Федеральное государственное бюджетное образовательное учреждение высшего образования «Пензенский Государственный университет»

Кафедра «Вычислительная техника»

Пояснительная записка

К курсовому проекту

по курсу “Логика и основы алгоритмизации

в инженерных задачах”

на тему “Реализация алгоритма Прима”

Выполнил:

студент группы 21ВВ1

Хорошильцев Егор

Приняли:

Акифьев И.В.

Юрова О.В.

Пенза 2022

**Содержание**

[Реферат 3](#_Toc122540858)

[Введение 4](#_Toc122540859)

[1 Постановка задачи 5](#_Toc122540860)

[2 Теоретическая часть задания 6](#_Toc122540861)

[3 Описание работы программы 8](#_Toc122540862)

[4 Ручной расчёт задачи 10](#_Toc122540863)

[5 Описание программы 12](#_Toc122540864)

[6 Тестирование 14](#_Toc122540865)

[Заключение 17](#_Toc122540866)

[Список литературы 18](#_Toc122540867)

[Приложение A, Листинг программы 19](#_Toc122540868)

[Приложение B, Результаты работы программы 24](#_Toc122540870)

# Реферат

Отчет 24 стр, 12 рисунков.

Алгоритм Прима

Цель исследования – разработка программы для построения минимального остовного дерева взвешенного связного неориентированного графа, используя алгоритм Прима.

В работе рассмотрены правила использования алгоритма Прима, благодаря которому происходит построение минимального остовного дерева взвешенного связанного неориентированного графа.

# Введение

Алгоритм Прима — алгоритм построения минимального остовного дерева взвешенного связного неориентированного графа. Алгоритм впервые был открыт в 1930 году чешским математиком Войцехом Ярником, позже переоткрыт Робертом Примом в 1957 году, и, независимо от них, Э. Дейкстрой в 1959 году.

На вход алгоритма подаётся связный неориентированный граф. Для каждого ребра задаётся его стоимость.

Сначала берётся произвольная вершина и находится ребро, инцидентное данной вершине и обладающее наименьшей стоимостью. Найденное ребро и соединяемые им две вершины образуют дерево. Затем, рассматриваются рёбра графа, один конец которых — уже принадлежащая дереву вершина, а другой — нет; из этих рёбер выбирается ребро наименьшей стоимости. Выбираемое на каждом шаге ребро присоединяется к дереву. Рост дерева происходит до тех пор, пока не будут исчерпаны все вершины исходного графа.

Результатом работы алгоритма является остовное дерево минимальной стоимости.

В качестве среды разработки мною была выбрана среда *MicrosoftVisualStudio 2010*, язык программирования – Си.

# 1 Постановка задачи

Требуется разработать программу, которая нужна для построения минимального остовного дерева взвешенного связного неориентированного графа.

Пользователь создаёт матрицу смежности, вводя размер графа, номера его рёбер, а также их размерность. После этого на экран должно выводиться сообщение о том, что матрица смежности сохранена в файл, а также выводится результат алгоритма на экран и так же в файл. Необходимо предусмотреть различные исходы поиска, чтобы программа не выдавала ошибок и работала правильно.

Устройства ввода – клавиатура и мышь.

Задания выполняются в соответствии с вариантом **№18.**

# 2 Теоретическая часть задания

Извлечём из множества вершину a, так как её приоритет минимален. Рассмотрим смежные с ней вершины b, c, и e. Обновим их приоритеты, как веса соответствующих рёбер ab, ac и ae, которые будут добавлены в ответ.

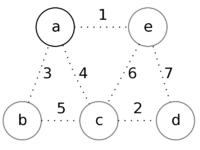


Рисунок 1- Пример графа для алгоритма Прима

Теперь минимальный приоритет у вершины е.Извлечём её и рассмотрим смежные с ней вершины a, c, и d. Изменим приоритет только у вершины d, так как приоритеты вершин a и с меньше, чем веса у соответствующих рёбер ea и ec, и установим приоритет вершины d равный весу ребра ed, которое будет добавлено в ответ.

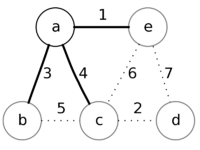


Рисунок 2 - Пример графа для алгоритма Прима

После извлечения вершины b ничего не изменится, так как приоритеты вершин a и с меньше, чем веса у соответствующих рёбер ba и bc. Однако, после извлечения следующей вершины — c, будет обновлён приоритет у вершины d на более низкий (равный весу ребра cd) и в ответе ребро ed будет заменено на cd.

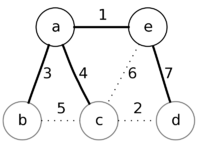


Рисунок 3 - Пример графа для алгоритма Прима

Далее будет рассмотрена следующая вершина — d, но ничего не изменится, так как приоритеты вершин e и с меньше, чем веса у соответствующих рёбер de и dc. После этого алгоритм завершит работу, так как в заданном множестве не останется вершин, которые не были бы рассмотрены.

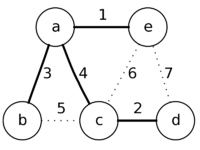


Рисунок 4 – Пример графа для алгоритма Прима

# 3 Описание работы программы

При создании программы было создано два массива – G[MAX][MAX], который отвечает за матрицу смежности, и spanning[MAX][MAX], который отвечает за матрицу связующего дерева.

Сама функция алгоритма начинается с инициализации массива пройденной дистанции (int distance[MAX]) и количества пройденных вершин (int visited [MAX]). Далее начинается цикл while, в котором и происходит алгоритм Прима. Для начала переменной min\_distance присваивается значение заранее созданной переменной infinity, вес которой составляет 9999, начинается цикл for для поиска вершины на минимальном расстоянии от дерева. Затем происходит добавление рёбер в связующее дерево. Потом начинается цикл for, в котором происходит обновление массива distance.

**prims()**

1. для i = 0 пока i < n делать i = i + 1

2. для j = 0 пока j < n делать j = j + 1

3. Если G[i][j] == 0

4. cost[i][j] = infinity

5. иначе

6. cost[i][j] = G[i][j]

7. spanning[i][j] = 0

8. конец цикла

9. distance[0] = 0

10. visited[0] =1

11. для i = 1 пока i < n делать i = i + 1

12. distance[i] = cost[0][i]

13. from[i] = 0

14. visited[i] = 0

15. конец цикла

16. min\_cost = 0

17. no\_of\_edges = n – 1

18. пока no\_of\_edges > 0

19. min\_distance = infinity

20. для i = 1 пока i < n делать i = i + 1

21. если visited[i] == 0 и distance[i] < min\_distance

22. v = i

23. min\_distance=distance[i]

24. конец цикла

25. u=from[v]

26. spanning[u][v] = distance[v]

27. spanning[v][u] = distance[v]

28. no\_of\_edges = no\_of\_edges – 1

29. visited[v] = 1

30. для i = 1 пока i < n делать i = i + 1

31. если visited[i] == 0 и cost[i][v] < distance[i]

32. distance[i] = cost[i][v]

33. from[i] = v

31. конец цикла

32. min\_cost = min\_cost + cost[u][v]

33. Конец цикла

# 4 Ручной расчёт задачи

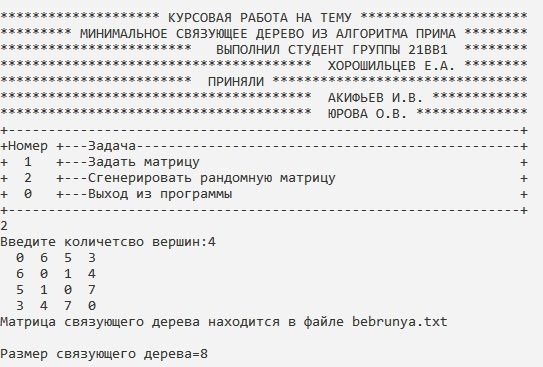


Рисунок 5 - результат работы алгоритма при случайной генерации матрицы

Проведём проверку работы программы посредством ручных вычислений на примере графа с 4 вершинами (рисунок 5).

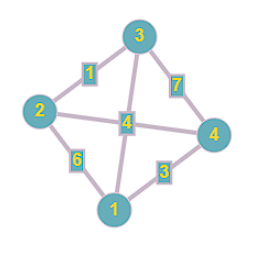


Рисунок 6 - граф по созданной матрице

Выбираем точку 1, она соединена с 2, 3, 4 точками. Вершина 4 ближайшая к 1, так как расстояние между ними 3. Выбираем ребро 1 – 4.

Следующая ближайшая вершина от вершины 1 или 2 – 3. Расстояние от 1 до 3 составляет 4. Выбираем ребро 1 – 3.

Аналогичным способом выбирается вершина 2. Расстояние от 3 до 2 – 1.

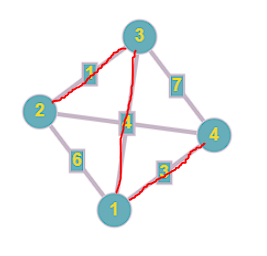


Рисунок 7 - минимальное остовное дерево

Выбраны все вершины, минимальное остовное дерево построено (выделено красным). В этом случае его вес равен 8.

Таким образом, можно сделать вывод, что программа работает полностью корректно.

# 5 Описание программы

Для написания данной программы был использован один из самых известных и приятных для освоения языков – СИ.

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

Работа программы начинается с выбора пользователем, как ему сгенерировать матрицу.

printf("+--------------------—----------------—----------------------------+\n");

printf("+Номер +---Задача------------------------------------------------+\n");

printf("+ 1 +---Задать матрицу +\n");

printf("+ 2 +---Сгенерировать рандомную матрицу +\n");

printf("+ 0 +---Выход из программы +\n");

printf("+----------------------------------------------------------------------+\n");

Если пользователь выбирает сгенерировать матрицу со случайными значениями, то ему предлагается просто ввести количество вершин, а затем ему выводится матрица смежности.

printf("Введите количетсво вершин:");

scanf("%d",&n);

for (i = 0; i < n; i++) {

for (j = i; j < n; j++) {

if (i==j){

G[i][j] = 0;

continue;

}

G[i][j] = 1 + rand() % 10;

G[j][i] = G[i][j];

//printf("%3d", array[i][j]);

}

}

for (i = 0; i < n; i++) {

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

printf("%3d", G[i][j]);

}

printf("\n");

}

Если же пользователь выбирает самостоятельно задать матрицу, то он должен ввести количество вершин, а затем самостоятельно полностью ввести матрицу.

printf("Введите количетсво вершин:");

scanf("%d",&n);

printf("\nВведите матрицу смежности:\n");

for(i=0;i<n;i++)

for(j=0;j<n;j++)

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

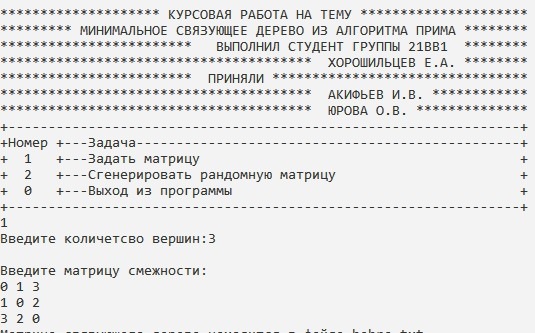


Рисунок 8 - Ввод значений и генерация матрицы смежности

`

# 6 Тестирование

Среда разработки *MicrosoftVisualStudio 2010* 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 представлены результаты работы программы.

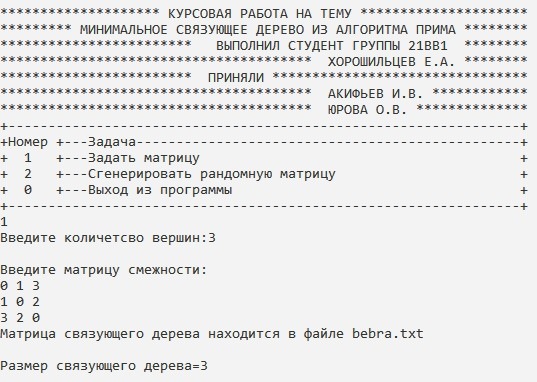


Рисунок 9 - Результат работы программы при вводе матрицы смежности вручную.

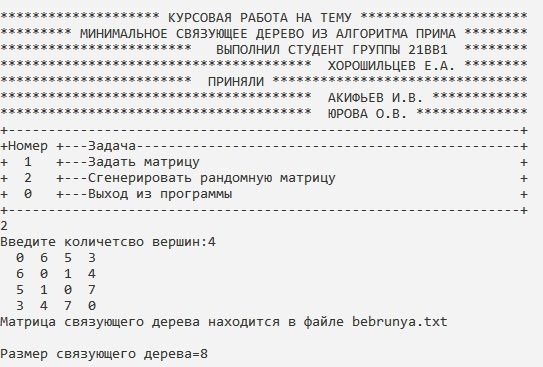


Рисунок 10 - Результат работы программы при генерации матрицы со случайными значениями.

Таблица 1 – Описание поведения программы при тестировании

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Запуск программы | Вывод сообщений о вводе количества вершин, а затем ввод рёбер и их размера | Верно |
| Работа алгоритма Прима | Вывод сообщений о размере связующего дерева | Верно |
| Создание матрицы связующего дерева | Вывод сообщения о том, что матрица была сохранена в файл | Верно |

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

# Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм Прима в *MicrosoftVisualStudio 2010*.

Выполняя данную курсовую работу, я не только воспользовался своими уже полученными в течение курса знаниями, но и приобрёл новые в результате поиска решения своих критических ошибок в сети интернет.

Моя программа имеет все нужные функции, но всё же может быть улучшена и доработана до более высокого уровня.

# Список литературы

1. Язык Си: кратко и ясно. Учебное пособие. Д.В. Парфенов
2. А.А. Тюгашев. Языки программирования. Учебное пособие. 2018 г.

# Приложение A

# Листинг программы

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<locale.h>

#define infinity 9999

#define MAX 20

int G[MAX][MAX],spanning[MAX][MAX],n;

int prims();

int adj()

{

FILE \*f;

f = fopen("bebra.txt", "w+");

setlocale(LC\_ALL, "Rus");

int i,j,total\_cost;

printf("Введите количетсво вершин:");

scanf("%d",&n);

printf("\nВведите матрицу смежности:\n");

for(i=0;i<n;i++)

for(j=0;j<n;j++)

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

total\_cost=prims();

if(total\_cost >= 9999)

{

printf("Матрица смежности была задана не верно");

printf("\n");

}

else{

printf("Матрица связующего дерева находится в файле bebra.txt");

fprintf(f, "\nМатрица связующего дерева:\n");

for(i=0;i<n;i++)

{

fprintf(f,"\n");

for(j=0;j<n;j++)

fprintf(f, "%d\t",spanning[i][j]);

}

printf("\n\nРазмер связующего дерева=%d",total\_cost);

printf("\n");

}

return 0;

}

int rand\_adj()

{

FILE \*fp;

fp = fopen("bebrunya.txt", "w+");

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

int i,j,total\_cost;

printf("Введите количетсво вершин:");

scanf("%d",&n);

for (i = 0; i < n; i++) {

for (j = i; j < n; j++) {

if (i==j){

G[i][j] = 0;

continue;

}

G[i][j] = 1 + rand() % 10;

G[j][i] = G[i][j];

//printf("%3d", array[i][j]);

}

}

for (i = 0; i < n; i++) {

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

printf("%3d", G[i][j]);

}

printf("\n");

}

total\_cost=prims();

printf("Матрица связующего дерева находится в файле bebrunya.txt");

fprintf(fp, "\nМатрица связующего дерева:\n");

for(i=0;i<n;i++)

{

fprintf(fp, "\n");

for(j=0;j<n;j++)

fprintf(fp, "%d\t",spanning[i][j]);

}

printf("\n\nРазмер связующего дерева=%d",total\_cost);

printf("\n");

return 0;

}

int prims()

{

int cost[MAX][MAX];

int u,v = 1,min\_distance,distance[MAX],from[MAX];

int visited[MAX],no\_of\_edges,i,min\_cost,j;

//создание cost[][] matrix,spanning[][]

for(i=0;i<n;i++)

for(j=0;j<n;j++)

{

if(G[i][j]==0)

cost[i][j]=infinity;

else

cost[i][j]=G[i][j];

spanning[i][j]=0;

}

//инициализация visited[],distance[] and from[]

distance[0]=0;

visited[0]=1;

for(i=1;i<n;i++)

{

distance[i]=cost[0][i];

from[i]=0;

visited[i]=0;

}

min\_cost=0; //вес связующего дерева

no\_of\_edges=n-1; //количество добавляемых ребер

while(no\_of\_edges>0)

{

//поиск вершины на минимальном расстоянии от дерева

min\_distance=infinity;

for(i=1;i<n;i++)

if(visited[i]==0&&distance[i]<min\_distance)

{

v=i;

min\_distance=distance[i];

}

u=from[v];

//добавить ребро в связующее дерево

spanning[u][v]=distance[v];

spanning[v][u]=distance[v];

no\_of\_edges--;

visited[v]=1;

//обновление массива distance[]

for(i=1;i<n;i++)

if(visited[i]==0&&cost[i][v]<distance[i])

{

distance[i]=cost[i][v];

from[i]=v;

}

min\_cost=min\_cost+cost[u][v];

}

return(min\_cost);

}

void menu()

{

setlocale(LC\_ALL, "Rus");

printf("\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* КУРСОВАЯ РАБОТА НА ТЕМУ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\* МИНИМАЛЬНОЕ СВЯЗУЮЩЕЕ ДЕРЕВО ИЗ АЛГОРИТМА ПРИМА \*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ВЫПОЛНИЛ СТУДЕНТ ГРУППЫ 21ВВ1 \*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ХОРОШИЛЬЦЕВ Е.А. \*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ПРИНЯЛИ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* АКИФЬЕВ И.В. \*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ЮРОВА О.В. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("+--------------------—----------------—--------------------------+\n");

printf("+Номер +---Задача------------------------------------------------+\n");

printf("+ 1 +---Задать матрицу +\n");

printf("+ 2 +---Сгенерировать рандомную матрицу +\n");

printf("+ 0 +---Выход из программы +\n");

printf("+----------------------------------------------------------------+\n");

}

int main()

{

while (1)

{

menu();

int n = 0;

scanf("%d", &n);

switch (n)

{

case 1:

adj();

break;

case 2:

rand\_adj();

break;

case 0:

exit(0);

default:

{

printf("Ошибка \ n");

break;

}

}

}

return 0;

}

# Приложение B

# Результаты работы программы

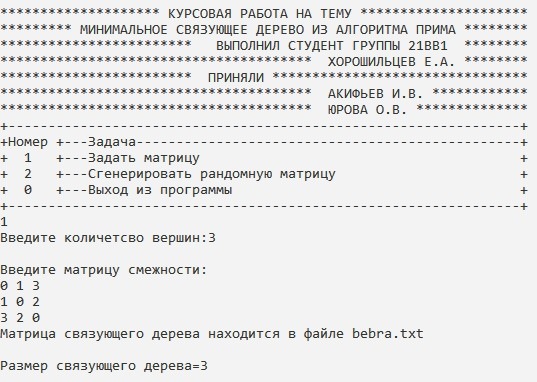


Рисунок 11 - Результат работы программы при ручном вводе

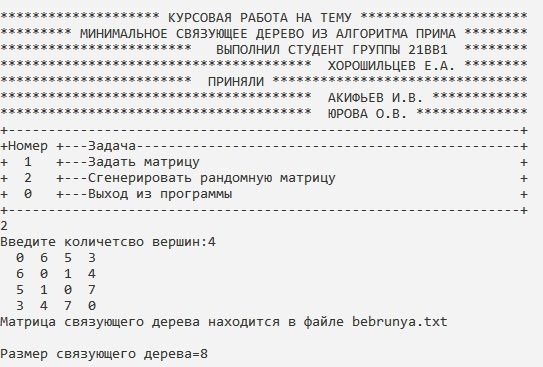


Рисунок 12 - Результат работы программы при случайной генерации матрицы