Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
"Белгородский государственный технологический университет им. В. Г. Шухова"
(БГТУ им. В.Г. Шухова)

Институт энергетики, информационных технологий и управляющих систем

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа № 4.1
по дисциплине дискретная математика
тема: Маршруты

Выполнил: студент группы ПВ-223
Игнатьев Артур Олегович
Проверил: доцент
Рязанов Юрий Дмитриевич

Белгород 2023

Цель работы: 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.”Варианты заданий”, п.а) матрицей смежности, матрицей инцидентности, диаграммой.

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

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

7. Написать программу, получающую все простые максимальные цепи, выходящие из заданной вершины графа. Использовать программу для

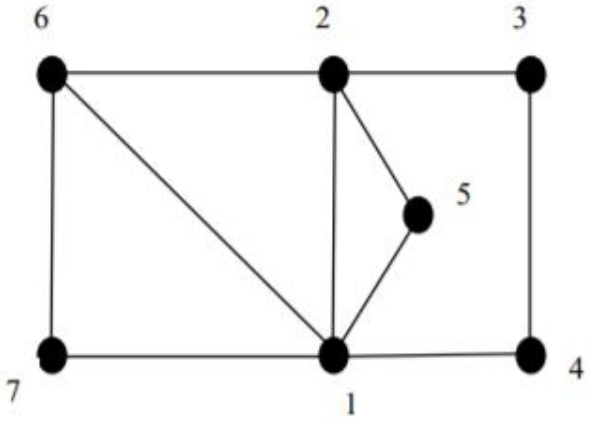
получения всех простые максимальных цепей, выходящих из заданной вершины в графах G1 и G2 (см.”Варианты заданий”, п.а).

Вариант 4

а) матрица инцидентности графа G_1

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

диаграмма графа G_2



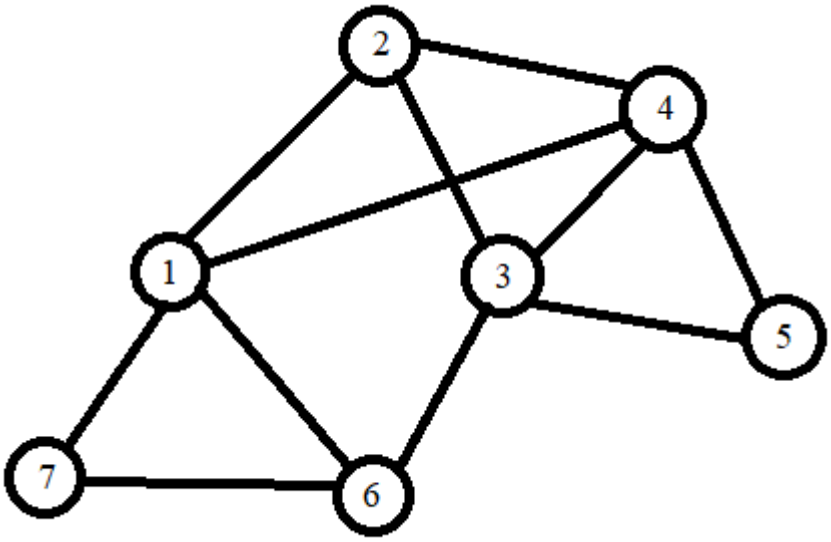
б) последовательности вершин

1. (2, 3, 4, 1, 5)
2. (4, 3, 6, 5, 3, 4, 2)
3. (6, 1, 2, 5, 1, 7)
4. (1, 7, 6. 2, 1)
5. (2, 1. 7, 6, 1. 4, 3, 2)

1. Представить графы G1 и G2 матрицей смежности, матрицей инцидентности, диаграммой.

G1

Диаграмма:



Матрица смежности:

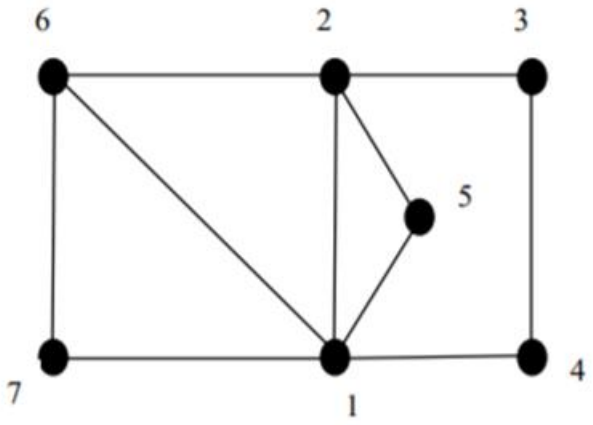
0	1	0	1	0	1	1
1	0	1	1	0	0	0
0	1	0	1	1	1	0
1	1	1	0	1	0	0
0	0	1	1	0	0	0
1	0	1	0	0	0	1
1	0	0	0	0	1	0

Матрица инцидентности:

1	1	1	1	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	1	0	1	1	1	0	0
0	0	1	0	0	1	1	0	0	1	0
0	0	0	0	0	0	0	0	1	1	0
0	1	0	0	0	0	0	1	0	0	1
1	0	0	0	0	0	0	0	0	0	1

G2

Диаграмма:



Матрица смежности:

0	1	0	1	1	1	1
1	0	1	0	1	1	0
0	1	0	1	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	0	0	0	0	1
1	0	0	0	0	1	0

Матрица инцидентности:

1	0	1	1	0	0	0	0	1	1
1	1	0	0	0	1	1	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	1	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	1
0	0	0	0	0	0	0	1	1	0

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

1. (2, 3, 4, 1, 5)
Для G1: не является маршрутом
Для G2: маршрут, цепь, простая цепью

2. (4, 3, 6, 5, 3, 4, 2)
Для G1: не является маршрутом
Для G2: не является маршрутом

3. (6, 1, 2, 5, 1,7)
Для G1: не является маршрутом
Для G2: маршрут, цепь

4. (1, 7, 6, 2, 1)
Для G1: не является маршрутом
Для G2: маршрут, цепь, простая цепь, простой цикл,цикл

5. (2, 1, 7, 6 ,1, 4, 3,2)
Для G1: не является маршрутом
Для G2: маршрут, цепь, цикл

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).

```
#include <stdio.h>
#include <malloc.h>
#include <windows.h>

#define N 15

// Является ли данная последовательность pos маршрутом для графа graf
int is_marshrut(int *pos, int **graf, int n) {
    int flag = 1, i = 0;
    while (flag && (i < (n - 1))) {
        if ((graf[pos[i] - 1][pos[i + 1] - 1]) == 0)
            flag = 0;
        i++;
    }
    return flag;
}

// Является ли данная последовательность pos цепью для графа graf
int is_cep(int *pos, int n) {
    int a[N] = { 0 };// Логическое множество для ребер
    int i = 0;
    int flag = 1;
    while (flag && (i < (n - 1))) {
        if (a[pos[i]] == pos[i + 1]) {
            flag = 0;
        }
        a[pos[i]] = pos[i + 1];
        i++;
    }
    return flag;
}
// Является ли данная последовательность pos простой цепью для графа graf
int is_easy_cep(int *pos, int n, int m) {
    int i;
    int *a;
    int flag = 1;
    a = (int*)malloc((m + 1) * sizeof(int));
    for (i = 0; i < (m + 1); i++) {
        a[i] = 0;
    }
    i = 0;
    while (flag && i < (n - 1)) {
        if (a[pos[i]] == 1)
            flag = 0;
        a[pos[i]] = 1;
        i++;
    }
    return flag;
}
// Является ли данная последовательность pos циклом для графа graf
int is_cikl(int *pos, int n) {
    if (pos[0] == pos[n - 1]) {
        return 1;
    }
    return 0;
}
int is_easy_cikl(int *pos, int n, int m) {
```

```
    if (pos[0] == pos[n - 1]) {
        int i;
        int *a;
        int flag = 1;

        a = (int*)malloc((m) * sizeof(int));
        for (i = 1; i < m; i++) {
            a[i] = 0;
        }
        i = 0;
        while (flag && i < (n - 1)) {
            if (a[pos[i]] == 1)
                flag = 0;
            a[pos[i]] = 1;
            i++;
        }
        return flag;
    }
    return 0;
}
//Инициализация графа a ввиде матрицы смежности n*n
int ** init_graf(int n) {
    int i, j;
    int **a;

    // Выделение памяти под указатели на строки
    a = (int**)malloc(n * sizeof(int*));
    printf("\nВведите граф ввиде матрицы смежности %d: \n", n);
    // Ввод элементов графа
    for (i = 0; i < n; i++) {
        // Выделение памяти под хранение строк
        a[i] = (int*)malloc(n * sizeof(int));
        for (j = 0; j < n; j++) {
            scanf("%d", &a[i][j]);
        }
    }
    return a;
}
int * init_posl(int n) {
    int i;
    int *a;
    // Выделение памяти
    a = (int*)malloc(n * sizeof(int));
    printf("\nВведите последовательность длинны %d: ", n);
    // Ввод элементов последовательности
    for (i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }
    return a;
}

void output_marsh(int *a, int l) {
    int i;
    for (i = 0; i < (l + 1); i++)
        printf("%d ", (a[i]));
    printf("\n");
}

void marshruti(int *a, int **graf, int i, int l, int n) {

    int x;
    for (x = 1; x <= n; x++) {
        if ((graf[a[i - 1]][x] == 1) && (x != a[i - 1])) {
            a[i] = x;
```

```
            if (i == l)
                output_marsh(a, l);
            else
                marshruti(a, graf, i + 1, l, n);
        }
    }
    return;
}

void output_marsh1(int *a) {
    int i = 0;
    while (a[i] > 0) {
        printf("%d ", a[i]);
        i++;
    }

    printf("\n");
}

int find_in_cep(int a, int **graf, int *log_v, int n) {
    for (int i = 1; i <= n; i++) {
        if ((graf[a][i] == 1) && (i != a) && (log_v[i] == 0))
            return 1;
    }
    return 0;
}

void all_max_easy_cepi(int *a, int **graf, int *log_v, int i, int n) {
    int x;
    for (x = 1; x <= n; x++) {
        if ((graf[a[i - 1]][x] == 1) && (x != a[i - 1]) && (log_v[x] == 0)) {
            a[i] = x;
            if (!(find_in_cep(a[i], graf, log_v, n)))
                output_marsh1(a);
            else {
                log_v[x] = 1;
                all_max_easy_cepi(a, graf, log_v, i + 1, n);
                log_v[x] = 0;
            }
        }
    }
}


int main() {
    SetConsoleOutputCP(CP_UTF8);

    int n, m;
    int *a; // массив последовательности
    int **graf; // Матрица графа

    printf("\nВведите длинну последовательности: ");
    scanf("%d", &n);

    a = init_posl(n);

    printf("\nВведите мощность множества элементов графа: ");
    scanf("%d", &m);
    graf = init_graf(m);

    if (is_marshrut(a, graf, n)) {
        printf("\n Данная последовательность для данного графа является
маршрутом!");
        if (is_cep(a, n)) {
```

```
            printf("\n Данная последовательность для данного графа является
цепью!");
            if (is_easy_cep(a, n, m))
                printf("\n Данная последовательность для данного графа
является простой цепью!");
            if (is_easy_cikl(a, n, m))
                printf("\n Данная последовательность для данного графа
является простым циклом!");
        }
        else {
            printf("\n Данная последовательность для данного графа не
является цепью, а значит и пр. цепью и пр. циклом!");
        }

        if (is_cikl(a, n))
            printf("\n Данная последовательность для данного графа является
циклом!");
    }
    else {
        printf("\n Данная последовательность для данного графа не является
маршрутом! Следовательно и ничем больше.");
    }


    return 0;
}
```

```
Введите длинну последовательности:5

Введите последовательность длинны 5:2 3 4 1 5

Введите мощность множества элементов графа:7

Введите граф ввиде матрицы смежности 7:
0 1 0 1 1 1 1
1 0 1 0 1 1 0
0 1 0 1 0 0 0
1 0 1 0 0 0 0
1 1 0 0 0 0 0
1 1 0 0 0 0 1
1 0 0 0 0 1 0

 Данная последовательность для данного графа является маршрутом!
 Данная последовательность для данного графа является цепью!
 Данная последовательность для данного графа является простой цепью!
```

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.”Варианты заданий”, п.а).

```
int main() {
    SetConsoleOutputCP(CP_UTF8);

        int n, l;
        int *a;
        int **graf;

        printf("\nВведите длинну маршрута: ");
        scanf("%d", &l);
        a = init_posl((l + 1));

        printf("\nВведите мощность множества элементов графа: ");
        scanf("%d", &n);
        graf = init_graf(n);

        printf("\nВведите вершину для которой необходимо найти маршруты: ");
        scanf("%d", &a[0]);

        marshruti(a, graf, 1, l, n);

    return 0;
}
```

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.”Варианты заданий”, п.а).

```
int main() {
    SetConsoleOutputCP(CP_UTF8);

    int i, j;
    int n, l;
    int *a;
    int **graf;
    int **r;
    printf("\nВведите длинну маршрута: ");
    scanf("%d", &l);
    a = init_posl((l + 1));

    printf("\nВведите мощность множества элементов графа: ");
    scanf("%d", &n);
    graf = init_graf(n);
    r = (int**)malloc((n + 1) * sizeof(int*));
    // Ввод элементов графа
    for (i = 1; i <= n; i++) {
        // Выделение памяти под хранение строк
        r[i] = (int*)malloc((n + 1) * sizeof(int));
        for (j = 1; j <= n; j++) {
```

```
            r[i][j] = 0;
        }
    }

    int v = 1;

    while (v <= n) {
        a[0] = v;
        marshruti(a, graf, 1, l, n);
        v++;
    }
    j = 1;
    for (i = (j + 1); i <= (n - 1); i++) {
        for (j = 1; j <= n; j++) {
            printf("Между вершинами %d и %d - %d маршрутов.\n", i, j,
r[i][j]);
        }
    }
    return 0;
}
```

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.”Варианты заданий”, п.а).

```
int main() {
    SetConsoleOutputCP(CP_UTF8);

    int l, n;
    int *a;
    int **graf;
    printf("\nВведите длинну маршрута: ");
    scanf("%d", &l);
    a = init_posl((l + 1));

    printf("\nВведите мощность множества элементов графа: ");
    scanf("%d", &n);
    graf = init_graf(n);

    printf("\nВведите 2 вершины между которыми необходимо найти маршруты: ");
    scanf("%d %d", &a[0], &a[l]);

    marshruti(a, graf, 1, l, n);


    return 0;
}
```

7. Написать программу, получающую все простые максимальные цепи, выходящие из заданной вершины графа. Использовать программу для получения всех простые максимальных цепей, выходящих из заданной вершины в графах G1 и G2 (см.”Варианты заданий”, п.а).

```
int main() {
    SetConsoleOutputCP(CP_UTF8);

    int *log_v;
    int *a; // массив маршрута
    int n;// Мощность квадратной матрицы смежности графа
    int **graf; // Матрица графа
    printf("\nВведите мощность множества элементов графа: ");
    scanf("%d", &n);
    graf = init_graf(n);

    int i;
    a = init_posl((n));
    for (i = 0; i < n; i++)
        a[i] = 0;

    printf("\nВведите вершину для которой необходимо найти все простые
максимальные цепи: ");
    scanf("%d", &a[0]);

    log_v = init_posl((n + 1));
    for (i = 0; i <= n; i++)
        log_v[i] = 0;

    log_v[a[0]] = 1;

    all_max_easy_cepi(a, graf, log_v, i, n);

    return 0;
}
```

Вывод: на этой лабораторной работе я изучил основные понятия теории графов, способы задания графов, научиться программно реализовыbak алгоритмы получения и анализа маршрутов в графах.