

Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительной техники»

### ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проектированию

по курсу «Логика и основы алгоритмизации в инженерных  
задачах»

на тему «Реализация алгоритма Форда-Беллмана»

29.11.24  
сдано  
ФЗ

Выполнил:

Студент группы 22ВВВ3

Перкин П. О.

Проверил:

Юрова О. В.

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ 

«    »      20    

## ЗАДАНИЕ

на курсовое проектирование по курсу

Логика и основы алгоритмизации в инженерных задачах  
Студенту Теркин Павлу Олеговичу Группа 22ВВ12  
Тема проекта Реализация алгоритма Форда-Беллмана

Исходные данные (технические требования) на проектирование

Разработка алгоритмов и программного обеспечения в соответствии с данным заданием курсового проекта.

Пояснительная записка должна содержать:

1. Постановку задачи;
2. Теоретическую часть задания;
3. Описание алгоритма поставленной задачи;
4. Пример ручного расчета задачи и вычисления;
5. Описание самой программы;
6. Теста;
7. Список литературы;
8. Листы программы;
9. Результаты работы программы

## Объем работы по курсу

### 1. Расчетная часть

Ручной расчёт работы алгоритма.

### 2. Графическая часть

Схема алгоритма в формате блок-схема.

### 3. Экспериментальная часть

Тестирование программы;  
Результаты работы программы на тестовых  
задачах.

### Срок выполнения проекта по разделам

- 1 Исследование теоретической части курсового
- 2 Разработка алгоритма программы
- 3 Разработка программы
- 4 Тестирование и завершение разработки программы
- 5 Составление пояснительной записки
- 6
- 7
- 8

Дата выдачи задания " 20 " 09. 23

Дата защиты проекта " " "

Руководитель Юрова О.В.

Задание получил " " 20 г.

Студент Беркин Павел Олегович

## Содержание

|                                      |    |
|--------------------------------------|----|
| Введение.....                        | 5  |
| 1.Постановка задачи.....             | 6  |
| 2.Теоретическая часть задания .....  | 7  |
| 3.Описание алгоритма программы ..... | 8  |
| 4.Описание программы.....            | 9  |
| 5.Тестирование .....                 | 11 |
| Заключение.....                      | 14 |
| Список литературы .....              | 15 |
| Листинг программы .....              | 16 |

Алгоритм Форда-Беллмана – это один из фундаментальных алгоритмов в теории графов и комбинаторной оптимизации, который применяется для решения задач кратчайших путей во взвешенных ориентированных графах. Назван в честь американских математиков Лестера Форда и Ричарда Беллмана, этот алгоритм является универсальным и мощным инструментом, который нашел широкое применение в практических задачах, включая транспортные сети, маршрутизацию компьютерных сетей, планирование производственных процессов и многие другие.

Основная цель алгоритма Форда-Беллмана – найти кратчайшие расстояния от одной изначально выбранной вершины графа до всех остальных вершин. Это делает его важным инструментом в решении ряда практических задач, связанных с поиском оптимальных путей в различных сценариях.

Одной из ключевых особенностей алгоритма Форда-Беллмана является его способность работать с графами, содержащими рёбра с отрицательными весами. Это делает его универсальным инструментом для решения задач, в которых могут возникать отрицательные веса рёбер, что отличает его от некоторых других алгоритмов поиска кратчайших путей.

Алгоритм Форда-Беллмана заслуживает внимания, поскольку он представляет собой мощный инструмент анализа и оптимизации в различных областях, и его понимание может принести значительную выгоду при решении практических задач.

В качестве среды разработки мною была выбрана среда VisualStudio, язык программирования - C++.

Целью данной работы является подробное рассмотрение алгоритма Форда-Беллмана, его применение в контексте теории графов, а также создание программы на языке C++ с использованием среды разработки VisualStudio для демонстрации работы алгоритма.

## **1.Постановка задачи**

Разработать программу для работы с графами, представленными в виде матрицы смежности. Программа должна обеспечивать генерацию случайных графов, поиск минимальных маршрутов.

Разработать программу, предназначенную для генерации и обработки матрицы смежности с помощью алгоритма Форда-Беллмана. Программа должна обладать разносторонним функционалом, включающим в себя генерацию случайных графов, поиск кратчайших путей, меню, загрузку в файл матрицы и загрузку матрицы из файла.

## 2. Теоретическая часть задания

Алгоритм Форда-Беллмана – это один из фундаментальных алгоритмов в теории графов, используемый для поиска кратчайших путей между всеми парами вершин в графе. Назван в честь математиков Лестера Форда и Ричарда Беллмана, внесших вклад в его разработку в середине 20 века.

Как и в алгоритме Дейкстры, создаётся таблица, в которой записывается расстояние до каждой вершины и предыдущая вершина. Расстояния для всех вершин, кроме исходной, инициализируются значением бесконечности. На начальном этапе, расстояния для всех вершин, кроме стартовой, устанавливаются на бесконечность.

Основной процесс алгоритма включает в себя итерации, где на каждом шаге рассматриваются все рёбра графа. Обновление расстояний осуществляется путем проверки возможности уменьшения расстояния до конечной вершины при использовании текущего расстояния и веса ребра. В дополнительной итерации проверяется наличие циклов отрицательного веса.

По завершении алгоритма, мы получаем таблицу с кратчайшими путями от стартовой вершины до всех остальных. Применительно к примеру графа, процесс алгоритма позволяет нам выявить оптимальные маршруты и преодолеть ограничения, такие как отрицательные веса на рёбрах. Алгоритм Форда-Беллмана, с учетом своей универсальности, остается востребованным в различных областях, где необходимо эффективное управление путями в графах.

Преимущества:

- Способен работать с графами, содержащими рёбра с отрицательными весами.
- Прост в реализации и понимании.

Ограничения:

- Неэффективен для больших графов из-за квадратичной сложности по времени.
- Не обрабатывает циклы отрицательного веса корректно.

### 3.Описание алгоритма программы

Для начала давайте определим, что такое матрица смежности. Матрица смежности — это способ представления графа в виде квадратной матрицы, в которой строки и столбцы соответствуют вершинам графа, а элементы матрицы указывают наличие (или отсутствие) ребра между вершинами.

Далее, алгоритм Форда-Беллмана решает проблему кратчайшего пути во взвешенном графе, где веса рёбер могут быть отрицательными. Алгоритм можно использовать для нахождения кратчайших путей из одной выбранной вершины до всех остальных.

Код программы на псевдокоде для генерации и обработки матрицы смежности с помощью алгоритма Форда-Беллмана может выглядеть примерно так:

```
Пока i меньше size - 1
    Для каждой вершины u от 0 до size - 1
        Для каждой вершины v от 0 до size - 1
            Если matrix[u][v] не равно INF и distance[u] не равно INF и
distance[u] + matrix[u][v] меньше distance[v]
                distance[v] = distance[u] + matrix[u][v]
```

## 4. Описание программы

Программа представляет собой консольное приложение, которое позволяет пользователю взаимодействовать с графами, используя матрицы смежности для представления графов. Пользователь может генерировать случайные графы, сохранять их в файлы, загружать из файлов и вычислять кратчайшие расстояния между вершинами с помощью алгоритма Беллмана-Форда. Программа также обрабатывает случаи некорректного ввода и предоставляет удобный интерфейс для взаимодействия.

Содержит такие функции, как:

`generateRandomMatrix(int size)`: Эта функция генерирует случайную матрицу смежности для графа с заданным количеством вершин. На главной диагонали матрицы находятся нули, так как расстояние от вершины до самой себя равно нулю. Остальные элементы матрицы заполняются случайными весами от 1 до 10. Если два узла не связаны, то соответствующий элемент матрицы будет равен INF.

`saveMatrixToFile(const vector<vector<int>>& matrix, const string& filename)`: Эта функция сохраняет матрицу смежности в файл. Сначала записывается размер матрицы, затем все элементы матрицы построчно. Это позволяет пользователю сохранять состояние графа для последующего использования.

`loadMatrixFromFile(const string& filename)`: Эта функция загружает матрицу смежности из файла. Сначала считывается размер матрицы, затем считываются все элементы матрицы. Это удобно для восстановления графа из сохранённого состояния.

`printMatrix(const vector<vector<int>>& matrix)`: Эта функция выводит матрицу смежности на экран в удобочитаемом формате. Если элемент равен INF, она выводит "INF", иначе показывает значение элемента. Это помогает пользователю визуально оценить структуру графа.

`bellmanFord(vector<vector<int>>& matrix, int size, int source)`: Эта функция реализует алгоритм Беллмана-Форда для нахождения кратчайших расстояний от заданной вершины (source) до всех остальных вершин в графе. Она также

проверяет наличие отрицательных циклов в графе, что важно для анализа его свойств.

`main()`: Основная функция программы. Она содержит меню, которое предоставляет пользователю возможность выбирать действия: случайная генерация матрицы, сохранение и загрузка матрицы из файла, выполнение алгоритма Беллмана-Форда и выход из программы. Функция обрабатывает ввод пользователя и выполняет соответствующие функции.

## 5.Тестирование

Среда разработки Microsoft Visual Studio 2022 предоставляет все средства, необходимые при разработке и отладке программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с вводом данных, изменением дизайна выводимых данных, алгоритмом программы, взаимодействием функций.

Ниже продемонстрирован результат тестирования программы при вводе пользователем различных количеств вершин. (Рисунок 1-8)

```
Меню:
1. Случайная генерация матрицы
2. Сохранить матрицу в файл
3. Загрузить матрицу из файла
4. Выполнить алгоритм Беллмана-Форда
5. Выход
Выберите действие: 1
Введите размер матрицы смежности: 3
Матрица смежности сгенерирована.
Матрица смежности:
0 10 2
10 0 6
2 6 0
```

Рисунок 1 - тестирование генерации случайной матрицы смежности размером 3 на 3.

```
Меню:
1. Случайная генерация матрицы
2. Сохранить матрицу в файл
3. Загрузить матрицу из файла
4. Выполнить алгоритм Беллмана-Форда
5. Выход
Выберите действие: 4
Расстояние от вершины 0 до 0 = 0
Расстояние от вершины 0 до 1 = 8
Расстояние от вершины 0 до 2 = 2
Расстояние от вершины 1 до 0 = 8
Расстояние от вершины 1 до 1 = 0
Расстояние от вершины 1 до 2 = 6
Расстояние от вершины 2 до 0 = 2
Расстояние от вершины 2 до 1 = 6
Расстояние от вершины 2 до 2 = 0
```

Рисунок 2 – тестирование алгоритма Форда-Беллмана на сгенерированной матрице.

```
Меню:
1. Случайная генерация матрицы
2. Сохранить матрицу в файл
3. Загрузить матрицу из файла
4. Выполнить алгоритм Беллмана-Форда
5. Выход
Выберите действие: 2
Матрица успешно сохранена в файл matrix.txt
```

Рисунок 3 – тестирование записи сгенерированной матрицы в файл.

```
Меню:
1. Случайная генерация матрицы
2. Сохранить матрицу в файл
3. Загрузить матрицу из файла
4. Выполнить алгоритм Беллмана-Форда
5. Выход
Выберите действие: 3
Матрица успешно загружена из файла matrix.txt
Матрица смежности:
0 10 2
10 0 6
2 6 0
```

Рисунок 4 – тестирование загрузки матрицы из файла.

```
Меню:
1. Случайная генерация матрицы
2. Сохранить матрицу в файл
3. Загрузить матрицу из файла
4. Выполнить алгоритм Беллмана-Форда
5. Выход
Выберите действие: 5
Выход из программы.

D:\ж\лоа\kurs4\x64\Debug\kurs4.exe (процесс 15160) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно: _
```

Рисунок 5 – тестирование кнопки «Выход».

```
Меню:
1. Случайная генерация матрицы
2. Сохранить матрицу в файл
3. Загрузить матрицу из файла
4. Выполнить алгоритм Беллмана-Форда
5. Выход
Выберите действие: 4
Матрица пуста! Сначала сгенерируйте или загрузите матрицу.
```

Рисунок 6 – тестирование алгоритма без сгенерированной или загруженной матрицы.

```
Меню:
1. Случайная генерация матрицы
2. Сохранить матрицу в файл
3. Загрузить матрицу из файла
4. Выполнить алгоритм Беллмана-Форда
5. Выход
Выберите действие: текст
Некорректный ввод! Пожалуйста, введите число от 1 до 5.
Выберите действие: _
```

Рисунок 7 – тестирование ввода текста при выборе в меню.

```
Меню:
1. Случайная генерация матрицы
2. Сохранить матрицу в файл
3. Загрузить матрицу из файла
4. Выполнить алгоритм Беллмана-Форда
5. Выход
Выберите действие: 1
Введите размер матрицы смежности: текст
Некорректный размер матрицы! Попробуйте снова.
Введите размер матрицы смежности: _
```

Рисунок 8 – тестирование ввода текста при вводе размера матрицы.

Таблица 1 – Описание поведения программы при тестировании.

| Описание теста                      | Ожидаемый результат                                                                  | Полученный результат |
|-------------------------------------|--------------------------------------------------------------------------------------|----------------------|
| Запуск программы                    | Вывод меню                                                                           | Верно                |
| Выбор случайной генерации матрицы   | Случайная генерация матрицы размером заданным пользователем и вывод матрицы на экран | Верно                |
| Реализация алгоритма Форда-Беллмана | Вывод кратчайшего расстояния от вершины до вершины                                   | Верно                |
| Запись в файл                       | Запись сгенерированной случайной матрицы в файл matrix.txt                           | Верно                |
| Загрузка из файла                   | Загрузка матрицы из файла matrix.txt и вывод загруженной матрицы на экран.           | Верно                |
| Выход из программы                  | Вывод надписи «Выход из программы» и завершение работы программы с кодом 0.          | Верно                |

В результате тестирования было выявлено, что программа успешно проверяет данные на соответствие необходимым требованиям.

## **Заключение**

В результате выполнения данной курсовой работы была разработана программа, реализующая алгоритм Форда-Беллмана в среде Visual Studio.

В процессе работы выявлены некоторые аспекты, требующие внимания для дальнейшего совершенствования программы. Расширение функциональности программы представляет собой важное направление для будущего развития, включая добавление дополнительных алгоритмов обработки графов.

Курсовая работа стала важным этапом в обучении, позволяя углубить понимание принципов работы графов и алгоритмов, а также приобрести практические навыки в области разработки программ, связанных с теорией графов. Полученные знания и опыт представляют собой основу для дальнейших исследований и развития в данной области.

## Список литературы

1. Форд, Л. Р.; Беллман, Р. (1958). "On a Routing Problem". Quarterly of Applied Mathematics. 16 (1): 85–88.
2. Кормен, Т. Х.; Лейзерсон, Ч. Э.; Ривест, Р. Л.; Штайн, К. (2005). Алгоритмы. Построение и анализ (2-е издание). Москва: Вильямс.
3. Кнут, Д. Э. (1998). Искусство программирования. Том 1. Основные алгоритмы (3-е издание). Москва: Вильямс.
4. Куликов, В. А. (2009). Алгоритмы на графах. Москва: Солон-Пресс.
5. Кормен, Т. Х.; Лейзерсон, Ч. Э.; Ривест, Р. Л.; Штайн, К. (2017). Изучаем алгоритмы: Иллюстрированное пособие. Москва: ДМК Пресс.

## Листинг программы

```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <fstream>

using namespace std;

const int INF = 9999;

vector<vector<int>> generateRandomMatrix(int size) {
    vector<vector<int>> matrix(size, vector<int>(size, INF));

    srand(time(NULL));

    for (int i = 0; i < size; i++) {
        for (int j = i; j < size; j++) {
            if (i == j) {
                matrix[i][j] = 0;
            }
            else {
                int weight = rand() % 10 + 1;
                matrix[i][j] = weight;
                matrix[j][i] = weight;
            }
        }
    }

    return matrix;
}

void saveMatrixToFile(const vector<vector<int>>& matrix, const string&
filename) {
    ofstream outFile(filename);
    if (outFile.is_open()) {
        int size = matrix.size();
        outFile << size << endl;
        for (const auto& row : matrix) {
            for (const auto& val : row) {
                outFile << val << " ";
            }
            outFile << endl;
        }
        outFile.close();
        cout << "Матрица успешно сохранена в файл " << filename << endl;
    }
    else {
        cout << "Не удалось открыть файл для записи." << endl;
    }
}

vector<vector<int>> loadMatrixFromFile(const string& filename) {
    ifstream inFile(filename);
```

```

vector<vector<int>>> matrix;

if (inFile.is_open()) {
    int size;
    inFile >> size;
    matrix.resize(size, vector<int>(size));

    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            inFile >> matrix[i][j];
        }
    }
    inFile.close();
    cout << "Матрица успешно загружена из файла " << filename << endl;
}
else {
    cout << "Не удалось открыть файл для чтения." << endl;
}

return matrix;
}

void printMatrix(const vector<vector<int>>>& matrix) {
    cout << "Матрица смежности:" << endl;
    for (const auto& row : matrix) {
        for (const auto& val : row) {
            if (val == INF) {
                cout << "INF ";
            }
            else {
                cout << val << " ";
            }
        }
        cout << endl;
    }
}

void bellmanFord(vector<vector<int>>& matrix, int size, int source) {
    vector<int> distance(size, INF);
    distance[source] = 0;

    for (int i = 0; i < size - 1; i++) {
        for (int u = 0; u < size; u++) {
            for (int v = 0; v < size; v++) {
                if (matrix[u][v] != INF && distance[u] != INF && distance[u]
+ matrix[u][v] < distance[v]) {
                    distance[v] = distance[u] + matrix[u][v];
                }
            }
        }
    }

    for (int u = 0; u < size; u++) {
        for (int v = 0; v < size; v++) {

```

```

        if (matrix[u][v] != INF && distance[u] != INF && distance[u] +
matrix[u][v] < distance[v]) {
            cout << "Граф содержит отрицательный цикл" << endl;
            return;
        }
    }
}

for (int i = 0; i < size; i++) {
    cout << "Расстояние от вершины " << source << " до " << i << " = "
<< distance[i] << endl;
}
}

int main() {
    setlocale(LC_ALL, "Russian");
    int size, choice;
    vector<vector<int>> matrix;

    do {
        cout << "\nМеню:\n";
        cout << "1. Случайная генерация матрицы\n";
        cout << "2. Сохранить матрицу в файл\n";
        cout << "3. Загрузить матрицу из файла\n";
        cout << "4. Выполнить алгоритм Беллмана-Форда\n";
        cout << "5. Выход\n";
        cout << "Выберите действие: ";

        while (!(cin >> choice)) {
            cout << "Некорректный ввод! Пожалуйста, введите число от 1 до
5." << endl;

            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Выберите действие: ";
        }

        switch (choice) {
        case 1:
            cout << "Введите размер матрицы смежности: ";
            while (!(cin >> size) || size <= 0) {
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                cout << "Некорректный размер матрицы! Попробуйте снова." <<
endl;

                cout << "Введите размер матрицы смежности: ";
            }
            matrix = generateRandomMatrix(size);
            cout << "Матрица смежности сгенерирована." << endl;
            printMatrix(matrix);
            break;

        case 2:
            if (matrix.empty()) {
                cout << "Матрица пуста! Сначала сгенерируйте или загрузите
матрицу." << endl;

```

```

        }
        else {
            saveMatrixToFile(matrix, "matrix.txt");
        }
        break;

    case 3:
        matrix = loadMatrixFromFile("matrix.txt");
        printMatrix(matrix);
        break;

    case 4:
        if (matrix.empty()) {
            cout << "Матрица пуста! Сначала сгенерируйте или загрузите
матрицу." << endl;
        }
        else {
            for (int i = 0; i < matrix.size(); i++) {
                bellmanFord(matrix, matrix.size(), i);
            }
        }
        break;

    case 5:
        cout << "Выход из программы." << endl;
        break;

    default:
        cout << "Некорректный выбор! Попробуйте снова." << endl;
        break;
    }
} while (choice != 5);

return 0;
}

```