

Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К курсовой работе

По курсу «Логика и основы алгоритмизации в инженерных задачах»

На тему «Реализация алгоритма Прима»

Выполнил студент группы 22ВВВ3 (22ВВП2):

Городничев М.И.

16.12.23
отлично
фв

Приняли:

Юрова О.В.

Акифьев И.В.

Пенза 2023

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

«___» _____ 20__

ЗАДАНИЕ

на курсовое проектирование по курсу

„Логика и основы алгоритмизации в инженерных задачах”
Студенту Горозничеву Максиму Игоревичу Группа 22 ВВВЗ (22 ВВТ2)
Тема проекта Реализация алгоритма Кристи

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 расчёта задачи и вычислений (на небольшом участке работы алгоритма);
5. Описание самой программы;
6. Тесты;
7. Список литературы;
8. Листинг программы;
9. Результаты работы программы;

Объем работы по курсу

1. Расчетная часть

Пунктной расчёт работы алгоритма

2. Графическая часть

Схема алгоритма в формате блок-схем

3. Экспериментальная часть

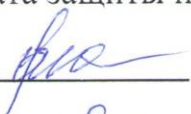
Тестирование программы;
Результаты работы программы на тестовых данных

Срок выполнения проекта по разделам

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

Дата выдачи задания “ 6 ” сентября

Дата защиты проекта “ ”

Руководитель Юрова О.В. 

Задание получил “ 6 ” сентября 20 23 г.

Студент Горюничев Максим Игоревич Горю

Содержание

Реферат.....	5
Введение	6
1. Постановка задачи.....	7
2. Теоретическая часть задания	8
3. Описание алгоритма программы.....	9
4. Описание программы	10
5. Тестирование	14
6. Ручной расчёт задачи	16
Заключение	17
Список литературы	18
Приложение А. Листинг программы.....	19

Реферат

Отчёт 22 стр., 11 рисунков, 1 таблица, 1 приложение.

ГРАФ, ТЕОРИЯ ГРАФОВ, АЛГОРИТМ ПРИМА.

Цель исследования – разработка программы, способная эффективно выделять минимальное остовное дерево взвешенного связного неориентированного графа с использованием алгоритма Прима.

В работе рассмотрены основные шаги алгоритма Прима для построения минимального остовного дерева взвешенного графа. Установлено, что с помощью данного алгоритма можно выделить минимальное остовное дерево независимо от степени связности и весов рёбер исходного графа.

Введение

Алгоритм Прима, также известный как алгоритм построения минимального остовного дерева, применяется для поиска минимального остовного дерева в связном взвешенном неориентированном графе. Основная цель алгоритма состоит в том, чтобы построить дерево, содержащее все вершины графа, при этом минимизируя общий вес рёбер.

В отличие от алгоритма поиска в глубину или поиска в ширину, алгоритм Прима работает путём постепенного построения остовного дерева из начальной вершины. Он постепенно добавляет рёбра, соединяющие эту вершину с другими вершинами графа. Это происходит путём выбора наименьшего по весу ребра, связывающего уже выбранные вершины с еще не выбранными.

Алгоритм Прима позволяет находить минимальные остовные деревья в графах, которые могут быть представлены исходя из локальной информации, то есть, когда вся сеть или граф не доступны целиком, и информация о структуре графа представлена постепенно или частично.

Однако стоит учесть, что алгоритм Прима требует наличия всех весов рёбер в графе и может быть менее эффективен в случае больших объёмов данных или графов с большим количеством вершин и рёбер. Тем не менее, он остаётся важным инструментом для нахождения минимальных остовных деревьев в различных прикладных задачах.

В качестве среды разработки мною была выбрана среда Microsoft Visual Studio 2019, язык программирования – Си.

Целью данной курсовой работы является разработка программы на языке Си, который является широко используемым. Именно с его помощью в данном курсовом проекте реализуется алгоритм Прима, осуществляющий построение минимального остовного дерева.

1. Постановка задачи

Требуется разработать программу, которая выделит минимальное остовное дерево для неориентированного взвешенного графа, используя алгоритм Прима.

Исходный граф в программе должен задаваться матрицей весов рёбер, причем при генерации данных должны быть предусмотрены граничные условия. Программа должна работать так, чтобы пользователь вводил количество вершин для генерации матрицы взвешенного графа. После обработки этих данных на экран должна выводиться матрица взвешенного графа и минимальное остовное дерево, полученное алгоритмом Прима. Необходимо предусмотреть различные исходы, такие как случаи отсутствия связности в графе или некорректного ввода данных, чтобы программа не выдавала ошибок и работала правильно. Устройство ввода – клавиатура и мышь.

2. Теоретическая часть задания

Граф G , изображенный на рисунке 1, состоит из множества вершин X_1, X_2, \dots, X_n и множества ребер, соединяющих определенные вершины. Ребра в этом графе имеют веса, показывающие стоимость прохождения между вершинами. Граф с такими ребрами называется взвешенным.

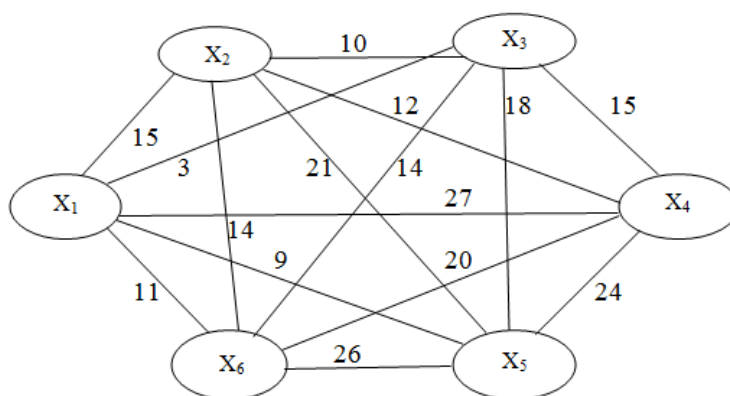


Рисунок 1 – Пример взвешенного графа

Представление графа в виде матрицы смежности позволяет хранить информацию о ребрах графа в квадратной матрице, где присутствие пути из одной вершины в другую обозначается числом, представляющим вес ребра, а отсутствие пути – специальным обозначением, например, нулём.

Существует множество алгоритмов для работы с взвешенными графами, и одним из них является алгоритм Прима. Этот алгоритм направлен на построение минимального остовного дерева в графе и на работу с неориентированными графами (для ориентированных графов он не подойдёт). Он начинается из начальной вершины и последующим добавлением новых ребер и вершин в дерево, выбирая ребро минимального веса, связывающее уже существующее дерево с вершиной, не включенной в дерево. Процесс продолжается до тех пор, пока все вершины не будут включены в остовное дерево.

3. Описание алгоритма программы

Цель алгоритма Прима в графе заключается в построении минимального остоного дерева путем пошагового добавления ребер с наименьшим весом. Начиная с начальной вершины, мы постепенно расширяем остоное дерево, выбирая ребро минимального веса, которое связывает уже выбранные вершины с вершиной из оставшихся. Этот процесс продолжается, пока все вершины не будут включены в остоное дерево или пока не будет достигнут критерий остановки. Ниже представлен псевдокод:

функция Алгоритм Прима (граф[`MAX_SIZE`][`MAX_SIZE`], размер, родитель[`MAX_SIZE`])

```
минВес[MAX_SIZE]  
посещено[MAX_SIZE]  
  
для i = 0 пока i < размер  
    минВес[i] = INT_MAX  
    посещено[i] = false  
    родитель[i] = -1  
  
минВес[0] = 0  
  
для count = 0 пока count < размер - 1  
    минВершина = -1  
    для v = 0 пока v < размер  
        если не посещено[v] и (минВершина == -1 или  
минВес[v] < минВес[минВершина])  
            минВершина = v  
  
    посещено[минВершина] = true  
  
    для v = 0 пока v < размер  
        если граф[минВершина][v] != 0 и не посещено[v] и  
граф[минВершина][v] < минВес[v]  
            родитель[v] = минВершина  
            минВес[v] = граф[минВершина][v]  
  
вывод("\nМинимальное остоное дерево:\n")  
для i = 1 пока i < размер  
    если родитель[i] != -1  
        вывод("Ребро: " + родитель[i] + " - " + i + ", Вес:  
" + граф[i][родитель[i]])  
    иначе  
        вывод("Ребро: " + findUnconnectedNode(родитель,  
размер) + " - " + i + ", не существует")
```

4. Описание программы

Для написания данной программы использован язык программирования Си. Язык программирования Си - универсальный язык программирования, который завоевал особую популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней.

Проект был создан в виде консольного приложения Win32 (Visual C++).

Данная программа является многомодульной, поскольку состоит из нескольких функций: `saveGraphToFile`, `savePrimResultToFile`, `fileExists`, `loadGraphFromFile`, `fillUndirectedGraphRandom`, `printGraph`, `findUnconnectedNode`, `primAlgorithm`, `main(void)`.

Работа программы начинается с вывода общей информации о курсовой работе (рис. 2).

```
printf("Курсовая работа\n");  
printf("По курсу \"Логика и основы алгоритмизации в инженерных задачах\"\n");  
printf("На тему \"Реализация алгоритма Прима\"\n");  
printf("Выполнил: Городничев Максим Игоревич, группа: 22BVB3 (22BVB2)\n\n");
```

Далее выводятся основные действия программы, которые пользователь может выбрать (рис. 3).

```
printf("Выберите действие:\n");  
printf("1. Загрузить неориентированный граф из файла\n");  
printf("2. Заполнить неориентированный граф компьютером\n");  
printf("0. Выйти из программы\n");  
printf("Выбор: ");  
scanf("%d", &choice);
```

Если пользователь выбрал первый пункт (рис. 4), то сначала программа просит ввести название файла. Затем она загружает этот файл (вызывая функцию `loadGraphFromFile`), выводит граф на экран (вызывая функцию `printGraph`), выполняет алгоритм Прима и выводит минимальное остовное

дерево на экран (вызывая функцию `primAlgorithm`), сохраняет граф и результат в файлы (вызывая функции `saveGraphToFile` и `savePrimResultToFile`).

```
case 1:
    printf("\nВведите имя файла для загрузки: ");
    scanf("%s", filename);
    if (!fileExists(filename)) {
        printf("\nВведён несуществующий файл\n\n");
        break;
    }
    loadGraphFromFile(graph, &size, filename);
    printGraph(graph, size);
    primAlgorithm(graph, size, parent);
    saveGraphToFile(graph, size, "graph.txt");
    savePrimResultToFile(parent, graph, size, "result.txt");
    break;
```

Если пользователь выбрал второй пункт (рис. 5), то сначала программа просит ввести размер графа. Затем она заполняет этот граф случайными числами (вызывая функцию `fillUndirectedGraphRandom`), выводит граф на экран (вызывая функцию `printGraph`), выполняет алгоритм Прима и выводит минимальное остовное дерево на экран (вызывая функцию `primAlgorithm`), сохраняет граф и результат в файлы (вызывая функции `saveGraphToFile` и `savePrimResultToFile`).

```
case 2:
    printf("\nВведите размер неориентированного графа: ");
    scanf("%d", &size);
    if (size <= 0) {
        printf("\nВведён неверный размер матрицы\n\n");
        break;
    }
    fillUndirectedGraphRandom(graph, size);
    printGraph(graph, size);
    primAlgorithm(graph, size, parent);
    saveGraphToFile(graph, size, "graph.txt");
    savePrimResultToFile(parent, graph, size, "result.txt");
    break;
```

Если пользователь выбрал пункт «0», то программа завершит свою работу (рис. 6).

Ниже представлена работа программы:

Курсовая работа
По курсу "Логика и основы алгоритмизации в инженерных задачах"
На тему "Реализация алгоритма Прима"
Выполнил: Городничев Максим Игоревич, группа: 22ВВВ3 (22ВВП2)

Рисунок 2 – Общая информация

```
Выберите действие:
1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф компьютером
0. Выйти из программы
Выбор: _
```

Рисунок 3 – Основные действия

```
Выберите действие:
1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф компьютером
0. Выйти из программы
Выбор: 1

Введите имя файла для загрузки: graph.txt

Граф:
  0   0  81  34  14   0   0
  0   0   0   0  83   0   0
81   0   0   0   0  19  70
34   0   0   0  16   0  39
14  83   0  16   0  58   8
  0   0  19   0  58   0   0
  0   0  70  39   8   0   0

Минимальное остовное дерево:
Ребро: 4 - 1, Вес: 83
Ребро: 5 - 2, Вес: 19
Ребро: 4 - 3, Вес: 16
Ребро: 0 - 4, Вес: 14
Ребро: 4 - 5, Вес: 58
Ребро: 4 - 6, Вес: 8

Граф успешно сохранен в файл graph.txt

Результат успешно сохранен в файл result.txt
```

Рисунок 4 – Загрузка графа из файла

```

Выберите действие:
1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф компьютером
0. Выйти из программы
Выбор: 2

Введите размер неориентированного графа: 7

Граф:
  0  0  32  36  14  0  42
  0  0  75  11  88  0  22
32  75  0  25  3  21  0
36  11  25  0  0  0  0
14  88  3  0  0  2  0
  0  0  21  0  2  0  0
42  22  0  0  0  0  0

Минимальное остовное дерево:
Ребро: 3 - 1, Вес: 11
Ребро: 4 - 2, Вес: 3
Ребро: 2 - 3, Вес: 25
Ребро: 0 - 4, Вес: 14
Ребро: 4 - 5, Вес: 2
Ребро: 1 - 6, Вес: 22

Граф успешно сохранен в файл graph.txt

Результат успешно сохранен в файл result.txt

```

Рисунок 5 – Заполнение графа компьютером

```

Выберите действие:
1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф компьютером
0. Выйти из программы
Выбор: 0

C:\Users\Admin\source\repos\KursovaiRabota2\Debug\KursovaiRabota2.exe (процесс 13552) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно...

```

Рисунок 6 – Завершение работы программы

5. Тестирование

Среда разработки Microsoft Visual Studio 2019 предоставляет все средства, необходимые при разработке и отладке многомодульной программы. Тестирование проводилось в процессе разработки, после завершения написания программы. В ходе тестирования проблем не было выявлено.

Таблица 1 – Описание поведения программы при тестировании

Описание теста	Ожидаемый результат	Полученный результат
Проверка запуска программы	Вывод общей информации и основных действий	Верно
Проверка на неправильный выбор основного действия	Программа должна оповестить пользователя об ошибке, предложить снова варианты	Верно
Проверка на введение несуществующего файла	Программа должна оповестить пользователя об ошибке, предложить снова варианты	Верно
Проверка на введение неправильного размера графа	Программа должна оповестить пользователя об ошибке, предложить снова варианты	Верно

Ниже приведены результаты тестов:

```
Курсовая работа
По курсу "Логика и основы алгоритмизации в инженерных задачах"
На тему "Реализация алгоритма Прима"
Выполнил: Городничев Максим Игоревич, группа: 22ВВВЗ (22ВВП2)

Выберите действие:
1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф компьютером
0. Выйти из программы
Выбор: _
```

Рисунок 7 – Проверка запуска программы

```
Выберите действие:
1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф компьютером
0. Выйти из программы
Выбор: 10

Сделан неверный выбор. Пожалуйста, выберите снова.
```

Рисунок 8 – Проверка на неправильный выбор основного действия

```
Выберите действие:
1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф компьютером
0. Выйти из программы
Выбор: 1

Введите имя файла для загрузки: hgjef.txt

Введён несуществующий файл
```

Рисунок 9 – Проверка на введение несуществующего файла

```
Выберите действие:
1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф компьютером
0. Выйти из программы
Выбор: 2

Введите размер неориентированного графа: -1

Введён неверный размер графа
```

Рисунок 10 – Проверка на введение неправильного размера графа

6. Ручной расчёт задачи

Проведём проверку программы посредством ручных вычислений на примере рисунка 11:

Граф:					
0	30	32	97	61	
30	0	66	89	53	
32	66	0	0	20	
97	89	0	0	97	
61	53	20	97	0	
Минимальное остовное дерево:					
Ребро: 0 - 1, Вес: 30					
Ребро: 0 - 2, Вес: 32					
Ребро: 1 - 3, Вес: 89					
Ребро: 2 - 4, Вес: 20					

Рисунок 11 – Граф и результат

Начинаем с вершины 0. Выбираем ребро с минимальным весом: ребро 0 - 1 с весом 30. Далее ищем следующие рёбра с минимальным весом по такому же принципу. Из вершины 0 можно пройти в вершину 2 и получить следующее ребро: ребро 0 - 2 с весом 32. Из вершины 2 можно перейти в вершину 4, получаем ребро: ребро 2 - 4 с весом 20. Из вершины 1 можно перейти в вершину 3, получаем ребро: ребро 1 - 3 с весом 89. В итоге мы получаем минимальное остовное дерево: 3 - 1 - 0 - 2 - 4. Таким образом, сделав ручной подсчет на данном примере, мы удостоверились, что программа работает верно.

Заключение

Таким образом, в процессе создания данного проекта была разработана программа, осуществляющая алгоритм Прима для поиска минимального остовного дерева во взвешенном связном графе при помощи среды Microsoft Visual Studio 2019.

При выполнении курсовой работы были получены практические навыки по разработке программного обеспечения и освоены техники создания матриц смежности, фундаментально опирающиеся на теорию графов. Также были углублены знания языка программирования Си.

Программа обладает ограниченным, но достаточным для использования функционалом.

Список литературы

1. Брайан Керниган, Деннис Ритчи «Язык программирования Си»
2. М. Уэйт, С. Прата, Д. Мартин «Язык Си. Руководство для начинающих»
3. Кормен Т., Лейзерсон Ч., Ривест Р. «Алгоритмы: Построение и анализ» - М.: МЦНМО, 2001. - 960 с.
4. Оре О. «Графы и их применение»: Пер. с англ. 1965. 176 с.

Приложение А. Листинг программы.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <locale.h>
#include <time.h>

#define MAX_SIZE 100

// Функция сохранения графа в файл
void saveGraphToFile(int graph[MAX_SIZE][MAX_SIZE], int size, const char* filename) {
    FILE* file = fopen(filename, "w");
    if (file == NULL) {
        printf("Ошибка при открытии файла\n");
        return;
    }

    fprintf(file, "%d\n", size);
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            fprintf(file, "%d ", graph[i][j]);
        }
        fprintf(file, "\n");
    }

    fclose(file);

    printf("\nГраф успешно сохранен в файл graph.txt\n\n");
}

// Функция сохранения результата алгоритма Прима в файл
void savePrimResultToFile(int parent[MAX_SIZE], int graph[MAX_SIZE][MAX_SIZE], int size,
const char* filename) {
    FILE* file = fopen(filename, "w");
    if (file == NULL) {
        printf("Ошибка при открытии файла для сохранения результата\n");
        return;
    }

    fprintf(file, "Минимальное остовное дерево:\n");
    for (int i = 1; i < size; ++i) {
        fprintf(file, "Ребро: %d - %d, Вес: %d\n", parent[i], i, graph[i][parent[i]]);
    }

    fclose(file);

    printf("Результат успешно сохранен в файл result.txt\n\n");
}

// Функция для проверки существования файла
bool fileExists(const char* filename) {
    FILE* file = fopen(filename, "r");
    if (file != NULL) {
        fclose(file);
        return true; // Файл существует
    }
    return false; // Файл не существует
}

// Загрузка графа из файла
void loadGraphFromFile(int graph[MAX_SIZE][MAX_SIZE], int* size, const char* filename) {
```

```

FILE* file = fopen(filename, "r");
if (file == NULL) {
    printf("\nОшибка при открытии файла\n");
    return;
}

fscanf(file, "%d", size);
for (int i = 0; i < *size; ++i) {
    for (int j = 0; j < *size; ++j) {
        fscanf(file, "%d", &graph[i][j]);
    }
}

fclose(file);
}

// Функция для заполнения неориентированного графа случайными числами
void fillUndirectedGraphRandom(int graph[MAX_SIZE][MAX_SIZE], int size)
{
    srand(time(NULL));

    // Заполнение графа нулями и единицами на главной диагонали
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            if (i == j) {
                graph[i][j] = 0; // Главная диагональ остается нулевой
            }
            else {
                graph[i][j] = rand() % 2; // Заполнение ребер 0 или 1
                graph[j][i] = graph[i][j]; // Отражение изменений для неориентированного
графа
            }
        }
    }

    // Замена единиц случайными числами
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            if (i != j && graph[i][j] == 1) {
                graph[i][j] = rand() % 100; // Заполнение случайным числом вместо единицы
графа
                graph[j][i] = graph[i][j]; // Отражение изменений для неориентированного
            }
        }
    }
}

// Вывод графа в консоль
void printGraph(int graph[MAX_SIZE][MAX_SIZE], int size)
{
    printf("\nГраф:\n");
    for (int i = 0; i < size; ++i)
    {
        for (int j = 0; j < size; ++j)
        {
            printf("%3d ", graph[i][j]);
        }
        printf("\n");
    }
}

// Функция для нахождения несвязных вершин
int findUnconnectedNode(int parent[MAX_SIZE], int size) {
    for (int i = 0; i < size; ++i) {

```

```

        if (parent[i] == -1) {
            return i;
        }
    }
    return -1; // Если не найдено несвязанных вершин
}

// Функция выполнения алгоритма Прима для графа
void primAlgorithm(int graph[MAX_SIZE][MAX_SIZE], int size, int parent[MAX_SIZE]) {
    int minEdge[MAX_SIZE]; // Минимальные веса рёбер
    bool visited[MAX_SIZE]; // Посещали ли вершины

    // Инициализация значений
    for (int i = 0; i < size; ++i) {
        minEdge[i] = INT_MAX; // Установка начальных значений весов рёбер как
        // максимальное значение
        visited[i] = false; // Начально ни одна вершина не посещена
        parent[i] = -1; // Начально нет родительской вершины
    }

    minEdge[0] = 0; // Стартовая вершина, вес ребра равен 0

    for (int count = 0; count < size - 1; ++count) {
        int minVertex = -1; // Минимальный индекс вершины
        for (int v = 0; v < size; ++v) {
            if (!visited[v] && (minVertex == -1 || minEdge[v] < minEdge[minVertex])) {
                minVertex = v; // Нахождение вершины с минимальным весом ребра
            }
        }

        visited[minVertex] = true; // Включаем найденную вершину в остовное дерево

        // Обновляем веса смежных вершин
        for (int v = 0; v < size; ++v) {
            if (graph[minVertex][v] != 0 && !visited[v] && graph[minVertex][v] <
minEdge[v]) {
                parent[v] = minVertex;
                minEdge[v] = graph[minVertex][v];
            }
        }
    }

    // Вывод ребер минимального остовного дерева
    printf("\nМинимальное остовное дерево:\n");
    for (int i = 1; i < size; ++i) {
        if (parent[i] != -1) {
            printf("Ребро: %d - %d, Вес: %d\n", parent[i], i, graph[i][parent[i]]);
        }
        else {
            printf("Ребро: %d - %d, не существует\n", findUnconnectedNode(parent, size),
i);
        }
    }
}

int main(void) {
    setlocale(LC_ALL, "RUS");

    int graph[MAX_SIZE][MAX_SIZE];
    int size, choice;
    char filename[50];
    int parent[MAX_SIZE];
    bool b = {};

```

```

printf("Курсовая работа\n");
printf("По курсу \"Логика и основы алгоритмизации в инженерных задачах\"\n");
printf("На тему \"Реализация алгоритма Прима\"\n");
printf("Выполнил: Городничев Максим Игоревич, группа: 22ВВВ3 (22ВВВ2)\n\n");

do {
    printf("Выберите действие:\n");
    printf("1. Загрузить неориентированный граф из файла\n");
    printf("2. Заполнить неориентированный граф компьютером\n");
    printf("0. Выйти из программы\n");
    printf("Выбор: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("\nВведите имя файла для загрузки: ");
            scanf("%s", filename);
            if (!fileExists(filename)) {
                printf("\nВведён несуществующий файл\n\n");
                break;
            }
            loadGraphFromFile(graph, &size, filename);
            printGraph(graph, size);
            primAlgorithm(graph, size, parent);
            saveGraphToFile(graph, size, "graph.txt");
            savePrimResultToFile(parent, graph, size, "result.txt");
            break;
        case 2:
            printf("\nВведите размер неориентированного графа: ");
            scanf("%d", &size);
            if (size <= 0) {
                printf("\nВведён неверный размер графа\n\n");
                break;
            }
            fillUndirectedGraphRandom(graph, size);
            printGraph(graph, size);
            primAlgorithm(graph, size, parent);
            saveGraphToFile(graph, size, "graph.txt");
            savePrimResultToFile(parent, graph, size, "result.txt");
            break;
        case 0:
            return 0; // Завершение программы при выборе "0"
        default:
            printf("\nСделан неверный выбор. Пожалуйста, выберите снова.\n\n");
            break;
    }
} while (true);
}

```