Федеральное государственное автономное образовательное учреждение

высшего образования

«Нижегородский государственный университет

им. Н.И. Лобачевского»

Институт информационно-вычислительных технологий математики и механики.

Отчёт по лабораторной работе

Поиск пути минимальной длины в графе.

Выполнил:

студент ф-та ИИТММ гр. 8203-2

Кипаренко И.В

Проверил:

ассистент каф. МО ЭВМ, ИИТММ

Сиднев А.А.

Нижний Новгород

2016 г.

**Содержание**

[Введение](#h.gjdgxs)

[Постановка задачи](#h.30j0zll)

[Руководство пользователя](#h.1fob9te)

[Руководство программиста](#h.3znysh7)

[Описание структур данных](#h.3znysh7)

[Описание алгоритмов](#h.2et92p0)

Описание структуры программы

[Заключение](#h.3dy6vkm)

Литература

**Введение**

Развитие технологий спутниковой навигации, а также вычислительной техники и телекоммуникации обусловили резкий рост интереса к навигационному программному обеспечению. В настоящий момент актуально создание интеллектуальных систем, способных прокладывать оптимальные маршруты, просчитывать их время, предсказывать маршруты пользователей в условиях меняющейся дорожной обстановки, учитывая дорожные ограничения, привычки пользователей и другую доступную информацию.

В математике сети дорог представляются взвешенным графом. Граф — это конструкция из вершин и соединяющих их ребер. Абсолютно неважно, какой вид имеют рёбра, и как вершины расположены в пространстве. В транспортной сети населенные пункты (или перекрестки) — это вершины графа, ребра — дороги, веса ребер — расстояния по этим дорогам.

Для взвешенных графов предлагается множество алгоритмов, например, популярный алгоритм Дейкстры для поиска кратчайшего пути от одной вершины до другой и всевозможных. Алгоритм Дейсктры можно расширить, если для транспортной сети необходимо найти не только кратчайший путь от одного населённого пункта до другого, но и, например, количество дорог, которые по этому маршруту требуется пройти.

**Постановка задачи**

Дорожная сеть города представляет собой N перекрёстков, соединённых M дорогами с двусторонним движением.

Стартовый перекрёсток (начало маршрута) - 1, финальный перекрёсток - N.

По заданной карте города вычислите длину кратчайшим по суммарной длине маршрута и количество дорог, которое удалось посмотреть команде. Из всех таких маршрутов выбрать тот, который включает в себя наибольшее количество дорог.

**Руководство пользователя**

Входные данные

Файл с представлением графа в следующем формате:

* Первая строка: N – число вершин, M – число ребер.
* Дальше строки характеризующие ребра:  
  Начальная вершина Конечная вершина Вес ребра.  
  (вершины нумеруются с 1)

Выходные данные

Длина минимального пути из 1 вершины в N и количество ребер на этом пути.

**Руководство программиста**

***Описание структуры программы***

Программа состоит из :

1. Библиотеки, с реализацией класса d - куч (PHeap).
2. Библиотеки, с реализацией класса АВЛ - деревьев (AVL\_Tree).
3. Проекта, реализующего алгоритм Дейкстры (X\_Dijkstra\_Sample)

***Описание структур данных***

Класс PHeap, унаследованный от класса приоритетной очереди P\_Queue реализует d – кучи, с дополнительными индексами, на основе массива.

Класс AVL\_Tree, унаследованный от класса приоритетной очереди P\_Queue реализует сбалансированные деревья, с дополнительными индексами, на основе бинарных деревьев.

У класса P\_Queue есть методы :

* min\_top() , который возвращает минимальный ранг в очереди.
* min\_pop(), который удаляет элемент с минимальным рангом из очереди.
* add(), который добавляет элемент в очередь.
* changePriority() , который изменяет приоритет, у элемента с данным рангом и индексом.
* getRank(), который возвращает ранг элемента, с данным индексом.

***Описание алгоритмов***

В программе был реализован алгоритм Дейкстры, с дополнительным отслеживанием количества ребер в минимальном пути, с применением приоритетной очереди. Алгоритм Дейкстры, находит все минимальные пути от данной вершины к остальным.

Алгоритм :

1. Создаём приоритетную очередь хранящую длину путей от 1 вершины к остальным. Для само1 первой вершины храним 0, для остальных некое число обозначающее что пути ещё нет (у нас INT\_MAX)
2. Создаём массив хранящий сколько дорог в каждом таком пути, и массив с предыдущими вершинами в каждом пути.
3. Выбираем вершину с минимальной длиной пути. Берем от неё все соседние ребра, и если мы нашли более короткий пути к другим вершинам, то изменяем, длину этого пути и все остальные параметры.
4. Шаг 3, пока не кончатся вершины.
5. Готово. Пути найдены.

**Заключение**

В ходе работы была полностью реализована поставленная задача. Была написана программа, выполняющая поиск кратчайшего маршрута и подсчитывающая количество дорог, которые содержит данный маршрут. Абстрактный тип данных очередь с приоритетами реализована с помощью структур данных d - куча и АВЛ - дерево. Были проведены расчёты с использованием данных из контрольных файлов. Разработано консольное приложение, демонстрирующее результат работы алгоритма для двух вариантов реализации приоритетной очереди.

Результаты для контрольных тестов в виде пары чисел (P, Q) — длина кратчайшего пути от старта до финиша и максимальное число дорог, которые может включать в себя кратчайший путь.

016 – (3,3)

017 – (799200, 999)

018 – (109,99)

019 – (392883,999)

**Литература**

1. Столлингс, В. Структурная организация и архитектура компьютерных систем, 5-е изд.: Пер. с англ. — М.: Издательский дом «Вильямс», 2002. — 896 с.: ил. — Парал. тит. англ.
2. Johnson M. Superscalar Microprocessor Design. — Englewood Cliff, New Jersey: Prentice Hall, 1991.
3. Касперски К. Техника оптимизации программ. Эффективное использование памяти. — СПб.: БХВ-Петербург, 2003. — 464 с.: ил.
4. Stone H. High performance Computer Architecture. — Reading, MA: Addison-Wesley, 1993.
5. Tullsen D.M., Eggers S.J. Effective Cache Prefetching on a Bus-Based Multiprocessor. — ACM Transactions on Computer Systems, pp. 57-88, Feb 1995.
6. Chandra D., Guo F., Kim S., Solihin Y. Predicting inter-thread cache contention on a chip multi-processor architecture. — Proceedings of the 11th International Symposium on High Performance Computer Architecture (HPCA), pp. 340–351, Feb 2005.
7. Press W., Teukolsky S., Vetterling W., Flannery B. Numerical Recipes in C. The Art of Scientific Computing. Second Edition. — Cambridge University Press, 1992.
8. Камаев А.М., Сиднев А.А., Сысоев А.В. Об одном подходе к анализу эффективности приложений // Труды 50-й научной конференции МФТИ «Современные проблемы фундаментальных и прикладных наук»: Часть I. Радиотехника и кибернетика. - М.: МФТИ, 2007.
9. Debugging and performance monitoring. Intel® 64 and IA-32 Architectures Software Developer’s Manual. Volume 3B: System Programming Guide, Part 2. May 2007. — [http://www.intel.com/products/processor/manuals/]
10. Юнаковский А.Д. Начала вычислительных методов для физиков. – Н. Новгород: ИПФ РАН, 2007.