);

int \*m=(int\*)calloc(l, sizeof(int));

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

scanf("%i",&m[i]);

i=0;

while (i<l && flag && (a[m[i]-1][m[i+1]-1]!=0))

{

j=0;

while (j<i-1 && flag)

{

flag = m[i]!=m[j];

j++;

}

i++;

}

return flag;

}

int IsCycle (int \*\*a, size\_t n)

{

int i, l;

printf("Длина маршрута: ");

scanf("%i", &l);

int \*m=(int\*)calloc(l, sizeof(int));

for (i=0;i<l;i++) //тут хранится порядок вершин

scanf("%i", &m[i]);

if(m[0]!=m[l-1])

return 0;

int \*\*tmp = (int\*\*)calloc(n, sizeof(int\*)); //нулевая матрица для провкерки

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

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

int k=0, p=0;

i=0;

while (i<l-1 && (a[m[i]-1][m[i+1]-1]!=0))

{

k=m[i]-1; //получили индексы

p=m[i+1]-1; //для записи в tmp

if (tmp[k][p]==1 && tmp[p][k]==1)

return 0;

tmp[k][p]=1;

tmp[p][k]=1;

i++;

}

return i==l-1 ? 1 : 0;

}

int IsSimpleCycle (int \*\*a, size\_t n)

{

int i=0, l, j, flag=1;

printf("Длина маршрута: ");

scanf("%i",&l);

int \*m=(int\*)calloc(l, sizeof(int));

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

scanf("%i",&m[i]);

if(m[0]!=m[l-1])

return 0;

i=0;

while (i<l-1 && flag && (a[m[i]-1][m[i+1]-1]!=0))

{

j=i+1;

while (j<l-1 && flag)

{

flag = m[i]!=m[j];

j++;

}

i++;

}

return flag;

}

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).

void MakeMarshouts (int \*\*g, size\_t n, size\_t l, int i, int x)

// g - матрица смежности рафа, n - размер, l -длина марш.

// заполняемое место = i

{

W[i-1] = x;

int j;

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

{

if (g[x-1][j])

{

if (i!=l)

MakeMarshouts(g, n, l, i+1, j+1);

else

{

W[i]=j+1;

OutputMarch(W, l+1);

}

}

}

}

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).

int\*\* KolMarsh (int \*\*a, size\_t n, size\_t l)

{

int i,j;

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

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

{

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

k[i]=a[i];

}

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

k=MultMatr(k, a, n, n);

return k;

}

int\*\* MultMatr(int\*\* a, int\*\* b, size\_t n, size\_t m)

{

int i, j, k;

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

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

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

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

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

{

//c[i][j]=0;

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

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

}

return c;

}

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).

void MakeMarshXY (int \*\*g, size\_t n, size\_t l, int i, int x, int y)

// g - матрица смежности рафа, n - размер, l -длина марш.

// заполняемое место = i, x-1, y-last

{

W[i-1] = x;

int j;

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

{

if (g[x-1][j])

if (i==l-1)

{

if (g[y-1][j])

{

W[i]=j+1;

W[l]=y;

OutputMarch(W, l+1);

}

}

else

MakeMarshoutsXY(g, n, l, i+1, j+1, y);

}

}