Министерство науки и образования РФ

Федеральное государственное автономное образовательное

учреждение высшего профессионального образования

«Санкт-Петербургский государственный электротехнический

университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»

(СПбГЭТУ «ЛЭТИ»)

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

Пояснительная записка к курсовой работе

на тему:

“ Графы”

по дисциплине “Алгоритмы и структуры данных”

Вариант 19

Выполнил студент гр. 4306: Табаков А. В.

Принял: Колинько П. Г.

Санкт-Петербург  
2015

Оглавление

Цель3

1. Задание3
2. Содержание работы3
3. Способ представления данных в памяти3
4. Описание алгоритма Прима3
5. Оценка временной сложности3
6. Контрольные примеры4
7. Результаты решения задачи6

Вывод7

Список используемой литературы8

Приложение9

**Цель**

Получить практические навыки работы с графами на языке программирования «C\C++».

1. **Задание**

Реализовать алгоритм нахождения стягивающего дерева наименьшей стоимости неориентированного графа с нагруженными рёбрами (алгоритм Прима).

1. **Уточнение задания**

Разработать и реализовать в виде программы алгоритм по предложенному индивидуальному заданию. Дать теоретическую оценку временной сложности алгоритма и сравнить её с оценкой по тексту программы. Учесть оценки сложности внутренних алгоритмов ввода исходных данных и вывода результата.

1. **Способ представления данных в памяти**

При выполнении задания для хранения информации о неориентированном графе был выбран метод хранения массива рёбер графа для исходных данных и результата. Матрица смежности в данной работе не подходит, так как рёбра имеют веса. Из предыдущего логично предположить, что подойдёт весовая матрица, но, во-первых, она будет иметь пустые ячейки, вдобавок к этому веса будут повторяться, т. к. граф неориентированный. Во-вторых, для одного обхода всей матрицы временная сложность будет порядка O(n^2). Матрица инцидентности имеет схожую проблему, обход всей матрицы займёт порядка O(n^2) времени.

Вывод: отсортированный массив рёбер, по нашему мнению, лучше всего подходит для хранения результата ввода и вывода.

Для выполнения обработки были использованы список рёбер, создаваемый из массива рёбер, также был использован массив обработанных вершин.

1. **Описание алгоритма Прима**

Отсортировать массив рёбер.

Записать массив рёбер в список рёбер.

Взять первую вершину из списка рёбер.

Записать вершину в массив обработанных вершин.

ПОВТОРЯТЬ ПОКА (Все вершины не занесены в массив обработанных)

ПОВТОРЯТЬ ПОКА (Не конец списка рёбер и вершина не обработана)

1. Выделить первое ребро из списка, соединяющего обработанную вершину с необработанной.
2. Записать необработанную вершину в массив обработанных.
3. Записать данное ребро в массив с результатами.
4. Удалить данное ребро из списка рёбер.

КОНЕЦ ЦИКЛА

КОНЕЦ ЦИКЛА

1. **Оценка временной сложности**

Временная сложность представлена в таблице 1*.*

Таблица. 1. Временная сложность

|  |  |  |
| --- | --- | --- |
| **Функция** | **Ожидаемая** | **Фактическая** |
| Создание графа (массив рёбер) | O(m) | O(m) |
| Обработка | O(m\*log m) | O(m\*log m) |
| Вывод | O(m) | O(m) |

1. **Контрольные примеры**

Контрольные примеры представлены в таблицах 2-4 и рисунках 1-3*.*

*Таблица 2. Количество вершин 5*

|  |  |  |
| --- | --- | --- |
| Ребро | Вес | Результат |
| a-b | 5 | a-e: -6,34 |
| a-c | 0 | a-d: -1.341 |
| a-d | -1,341 | a-c: 0 |
| a-e | -6,34 | b-e: 0.23 |
| b-c | 7,124 |  |
| b-d | 0,5 |
| b-e | 0,23 |
| c-d | 4 |
| d-e | 2 |

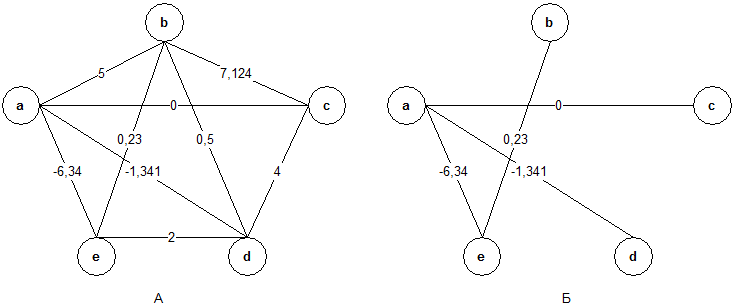


Рис.1. Граф, построенный по таблице 2 (А-исходный, Б-результат).

*Таблица 3. Количество вершин 4*

|  |  |  |
| --- | --- | --- |
| Ребро | Вес | Результат |
| a-b | 0 | a-d: -1,341 |
| a-c | -1 | a-c: -1 |
| a-d | -1,341 | a-b: 0 |
| b-c | 3,032 |  |
| b-d | 6 |
| c-d | 6 |

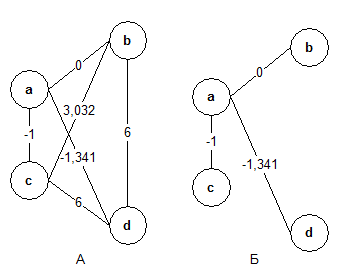


Рис.2. Граф, построенный по таблице 3 (А-исходный, Б-результат).

*Таблица 4. Количество вершин 3*

|  |  |  |
| --- | --- | --- |
| Ребро | Вес | Результат |
| a-b | -2 | a-b: -2 |
| a-c | 4 | b-c: 0.23 |
| b-c | 0,23 |  |

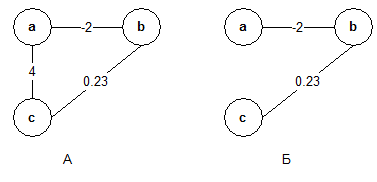


Рис.3. Граф, построенный по таблице 3 (А-исходный, Б-результат).

1. **Результаты решения задачи**

При выполнении программы были получены результаты, совпадающие со значениями, приведенными в пункте 3. Контрольные примеры. Результат выполнения контрольных примеров алгоритма Прима ЭВМ представлен на скриншотах ниже.

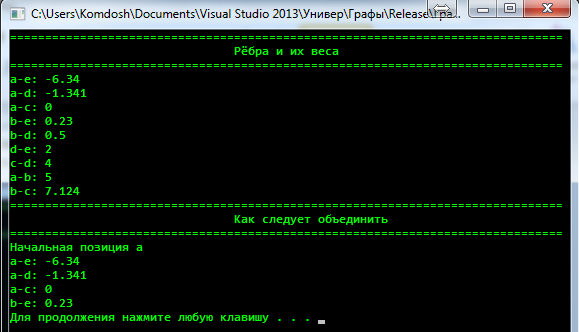


Рис. 4. Результат выполнения алгоритма Прима контрольный пример таблица 2.

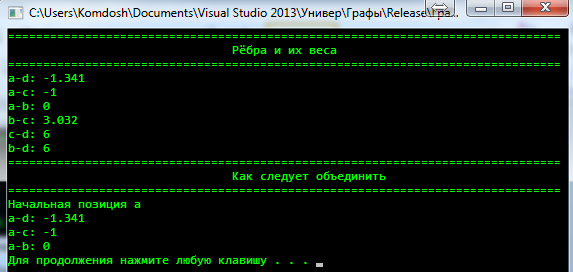


Рис. 5. Результат выполнения алгоритма Прима контрольный пример таблица 3.

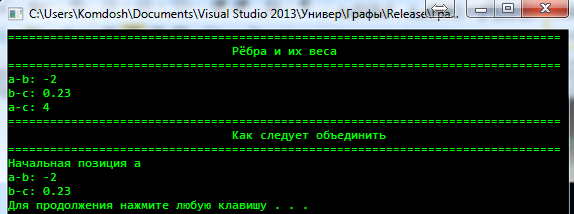


Рис. 6. Результат выполнения алгоритма Прима контрольный пример таблица 4.

**Вывод**

При выполнении курсовой работы были получены практические навыки работы с графами на языке программирования «C\C++». Также был закодирован оптимальный алгоритм, для поиска минимального остовного дерева, Прима. Временная сложность обработки алгоритма совпала с ожидаемой и равна O(n\*log n).

**Список используемых источников**

1. Алгоритмы и структуры данных: методические указания к лабораторным работам, практическим занятиям и курсовому проектированию. Федеральный образовательный стандарт / сост.: П.Г. Колинько. - СПб.: Изд-во СПБГЭТУ "ЛЭТИ", 2014. - 63 с.
2. Освой С++ самостоятельно за 21 день. Сиддхартха Рао. 688 стр., с ил.; ISBN 978-5-8459-1825-3; 7 издание.
3. http://stackoverflow.com – Сайт вопросов и ответов по программированию.
4. http://cyberforum.ru – Форум программистов и сисадминов.

**Приложение**

К данному отчёту прилагаются файлы на электронном носителе:

* Алгоритм Прима.cpp – исходный код программы
* Classes.h – заголовочный файл с классами
* matr – файл с тестом