Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

К курсовому проектированию

по курсу «Логика и основы алгоритмизации в ИЗ»

на тему «Реализация Алгоритма Прима»

Выполнили:

студент группы 22ВВВ2

Перфилов А.В.

Приняли:

К.э.н. доцент Акифьев И.В.

Пенза 2023

**Содержание**

[Реферат 4](#_TOC_250009)

[Введение 5](#_TOC_250008)

[Постановка задачи 6](#_TOC_250007)

[Теоретическая часть задания 7](#_TOC_250006)

[Описание алгоритма программы 9](#_TOC_250005)

[Описание программы 19](#_TOC_250004)

[Тестирование 24](#_TOC_250003)

[Ручной расчѐт задачи 29](#_TOC_250002)

[Заключение 31](#_TOC_250001)

Приложение A. Листинг программы 33

# Реферат

Отчет 20 стр, 8 рисунков.

Алгоритм Прима

Цель исследования – разработка программы, способная принимать на вход граф и на его основе выводить минимальное остовое дерево

В работе рассмотрен алгоритм прима на основе вводимых матриц смежности.

# Введение

Этот алгоритм назван в честь американского математика Роберта Прима (Robert Prim), который открыл этот алгоритм в 1957 г. Впрочем, ещё в 1930 г. этот алгоритм был открыт чешским математиком Войтеком Ярником (Vojtěch Jarník). Кроме того, Эдгар Дейкстра (Edsger Dijkstra) в 1959 г. также изобрёл этот алгоритм, независимо от них. Данный алгоритм позволяет найти минимально остовое дерево на основе входного графа.

В качестве среды разработки мною была выбрана среда MicrosoftVisualStudio2022, 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 нахождение минимального остового дерева.

# Постановка задачи

Дан взвешенный неориентированный граф G с n вершинами и m рёбрами. Требуется найти такое поддерево этого графа, которое бы соединяло все его вершины, и при этом обладало наименьшим возможным весом (т.е. суммой весов рёбер). Поддерево — это набор рёбер, соединяющих все вершины, причём из любой вершины можно добраться до любой другой ровно одним простым путём. Такое поддерево называется минимальным остовным деревом или просто минимальным остовом. Легко понять, что любой остов обязательно будет содержать n - 1 ребер.

Исходный граф в программе должен задаваться матрицей смежности, причём был реализован способ как ввода самостоятельно, так и с помощью генерации самой программой, при генерации данных должны быть предусмотрены граничные условия. Программа должна работать так, чтобы пользователь вводил количество вершин для генерации матрицы смежности. Необходимо предусмотреть различные исходы поиска, чтобы программа не выдавала ошибок и работала правильно.

Устройство ввода – клавиатура

# Теоретическая часть задания

Граф G задается множеством вершин X1, X2, ..., Xn. и множеством ребер , соединяющих между собой определенные вершины.

При представлении графа матрицей смежности информация о ребрах графа хранится в квадратной матрице, где присутствие пути из одной вершины в другую обозначается весом этого пути в виде числа от 0 до 10.

В данной работе реализуется классический вариант алгоритма прима. Он имеет очень простой вид. Искомый минимальный остов строится постепенно, добавлением в него рёбер по одному. Изначально остов полагается состоящим из единственной вершины (её можно выбрать произвольно). Затем выбирается ребро минимального веса, исходящее из этой вершины, и добавляется в минимальный остов. После этого остов содержит уже две вершины, и теперь ищется и добавляется ребро минимального веса, имеющее один конец в одной из двух выбранных вершин, а другой — наоборот, во всех остальных, кроме этих двух. И так далее, т.е. всякий раз ищется минимальное по весу ребро, один конец которого — уже взятая в остов вершина, а другой конец — ещё не взятая, и это ребро добавляется в остов (если таких рёбер несколько, можно взять любое). Этот процесс повторяется до тех пор, пока остов не станет содержать все вершины (или, что то же самое, n – 1 ребро)

Ниже представлен код функции alg\_prima

int\* otvet = (int\*)malloc(len \* sizeof(int));

int no\_edge; // номер грани

// создать массив для отслеживания выбранной вершины

//выбранное станет истинным, в противном случае — ложным

bool\* selected = (bool\*)malloc(len \* sizeof(bool));

// изначально установить выбранное значение false

memset(selected, false, sizeof(selected));

// установить количество ребер на 0

no\_edge = 0;

// количество концов в минимальном остовном дереве будет

// всегда меньше (V -1), где V — количество вершин в

//графе

// выберите 0-ю вершину и сделайте ее истинной

selected[0] = true;

int x; // номер строки

int y; // номер столбца

int l = 1;

// вывести грань и вес

printf("связи минимального остового дерева:\n");

printf("грань : вес\n");

while (no\_edge < len - 1) {

//Для каждой вершины множества S найдите все соседние вершины

// вычисляем расстояние от вершины, выбранной на шаге 1.

// если вершина уже находится в множестве S, иначе отбросим ее

//выбираем другую вершину, ближайшую к выбранной на шаге 1 вершине.

int min = INF;

x = 0;

y = 0;

for (int i = 0; i < len; i++) {

if (selected[i]) {

for (int j = 0; j < len; j++) {

if (!selected[j] && G[i][j]) { // не в selected и есть грань

if (min > G[i][j]) {

min = G[i][j];

x = i;

y = j;

}

}

}

}

}

printf("%d - %d : %d\n", x, y, G[x][y]);

otvet[l] = x;

l++;

otvet[l] = y;

l++;

otvet[l] = G[x][y];

l++;

selected[y] = true;

no\_edge++;

}

printf("\n");

otvet[0] = l;

Полный код программы можно увидеть в Приложении А.

**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 Win64.

Работа программы начинается с титульного листа курсовой работы. Далее запрашивается размер матрицы смежностей. После того как пользователь ввел размер матрицы, ему предоставляется выбрать способ задания матрицы, случайно, с помощью генерации, вручную или из текстового файла “input.txt” находящегося в папке с данной работой.

setlocale(LC\_ALL, "Rus");

int len;

int ch;

printf("\n\t\t\tКурсовая работа\nПо дисциплине Логика и основы алгоритмизации в инжинерных задачах\nНа тему : Реализация алгоритма Прима\"\n\nВыполнил студент группы 22ВВП1: Перфилов Александр\nПринял: к.э.н. доцент Акифьев И.В.\n");

getchar();

system("cls");

while (true) {

printf("Введите размер стороны матрицы смежностей: ");

if (scanf("%d", &len) != 1) { printf("Вы ввели не число"); break; };

system("cls");

int\*\* G = create\_msm(len);

int\* otvet = menu(len, G);

int\* menu(int len, int\*\* G) {

int ch;

printf("Меню\n");

printf("1-ввести матрица смежностей самостоятельно\n");

printf("2-создать матрица смежностей автоматически\n");

printf("3-скопировать матрив смежностей из файла\n");

if (scanf("%d", &ch) != 1) { printf("Вы ввели не число"); };

system("cls");

switch (ch) {

case 1:

manual\_input(G, len);

break;

case 2:

generate\_msm(len, G);

print\_msm(len, G);

break;

case 3:

read\_file(G);

print\_msm(len, G);

break;

default: printf("Вы ввели не число"); exit(0);

}

int\* otvet = alg\_prima(len, G);

return otvet;

Далее в зависимости от выбора вызывается функция создания матрицы.

int\*\* generate\_msm(int len, int\*\* m) {

int lastj = 0;

for (int i = 0; i < len; i++) {

for (int j = 0; j < len; j++)

{

if (i == j){

m[i][j] = 0;

}

else if (j >= lastj) {

int s = rand() % 10;

m[i][j] = s; // заполняем массив случайными числами

m[j][i] = s; // заполняем массив случайными числами

int lastj = j;

}

}

}

return m;

}

int\*\* manual\_input(int\*\* G, int len) {

int ch;

int lastj = 0;

for (int i = 0; i < len; i++) {

for (int j = 0; j < len; j++)

{

if (i == j) {

G[i][j] = 0;

}

else if (j >= lastj) {

printf("Введите размер пути из %d в %d: ", i, j);

if (scanf("%d", &ch) != 1) { printf("Вы ввели не число"); break; };

G[i][j] = ch;

G[j][i] = ch;

lastj = i + 1;

}

if (j == len - 1) {

printf("Массив: \n");

for (int l = 0; l < i + 1; l++) {

for (int n = 0; n < j + 1; n++) {

printf("%d", G[l][n]);

}

printf("\n");

}

printf("\n");

}

}

}

return G;

}

После этого с помощью функции “alg\_prima”, код которой был представлен выше, находится минимальное остовое дерево и предлагается сохранить результаты работы программы в файл.

printf("1 - Сохранить результаты в файл\n0 - ввести новые данные\n");

scanf("%d", &ch);

switch (ch) {

case 1:

save\_in\_file(G, len, otvet);

break;

case 0:

break;

}

system("cls");

}

Ниже можно увидеть оформление начального запроса и дальнейшие действия с ним.

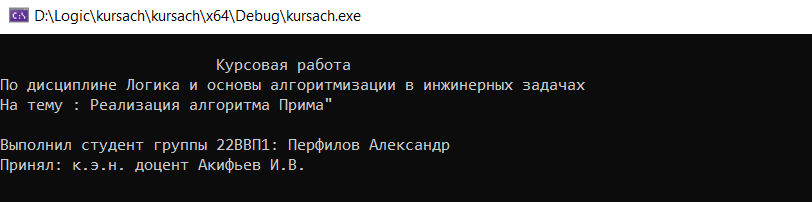


Рисунок 1

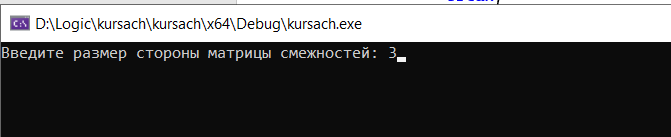


Рисунок 2

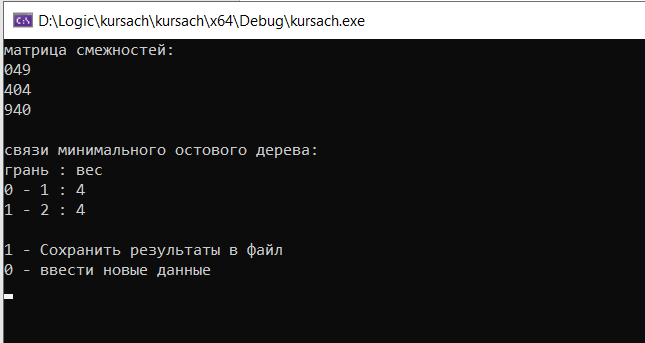


Рисунок 3

# Тестирование

Среда разработки MicrosoftVisualStudio 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 и вывод результата работы программы.

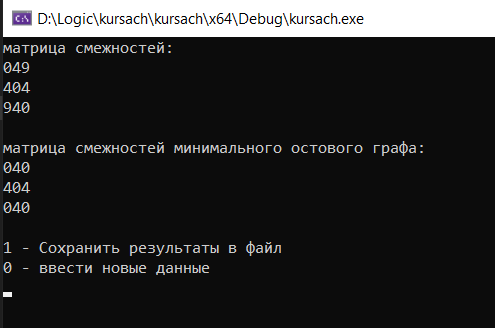


Рисунок 4 – Тестирование при вводе количество вершин = 3 и генерация матрицы.

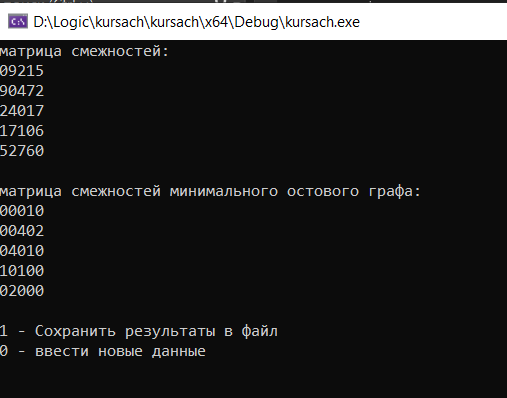


Рисунок 5 – Тестирование при вводе количество вершин = 5 и случайная генерация матрицы.

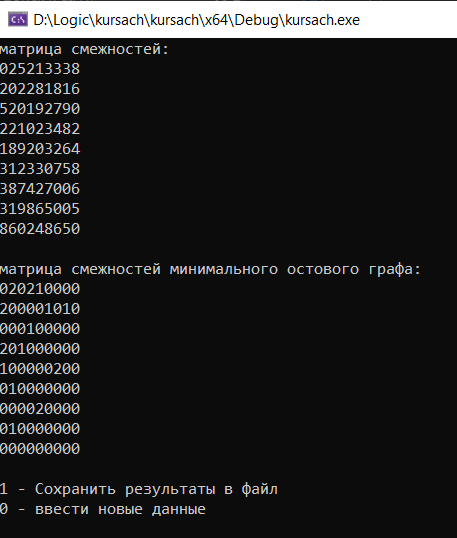


Рисунок 6 – Тестирование при вводе количество вершин = 9 и случайная генерация матрицы.

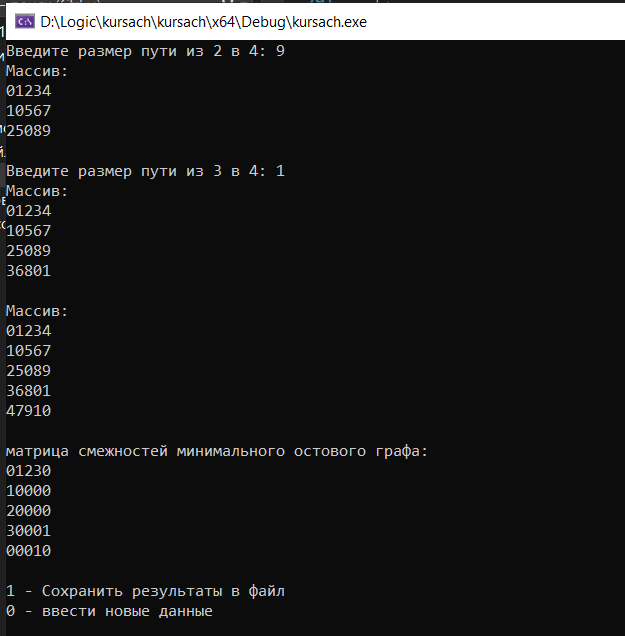


Рисунок 7– Тестирование при вводе матрицы вручную

**Ручной расчѐт задачи**

Проведем проверку программы посредством ручных вычислений на примере графа с пятью вершинами (рисунок 5).

Берем нулевую вершину и ищем минимальную дорогу, это дорога с весом 1 в вершину 3. Далее смотрим какая дорога самая короткая это дорога из вершины 3 в вершину 2. Из вершины 2 самая короткая дорога в вершину 1, потому что в вершине 0 мы уже были. Осталась последняя вершина и самая короткая дорога до нее из вершины 1.

Таким образом, можно сделать вывод, что программа работает верно.

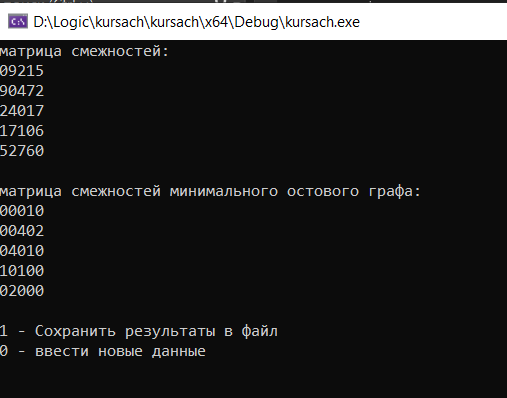


Рисунок 8 – результаты работы программы.

**Заключение**

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм Прима в MicrosoftVisualStudio 2022.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц смежностей, основанных на теории орграфов. Приобретены навыки по нахождению минимальных остовых графов. Углублены знания языка программирования Cи.

Недостатком разработанной программы является примитивный пользовательский интерфейс. Потому что программа работает в консольном режиме, не добавляющем к сложности языка сложность программного оконного интерфейса.

Программа имеет небольшой, но достаточный для использования функционал возможностей.

## Приложение А.

**Листинг программы.**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#include <string.h>

#include <locale.h>

#define INF 9999999

int\*\* create\_msm(int len) {

int\*\* m = (int\*\*)malloc(len \* sizeof(int\*));

for (int i = 0; i < len; i++) {

m[i] = (int\*)malloc(len \* sizeof(int));

}

/\*

for (int i = 0; i < len; i++) {

for (int j = 0; j < len; j++)

{

if (i == j) {

m[i][j] = 0;

}

}

}

\*/

return m;

}

int\*\* read\_file(int\*\* read\_G) {

FILE\* read;

read = fopen("input.txt", "r");

if (getc(read) == EOF) { printf("Файл пустой"); exit(0); };

if (read == NULL) { printf("Файла с таким названием не существует\n"); exit(0); }

char\* mas = (char\*)malloc(10 \* sizeof(char));

int len = 0;

while (!feof(read)) {

fscanf(read, "%c", &mas[len]);

len++;

if (mas[len - 1] == ('\n')) {

break;

}

}

len = len / 2;

fclose(read);

read = fopen("input.txt", "r");

for (int i = 0; i < len; i++) {

for (int j = 0; j < len; j++) {

fscanf(read, "%d", &read\_G[i][j]);

}

}

fclose(read);

return read\_G;

}

void save\_in\_file(int\*\* G, int len, int\* otvet) {

FILE\* Save;

Save = fopen("results.txt", "w");

for (int i = 0; i < len; i++) {

for (int j = 0; j < len; j++) {

int ch = G[i][j];

fprintf(Save, "%d", ch);

}

fprintf(Save, "\n");

}

fprintf(Save, "\n");

int ch = 3;

for (int i = 1; i < otvet[0]; i++) {

if (i == ch) {

fprintf(Save, ":");

ch += 3;

}

else if (i == ch - 1) {

fprintf(Save, "-");

}

fprintf(Save, "%d", otvet[i]);

if ( i % 3 == 0) {

fprintf(Save, "\n");

}

}

fclose(Save);

}

int\*\* manual\_input(int\*\* G, int len) {

int ch;

int lastj = 0;

for (int i = 0; i < len; i++) {

for (int j = 0; j < len; j++)

{

if (i == j) {

G[i][j] = 0;

}

else if (j >= lastj) {

printf("Введите размер пути из %d в %d: ", i, j);

if (scanf("%d", &ch) != 1) { printf("Вы ввели не число"); break; };

G[i][j] = ch; // заполняем массив случайными числами

G[j][i] = ch; // заполняем массив случайными числами

lastj = i + 1;

}

if (j == len - 1) {

printf("Массив: \n");

for (int l = 0; l < i + 1; l++) {

for (int n = 0; n < j + 1; n++) {

printf("%d", G[l][n]);

}

printf("\n");

}

printf("\n");

}

}

}

return G;

}

int\* alg\_prima(int len, int\*\* G) {

int\* otvet = (int\*)malloc(len \* sizeof(int));

int no\_edge;

bool\* selected = (bool\*)malloc(len \* sizeof(bool));

memset(selected, false, sizeof(selected));

no\_edge = 0;

selected[0] = true;

int x;

int y;

int l = 1;

printf("связи минимального остового дерева:\n");

printf("грань : вес\n");

while (no\_edge < len - 1) {

int min = INF;

x = 0;

y = 0;

for (int i = 0; i < len; i++) {

if (selected[i]) {

for (int j = 0; j < len; j++) {

if (!selected[j] && G[i][j]) {

if (min > G[i][j]) {

min = G[i][j];

x = i;

y = j;

}

}

}

}

}

printf("%d - %d : %d\n", x, y, G[x][y]);

otvet[l] = x;

l++;

otvet[l] = y;

l++;

otvet[l] = G[x][y];

l++;

selected[y] = true;

no\_edge++;

}

printf("\n");

otvet[0] = l;

return otvet;

}

void print\_msm(int len, int\*\* G) {

printf("матрица смежностей: \n");

for (int i = 0; i < len; i++) {

for (int j = 0; j < len; j++)

{

printf("%d", G[i][j]);

}

printf("\n");

}

printf("\n");

}

int\*\* generate\_msm(int len, int\*\* m) {

int lastj = 0;

for (int i = 0; i < len; i++) {

for (int j = 0; j < len; j++)

{

if (i == j){

m[i][j] = 0;

}

else if (j >= lastj) {

int s = rand() % 10;

m[i][j] = s; // заполняем массив случайными числами

m[j][i] = s; // заполняем массив случайными числами

int lastj = j;

}

}

}

return m;

}

int\* menu(int len, int\*\* G) {

int ch;

printf("Меню\n");

printf("1-ввести матрица смежностей самостоятельно\n");

printf("2-создать матрица смежностей автоматически\n");

printf("3-скопировать матрив смежностей из файла\n");

if (scanf("%d", &ch) != 1) { printf("Вы ввели не число"); };

system("cls");

switch (ch) {

case 1:

manual\_input(G, len);

break;

case 2:

generate\_msm(len, G);

print\_msm(len, G);

break;

case 3:

read\_file(G);

print\_msm(len, G);

break;

default: printf("Вы ввели не число"); exit(0);

}

int\* otvet = alg\_prima(len, G);

return otvet;

}

int main() {

setlocale(LC\_ALL, "Rus");

int len;

int ch;

printf("\n\t\t\tКурсовая работа\nПо дисциплине Логика и основы алгоритмизации в инжинерных задачах\nНа тему : Реализация алгоритма Прима\"\n\nВыполнил студент группы 22ВВП1: Перфилов Александр\nПринял: к.э.н. доцент Акифьев И.В.\n");

getchar();

system("cls");

while (true) {

printf("Введите размер стороны матрицы смежностей: ");

if (scanf("%d", &len) != 1) { printf("Вы ввели не число"); break; };

system("cls");

int\*\* G = create\_msm(len);

int\* otvet = menu(len, G);

printf("1 - Сохранить результаты в файл\n0 - ввести новые данные\n");

scanf("%d", &ch);

switch (ch) {

case 1:

save\_in\_file(G, len, otvet);

break;

case 0:

break;

}

system("cls");

}

}