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 ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №4.4

по дисциплине: Дискретная математика
тема: «Кратчайшие пути во взвешенном орграфе»

Выполнил: ст. группы ПВ-221
Лоёк Никита Викторович

Проверили:
Бондаренко Татьяна Владимировна
Рязанов Юрий Дмитриевич

Белгород 2023 г.

Лабораторная работа № 4.4

Цель работы: изучить алгоритм Дейкстры нахождения кратчайших путей между вершинами взвешенного орграфа, научиться рационально использовать его при решении различных задач.

Задания

1. Изучить алгоритм Дейкстры нахождения кратчайших путей между вершинами взвешенного орграфа.

```
bool IsAll(vector<bool> &v) {
    bool isAll = true;

    for (int i = 0; i < v.size() && isAll; ++i) {
        isAll = isAll && v[i];
    }

    return isAll;
}


vector<int> GetAdjacent(vector<vector<int>> &matrix, int v) {
    vector<int> adjacentVertexes;

    for (int i = 0; i < matrix.size(); ++i) {
        if (matrix[v - 1][i] > 0) {
            adjacentVertexes.push_back(i + 1);
        }
    }

    return adjacentVertexes;

}


vector<int> GetMinWays(vector<vector<int>> &matrix, int vStart, int vEnd) {
    vector<int> t(matrix.size(), -1);
    vector<int> d(matrix.size(), -1);
    vector<bool> v(matrix.size(), false);

    t[vStart - 1] = 0;
    v[vStart - 1] = true;
    d[vStart - 1] = 0;

    int nextYw = 0;
    int nextY = vStart;
    while (!IsAll(v) && vStart != vEnd && nextYw != -1) {
        vector<int> adjacent = GetAdjacent(matrix, vStart);
        nextYw = -1;

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

            if (d[adjacent[i] - 1] == -1) {
                d[adjacent[i] - 1] = d[vStart - 1] + matrix[vStart - 1][adjacent[i] - 1];
                t[adjacent[i] - 1] = vStart;
            } else {
                if (d[adjacent[i] - 1] > d[vStart - 1] + matrix[vStart - 1][adjacent[i] - 1]) {
                    d[adjacent[i] - 1] = d[vStart - 1] + matrix[vStart - 1][adjacent[i] - 1];
                    t[adjacent[i] - 1] = vStart;
                }
            }
        }

        for (int i = 0; i < d.size(); ++i) {
            if (d[i] != -1 && !v[i]) {
                if (nextYw == -1) {
                    nextYw = d[i];
                    nextY = i + 1;
                } else {
```

```
                    if (d[i] < nextYw) {
                        nextYw = d[i];
                        nextY = i + 1;
                    }
                }
            }
        }

        vStart = nextY;
        v[vStart - 1] = true;
    }

    return t;
}
```

2. Используя алгоритм Дейкстры, разработать и реализовать алгоритм решения задачи (см. варианты заданий).

Найти вершину взвешенного орграфа, через которую проходит наибольшее число кратчайших путей от заданной вершины до всех остальных. Вывести кратчайшие пути от заданной вершины до каждой вершины орграфа.

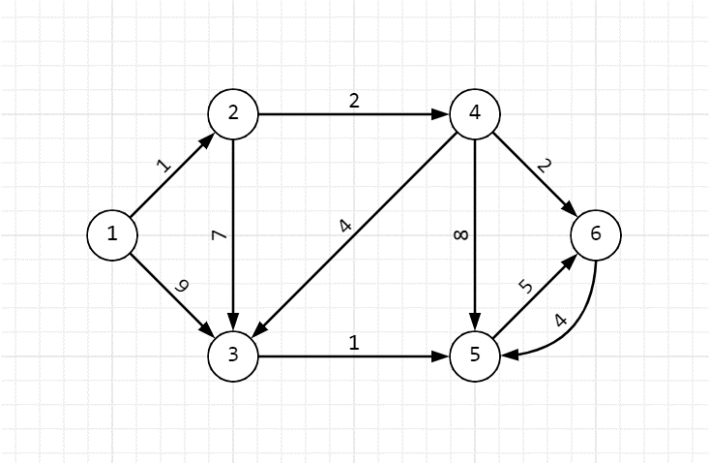
```
bool IsAll(vector<bool> &v) {
    bool isAll = true;

    for (int i = 0; i < v.size() && isAll; ++i) {
        isAll = isAll && v[i];
    }

    return isAll;
}


vector<int> GetMinWays_(vector<vector<int>> &matrix, int vStart) {
    vector<int> t(matrix.size(), -1);
    vector<int> d(matrix.size(), -1);
    vector<bool> v(matrix.size(), false);

    t[vStart - 1] = 0;
    v[vStart - 1] = true;
    d[vStart - 1] = 0;

    int nextYw = 0;
    int nextY = vStart;
    while (!IsAll(v) && nextYw != -1) {
        vector<int> adjacent = GetAdjacent(matrix, vStart);
        nextYw = -1;

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

            if (d[adjacent[i] - 1] == -1) {
                d[adjacent[i] - 1] = d[vStart - 1] + matrix[vStart - 1][adjacent[i] - 1];
                t[adjacent[i] - 1] = vStart;
            } else {
                if (d[adjacent[i] - 1] > d[vStart - 1] + matrix[vStart - 1][adjacent[i] - 1]) {
                    d[adjacent[i] - 1] = d[vStart - 1] + matrix[vStart - 1][adjacent[i] - 1];
                    t[adjacent[i] - 1] = vStart;
                }
            }
        }

        for (int i = 0; i < d.size(); ++i) {
            if (d[i] != -1 && !v[i]) {
                if (nextYw == -1) {
                    nextYw = d[i];
                    nextY = i + 1;
                } else {
                    if (d[i] < nextYw) {
                        nextYw = d[i];
                        nextY = i + 1;
                    }
                }
            }
        }

        vStart = nextY;
        v[vStart - 1] = true;
    }

    return t;
}


vector<int> TreeToSq(vector<int> &tree, int start, int end) {
    vector<int> sq;

    int out = end;
```

```
    while (out != start && out != -1) {
        sq.push_back(out);
        out = tree[out - 1];
    }
    sq.push_back(start);
    reverse(sq.begin(), sq.end());
    if (out == -1) {
        return vector<int>{};
    } else {
        return sq;
    }
}


void GetOftenVertex(vector<vector<int>> &matrix, int vStart) {
    vector<int> v(matrix.size(), 0);
    vector<bool> G(matrix.size(), false);
    G[vStart - 1] = true;

    vector<vector<int>> allSq;

    for (int i = 0; i < matrix.size(); ++i) {
        if (G[i] == false) {
            vector<int> tree = GetMinWays_(matrix, vStart);

            for (int j = 0; j < matrix.size(); ++j) {
                if (G[j] == false && tree[j] != -1) {
                    vector<int> sq = TreeToSq(tree, vStart, j + 1);
                    G[j] = true;

                    allSq.push_back(sq);

                    for (int k = 0; k < sq.size(); ++k) {
                        v[sq[k] - 1]++;
                    }
                }
            }
        }
    }

    int maxV;
    int maxSum = -1;

    for (int i = 0; i < v.size(); ++i) {
        if (vStart != i + 1) {
            if (v[i] > maxSum) {
                maxSum = v[i];
                maxV = i + 1;
            }
        }
    }

    if (allSq.empty()) {
        cout << "Not vertex" << endl;
    } else {
        cout << "Vertex: " << maxV << endl;
    }

    outputVertex(allSq);
}
```

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

Тест 1:

Изначальный Граф:



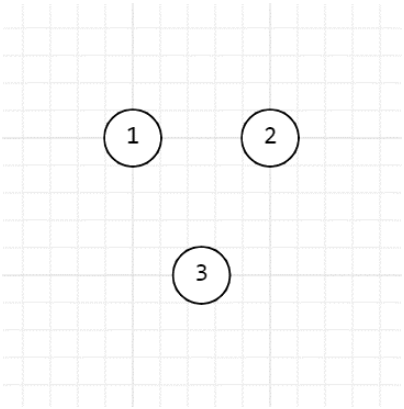
Результат работы программы:

Для начальной вершины 1:

```
0   1   9   0   0   0
0   0   7   2   0   0
0   0   0   0   1   0
0   0   4   0   8   2
0   0   0   0   0   5
0   0   0   0   4   0
Vertex: 2
1 2
1 2 4 3
1 2 4
1 2 4 3 5
1 2 4 6
```

Тест 2:

Изначальный Граф:



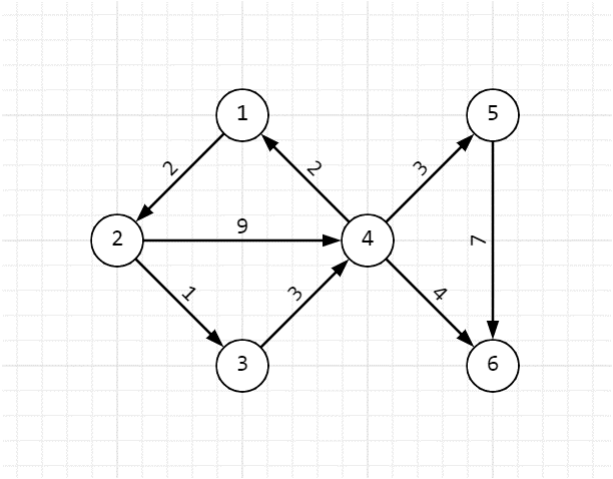
Результат работы программы:

Для начальной вершины 1:

```
0   0   0
0   0   0
0   0   0
Not vertex
```

Тест 3:

Изначальный Граф:



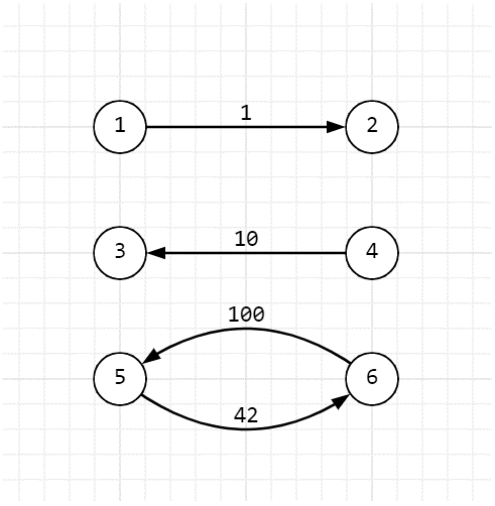
Результат работы программы:

Для начальной вершины 4:

```
0   2   0   0   0   0
0   0   1   9   0   0
0   0   0   3   0   0
2   0   0   0   3   4
0   0   0   0   0   7
0   0   0   0   0   0
Vertex: 1
4 1
4 1 2
4 1 2 3
4 5
4 6
```

Тест 4:

Изначальный Граф:



Результат работы программы:

Для начальной вершины 5:

```
0   1   0   0   0   0
0   0   0   0   0   0
0   0   0   0   0   0
0   0   10  0   0   0
0   0   0   0   0   42
0   0   0   0   100 0
Vertex: 6
5 6
```

Вывод

Вывод: в ходе работы я изучил алгоритм Дейкстры для нахождения кратчайших путей между вершинами взвешенного орграфа, научился рационально использовать его при решении различных задач.