|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  Федеральное государственное бюджетное  образовательное учреждение высшего образования  «тюменский индустриальный университет»  Институт геологии и нефтегазодобычи  Кафедра кибернетических систем | | | | | |
| РАЗРАБОТКА ПО ПОСТОРОЕНИЯ ГАМИЛЬТОНОВОЙ ЦЕПИ В ГРАФЕ ПО НЕРЕКУРСИВНОМУ АЛГОРИТМУ ГЕНЕРАЦИИ ВСЕХ ПРЕСТАНОВОК ВЕРШИН В ЛЕКСИГРАФИЧЕСКОМ ПОРЯДКЕ | | | | | |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  к курсовой работе  по дисциплине «Математические основы программирования» | | | | | |
|  |  | |  | РУКОВОДИТЕЛЬ: | |  |
|  |  | |  | *ст. преп. кафедры КС*  *\_\_\_\_\_\_\_\_\_\_\_ Лаптева У. В.* | |  |
|  |  | |  |  | |  |
|  |  | |  | РАЗРАБОТЧИК: | |  |
|  |  | |  | *обучающийся группы УТСб-15-1*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кондаков А.Ю.* | |  |
|  |  | |  |  |  |
|  |  |  | Курсовая работа  защищена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |  |
|  | 2019 | | | |  |

**СОДЕРЖАНИЕ**

**Введение**3

**1.Постановка задачи на разработку ПО** 4

1.1 Характеристики комплекса задач4

1.2 Выходная информация5

1.3 Входная информация6

**2. Описание теоретических оснований решения задания7**

2.1.Основные понятия и определения.7

2.2 Способы задания графов10

2.3 Алгоритм перестановки13

**3. Схема решения задачи15**

**4. Описание структуры программы16**

**5. Описание тестирования программного обеспечения 18**

**6. Руководство пользователя21**

6.1 Введение21

6.2 Назначение и условия применения21

6.3 Подготовка к работе21

6.4 Описание операций21

6.5 Заключительные действия 22

**Заключение** 23

Список источников 24

Приложение 1.Листинг программы 25

**ВВЕДЕНИЕ**

Объектом исследования данной курсовой работы являются графы – абстрактные математические объекты, представляющие собой совокупность двух множеств: множества точек, которые называются вершинами, и множества ребер.

Предметом исследования данной курсовой работы являются гамильтоновы цепи в неориентированных графах. **Гамильтоновой цепью** в неориентированном графе называется цепь, проходящая через каждую его вершину один и только один раз.

Родившись при решении головоломок и занимательных игр, теория графов стала в настоящее время простым, доступным и мощным средством решения вопросов, относящихся к широкому кругу проблем. Графы буквально вездесущи. В виде графов можно, например, интерпретировать схемы дорог и электрические цепи, географические карты и молекулы химических соединений, связи между людьми и группами людей. За последние четыре десятилетия теория графов превратилась в один из наиболее бурно развивающихся разделов математики. Это вызвано запросами стремительно расширяющейся области приложений. Применяется при проектировании интегральных схем и схем управления, при исследовании автоматов, логических цепей, блок- схем программ, в экономике и статистике, химии и биологии, в теории расписаний. Поэтому актуальность темы обусловлена популярностью графов и связанных с ними методов исследований

1. **Постановка задачи на разработку ПО**
   1. Характеристики комплекса задач

Необходимо реализовать решение задачи, нахождения гамильтоновой цепи, в неориентированном графе используя не рекурсивный алгоритм генерации всех перестановок вершин графа в лексиграфическом порядке, на языке программирования С++.

Достоинства языка:

- С++ поддерживает разные технологии и стили программирования, включая объектно-ориентированное, обобщенное и метапрограммирование;

- более надежное освобождение ресурсов путем автоматического вызова деструкторов при уничтожении объектов;

- возможность создавать пользовательские функции-операторы, позволяющие записывать выражения над пользовательскими типами в алгебраической форме;

- важным достоинством является предсказуемое выполнение программ, что является важным для систем реального времени;

- при помощи шаблонов можно создавать обобщенные контейнеры и алгоритмы для разных типов данных;

- при помощи шаблонов и множественного наследия можно имитировать классы-примеси и комбинаторную параметризацию библиотек;

- поддержка понятия физической и логической константности, что делает программу надежнее, так как позволяет диагностировать ошибочные попытки изменения переменных;

- C++ - кросплатформенный язык, его стандарт накладывает минимальные требования на компьютер для запуска скомпилированной программы. На С++ разрабатывают программы для различных платформ и систем с помощью компиляторов для большого количества платформ;

- в С++ существует возможность низкоуровневой работы с адресами памяти, однако, при неосторожном использовании это легко может превратиться в недостаток;

Комплекс задач, реализуемый программным обеспечением, предназначен для:

- автоматизации нахождения всех гамильтоновых цепей в заданном пользователям графе, в виде матрицы смежности и количества вершин.

* 1. Выходная информация

Выходная информация представляет собой вывод на экран пользователя консоль единожды, включая в себя всю обработанную алгоритмом информацию:

- Количество вершин графа

- Матрицу смежности заданного графа

-Все возможные гамильтоновы цепи отсортированные в лексиграфическом порядке

Сроки выдачи и допустимое время решения определяются временем выполнения алгоритма программы, которое в свое очередь зависит от вычислительных мощностей компьютера пользователя

Таблица 1.2.1- Выходная информация

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Идентификатор | Назначение выходной информации | Форма представления | Периодичность выдачи | Получатели |
| n | Количество вершин графа | Видеокадр | Единожды | Пользователь ПО |
| mas[n][n] | Матрица смежности |
| a[n] | Массив в котором хранится последовательность вершин |

* 1. Входная информация

Входная информация задаётся через источник информации представляющий собой текстовый файл формата .txt

Идентификатор файла input.txt.

Файл должен, находится непосредственно в одной папке с программой, и должен содержать входную информацию включающую в себя количество вершин графа(n) и матрицу смежности (mas[n][n]).

Входное сообщение подаётся единожды.

Таблица 1.3.1-Входная информация

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Требуемая точность числового значения | Источник информации | Идентификатор  Источника информации |
| n | Целые числа | Текстовый файл формата .txt | Input.txt |
| mas[n][n] | 0 или 1 |

1. **ОПИСАНИЕ ТЕОРЕТИЧЕСКИХ ОСНОВ РЕШЕНИЯ ЗАДАНИЯ**
   1. Основные понятия и определения

Простым графом G(V, E) называется совокупность двух множеств – непустого множества V и множества E неупорядоченных пар различных элементов множества V . Множество V называется множеством вершин, множество E называется множеством ребер.

G(V, E) =< V, E >, V 6= ∅, E ∈ V × V.

Вместо термина «простой граф» часто употребляется термин «граф». Число вершин графа обозначим через p, число ребер через q.

Вершины графа будем обозначать латинскими буквами (a, b, c, ...), либо буквой v с указанием номера вершины (v1, v2, ..., vp). Ребра графа будем обозначать греческими буквами (α, β, γ, ...), либо буквой e с указанием номера ребра (e1, e2,..., eq).

Пусть V = {v1, v2, v3, v4}, E = {e1, e2, e3, e4, e5}, где e1 = (v2, v1),

e2 = (v3, v1), e3 = (v4, v3), e4 = (v1, v4), e5 = (v4, v2) тогда p = 4, q = 5.

Обычно граф изображают диаграммой: вершины – точками, ребра – линиями.

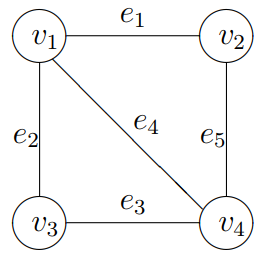


Рис.2.1.1Неориентированый граф.

Если элементами множества E являются упорядоченные пары (т.е. пары, в которых фиксирован порядок элементов), то граф называется ориентированным (или орграфом). В этом случае элементы множества V называются узлами, а элементы множества E – дугами. Первую вершину упорядоченной пары называют началом дуги, вторую – концом.

На диаграмме дуги изображаются стрелками, стрелка указывает направление от начала до конца дуги.

На рисунке 2.2 изображен орграф с теми же множествами вершин и ребер, что: V = {v1, v2, v3, v4}, E = {e1, e2, e3, e4, e5}, где e1 = (v2, v1), e2 = (v3, v1), e3 = (v4, v3), e4 = (v1, v4), e5 = (v4, v2).

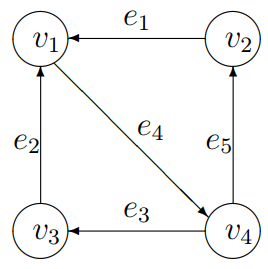


Рис.2.1.2 Ориентированный граф.

Маршрутом в графе называется последовательность вершин

и ребер вида v0e1v1e2...ekvk, в которой ei = (vi−1, vi). Вершина v0 называется начальной, а vk – конечной вершиной маршрута.

Это определение подходит также для псевдо-, мульти- и орграфов. Для неориентированного графа достаточно указать только последовательность вершин либо только последовательность ребер.

Если все ребра в маршруте различны, то маршрут называется цепью. Если все вершины (а значит и ребра) в маршруте различны, то маршрут называется простой цепью. В цепи v0e1v1e2...ekvk, вершины v0, vk называются концами цепи. Говорят, что цепь с концами u, v соединяет вершины u, v (обозначается < u, v >).

Если v0 = vk, маршрут называется замкнутым.

Если все ребра в замкнутом маршруте различны, то он называется циклом. Если все вершины в замкнутом маршруте, кроме первой и последней, различны, то он называется простым циклом.

Для орграфов цепь называется путем, а цикл контуром.

Гамильтоновой цепью в неориентированном графе называется цепь, проходящая через каждую его вершину один и только один раз.

Гамильтоновым циклом в неориентированном графе называется цикл, проходящий через каждую вершину один и только один раз за исключением начальной вершины, которая совпадает с конечной.

Рассмотрим граф G(V, E).

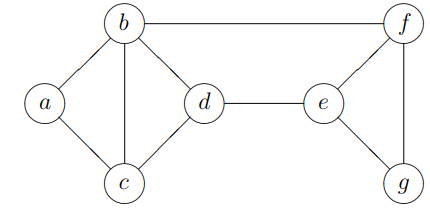


Рис.2.1.3 Пример графа для рассмотрения путей и маршрутов.

abdbc – маршрут, но не цепь;

abdcb – цепь, но не простая цепь;

abcde – простая цепь;

abdbca – замкнутый маршрут, но не цикл;

abfedbca – цикл, но не простой цикл;

abca – простой цикл.

abcdefg – гамильтонова цепь

acdegfba – гамильтонов цикл.

* 1. Способы задания графов

Рассмотрим граф G(V, E) с p вершинами и q ребрами. Он может быть задан несколькими способами.

* Списки вершин и ребер.(рис 2.2.1).
* Списки смежности. Каждой вершине графа сопоставляется список смежных ей вершин. (рис 2.2.2).

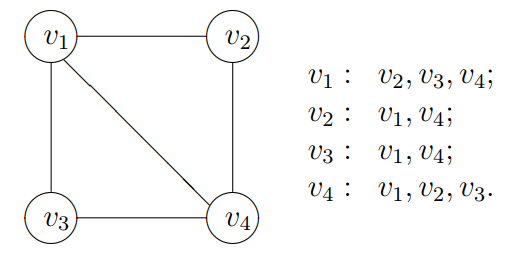


Рис.2.2.1 Граф и соотвествующий ему список вершин и ребёр.

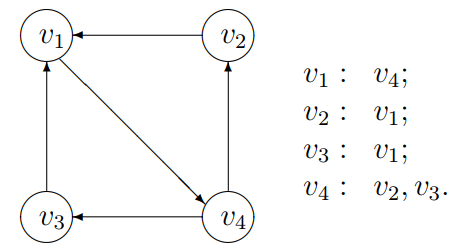
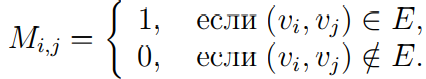
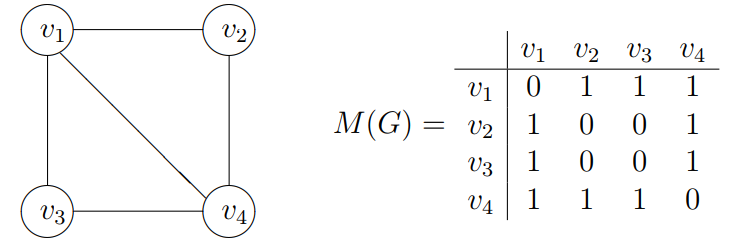


Рис.2.2.2 Ограф и соответвующий ему список смежности.

* Матрица смежности. Так называется матрица M : p × p,(рис.2.2.3).



Это определение подходит и для орграфа D(V, E).(рис.2.2.4).

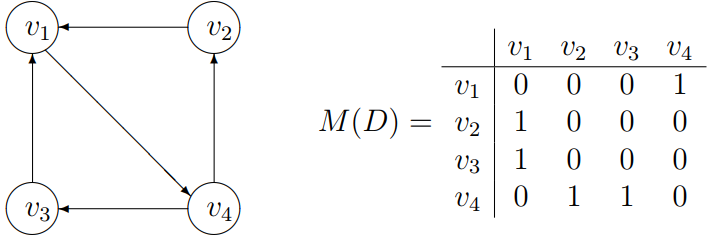
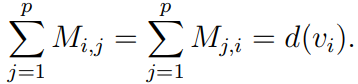
Рис.2.2.3 Неориентированный граф и соответствующая ему матрица смежности.

Рис.2.2.4 Ограф и соответствующая ему матрица смежности.

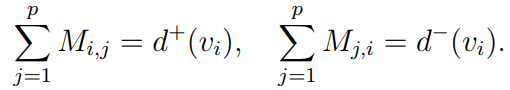
Приведем основные свойства матрицы смежности

1) Матрица смежности неориентированного графа симметрична относительно главной диагонали: M(G) = M(G) T .

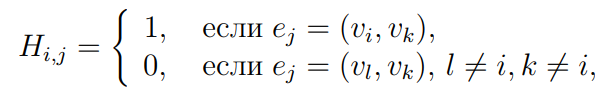
2) Сумма элементов матрицы смежности неориентированного графа M(G) по i-й строке (по i-му столбцу) равна степени i-й вершины:



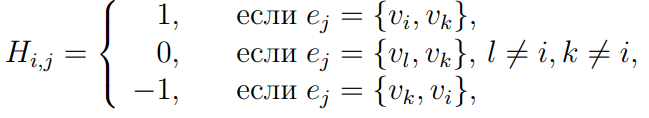
3) Суммы элементов матрицы смежности ориентированного графа M(D) по i-й строке и по i-му столбцу соответственно равны полустепени исхода и полустепени захода i-й вершины:



* Матрица инцидентности. Так называется матрица H : p × q,



для неориентированного графа,(рис.2.2.5).



для ориентированного графа.(рис.2.2.6).

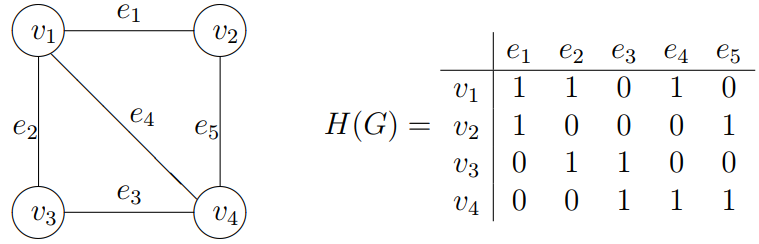
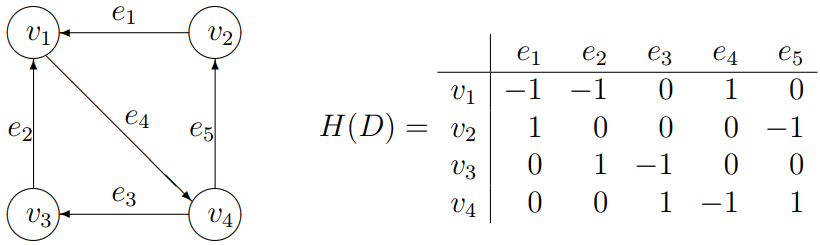


Рис.2.2.5 Неориентированный граф и соответствующая ему матрица инцидентности.

Рис.2.2.6 Ограф и соответствующая ему матрица инцидентности.

* 1. Алгоритм перестановки
  2. Алгоритм перестановок в лексиграфическом порядке

Перестановка – это комбинация элементов из N разных элементов взятых в определенном порядке. В перестановке важен порядок следования элементов, и в перестановке должны быть задействованы все N элементов.

Перестановки без повторений

Количество перестановок для N различных элементов составляет N!. Действительно:

* на первое место может быть помещен любой из N элементов (всего вариантов N),
* на вторую позицию может быть помещен любой из оставшихся

(N-1) элементов (итого вариантов N·(N-1)),

* если продолжить данную последовательность для всех N мест, то получим: N·(N-1)·(N-2)· … ·1, то есть всего N! перестановок.

Рассмотрим задачу получения всех перестановок чисел 1…N (то есть последовательности длины N), где каждое из чисел входит ровно по 1 разу. Существует множество вариантов порядка получения перестановок. Однако наиболее часто решается задача генерации перестановок в лексикографическом порядке (см. пример выше). При этом все перестановки сортируются сначала по первому числу, затем по второму и т.д. в порядке возрастания. Таким образом, первой будет перестановка 1 2 … N, а последней — N N-1 … 1.

Рассмотрим алгоритм решения задачи. Дана исходная последовательность чисел. Для получения каждой следующей перестановки необходимо выполнить следующие шаги:

1) Необходимо просмотреть текущую перестановку справа налево и при этом следить за тем, чтобы каждый следующий элемент перестановки (элемент с большим номером) был не более чем предыдущий (элемент с меньшим номером). Как только данное соотношение будет нарушено необходимо остановиться и отметить текущее число (позиция 1).

2) Снова просмотреть пройденный путь справа налево пока не дойдем до первого числа, которое больше чем отмеченное на предыдущем шаге.

3) Поменять местами два полученных элемента.

4) Теперь в части массива, которая размещена справа от позиции 1 надо отсортировать все числа в порядке возрастания. Поскольку до этого они все были уже записаны в порядке убывания необходимо эту часть подпоследовательность просто перевернуть.

Таким образом мы получим новую последовательность, которая будет рассматриваться в качестве исходной на следующем шаге.

1. **Схема решения задачи**

**Начало**

Инициализация функции перебора вершин в лексиграфическом порядке

**Конец**

Вывод последовательности на экран

**ДА**

**ДА**

**НЕТ**

Последняя перестановка?

Проверка на гамильтонову цепь

**НЕТ**

Чтение данных из файла

1. **Описание структуры программы**

Структура данных

* Количество вершин
* Матрица смежности
* Последовательность вершин

Количество вершин не может превышать 15, и быть меньше 2.

Матрица смежности задаётся пользователем. Должна иметь 0, на главной диагонали и быть симметрична относительно главной диагонали

Последовательность вершин не может превышать 15, и быть меньше 2.

Описание массивов в программе приведено в таблице 4.1.

Таблица 4.1 – Обозначения и описания массивов.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя массива | Размерность массива | Описание массива |
| 1 | mas | [Количество вершин][Количество вершин]- [n][n] | Для хранения данных о ребрах |
| 2 | a | [Количество вершин]- [n] | Для хранения текущей перестановки |

Описание переменных в программе приведено в таблице 4.2.

Таблица 4.2 - Описание переменных.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип переменной | Описание переменной |
| 1 | n | int | Количество вершин |
| 2 | i,j | int | Счетчики |

Состав и назначения процедур и функцийв программы приведены в таблице 4.3

Таблица 4.3 – Процедуры и функции программы.

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование процедуры, функции | Тип принимаемого значения | Тип возвращаемого значения | Назначение |
| ReadInputFile | int(n),int(\*\*mas) | void | Чтение данных из файла/вывод данных на экран |
| swap | Int(\*a),int(i),int(j) | void | Перестановка элементов в последовательности вершин |
| NextSet | Int(\*a),int(n) | bool | Перестановка в лексиграфическом порядке |
| Print | Int(\*a),int(n) | void | Вывод последовательности на экран |
| GamiltonChain | Int(\*a),int(n),int(\*\*mas) | bool | Проверка наличия гамильтоновой цепи |

1. **Описание тестирования программного обеспечения**

Для тестирования были выбраны произвольные графы, заданные матрицей смежности.

Тест №1.

На рисунке 5.1 изображен граф G1 заданный матрицей смежности  
M1=

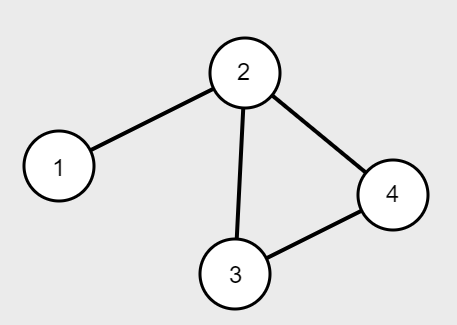


Рис.5.1 - Произвольный неориентированный граф G1.

Таблица 5.1 – Данные для теста №1.

|  |  |
| --- | --- |
| Исходные данные | Ожидаемый результат |
| Заданный граф  Матрица смежности M1 | Программой будут выданы 4 гамильтоновы цепи в лексиграфичесом порядке:  1-2-3-4; 1-2-4-3; 3-4-2-1; 4-3-2-1; |

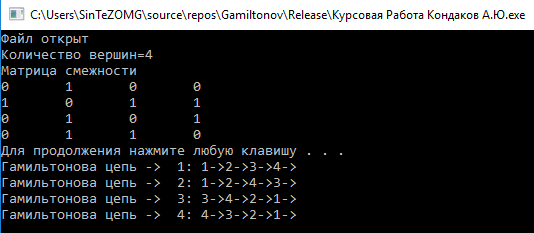


Рис.5.2 – Результат тестирования программы тест №1

Тест №2.

На рисунке 5.3 изображен граф G1, заданный матрицей смежности.

M2=

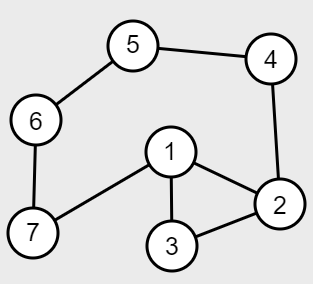


Рис.5.3 – Произвольный неориентированный граф

Таблица 5.2 – Данные для теста №2.

|  |  |
| --- | --- |
| Исходные данные | Ожидаемый результат |
| Заданный граф  Матрица смежности M1 | Программой будут выданы 18 гамильтоновых цепей в лексиграфичесом порядке:  1-3-2-4-5-6-7; 1-7-6-5-4-2-3;  2-3-1-7-6-5-4; 2-4-5-6-7-1-3;  3-1-2-4-5-6-7; 3-1-7-6-5-4-2;  3-2-1-7-6-5-4; 3-2-4-5-6-7-1;  4-2-3-1-7-6-5; 4-5-6-7-1-2-3;  4-5-6-7-1-3-2; 5-4-2-3-1-7-6;  5-6-7-1-3-2-4; 6-5-4-2-3-1-7;  6-7-1-3-2-4-5; 7-1-3-2-4-5-6;  7-6-5-4-2-1-3; 7-6-5-4-2-3-1; |

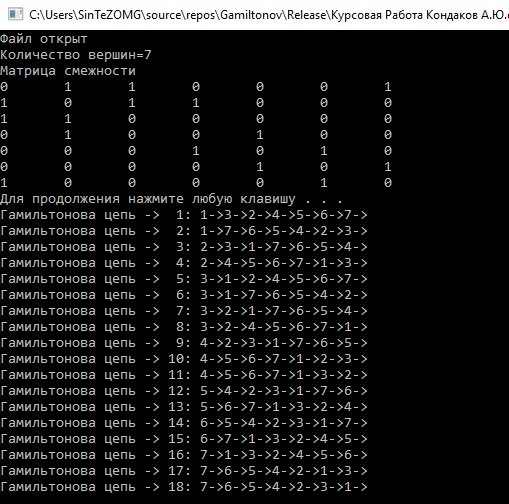


Рис.5.4 – Результат тестирования программы тест №2.

1. **Руководство пользователя**
   1. Введение

Требования настоящего руководства применяются при эксплуатации программного обеспечения.

Программа предназначена для решения задач по нахождения всех возможных гамильтоновых цепей в графах.

Пользователь программы должен иметь опыт работы с ОС семейства Windows, а также иметь представление о теории графов.

Для работы с программой не требуется ознакомление с эксплуатационной документацией.

* 1. Назначение и условия применения

Программа предназначена для поиска всех возможных гамильтоновых цепей на заданном графе.

* 1. Подготовка к работе

Условия, при соблюдении которых возможно выполнение операции

Количество вершин не может превышать 15, и быть меньше 2.

Матрица смежности должна иметь 0, на главной диагонали, и быть симметрична главной диагонали матрицы.

До работы программы, нужно обязательно заполнить или создать файл input.txt хранящий в себе данные о количестве вершин, и заданный граф представленный в виде матрицы смежности.

* 1. Описание операций

Запустить программную среду, используя соответствующее приложение.

* 1. Заключительные действия

Для завершения работы программы после выполнения всех операций по нахождения кратчайшего пути на графе и вывода ответа, необходимо нажать на любую клавишу на клавиатуре.

**ЗАКЛЮЧЕНИЕ**

В данной курсовой работе, были рассмотрены основные элементы теории графов. Гамильтоновы графы, цепи и циклы. На основании полученных данных была разработана программа реализации алгоритма перестановки вершин в лексиграфическом порядке, с поиском гамильтоновых цепей на каждой из возможных последовательностей вершин.

СПИСКОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ

1. Ю.Б.Буркатовская Теория графов. Часть 1: учебное пособие / Ю.Б.Буркатовская; Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2014. – 200 с.
2. Д.В.Карпов Теория графов СПб.: Санкт-Петербургское отделение Мат. института им. В. А. Стеклова РАН, 2017. — 482 с.
3. Генерация перестановок с повторение и без повторений [Электронный ресурс] // Режим доступа: https://prog-cpp.ru/permutation/(дата обращения: 24.06.2019)

Приложение 1 Листинг программы

// Курсовая Работа Кондаков А.Ю.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include "stdlib.h"

#include "iostream"

#include "time.h"

#include <clocale>

#include "fstream"

using namespace std;

void ReadInputFile(int n, int \*\*mas)

{

//Заполнение матрицы значениями из файла

string path = "input.txt";

ifstream fin;

fin.open(path);

fin >> n;

if (!fin.is\_open())

{

cout << "Ошибка открытия файла!" << endl;

}

else {

cout << "Файл открыт" << endl;

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

{

mas[i] = new int[n];

}

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

{

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

{

fin >> mas[i][j];

}

}

}

///////////////////////////////

//Вывод матрицы

cout << "Количество вершин=" << n <<"\n";

cout << "Матрица смежности \n";

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

{

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

{

cout << mas[i][j] << "\t";

}

cout << "\n";

}

///////////////////////////////////

system("pause");

}

void swap(int \*a, int i, int j) // процедура перестановки ( используется в функции перебора)

{

int s = a[i];

a[i] = a[j];

a[j] = s;

}

bool NextSet(int \*a, int n)// функция перебора в лексиграфическом порядке

{

int j = n - 2;

while (j != -1 && a[j] >= a[j + 1]) j--;

if (j == -1)

return false; // больше перестановок нет

int k = n - 1;

while (a[j] >= a[k]) k--;

swap(a, j, k);

int l = j + 1, r = n - 1; // сортируем оставшуюся часть последовательности

while (l<r)

swap(a, l++, r--);

return true;

}

void Print(int \*a, int n) // вывод перестановки

{

static int num = 1; // номер перестановки

cout.width(3); // ширина поля вывода номера перестановки

cout << num++ << ": ";

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

cout << a[i] << "->";

cout << endl;

}

bool GamiltonChain(int \*a, int n, int \*\*mas) // функция проверки наличия гамильтоновой цепи в графе

{

for (int i = 0; i < n - 1; i++) //цикл перебора вершин для проверки на матрице смежности

{

if (mas[a[i] - 1][a[i + 1] - 1] == 0)

return false; // возвращает false если хотя бы одного ребра нет в последовательности вершин

}

return true; // если каждая пара вешршин после проверки на матрице смежности = 1 , возвращает true (что и есть выполнение условия наличия гамильтоновой цепи в графе)

}

int main()

{

system("chcp 1251");

system("cls"); setlocale(LC\_ALL, "ru");

string path = "input.txt";

ifstream fin;

fin.open(path);

//Обьявление матрицы (матрица размера [n][n])

int n; // n- количество вершин в графе ,

fin >> n;

int \*\*mas = new int\*[n];

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

{

mas[i] = new int[n];

}

///////////////////////////////

ReadInputFile(n, mas);

int \*a; // а- массив номеров вершин (используется в функции перебора вершин в лексиграфическом порядке)

a = new int[n];

for (int i = 0; i < n; i++) // цикл заполнения массива номерами вершин от 1 до n

a[i] = i + 1;

if (GamiltonChain(a, n, mas)) { cout << "Гамильтонова цепь ->"; Print(a, n); }

while (NextSet(a, n))

if (GamiltonChain(a, n, mas)) {

cout << "Гамильтонова цепь ->";

Print(a, n);

}

cin.get(); cin.get();

return 0;

}