Федеральное государственное автономное образовательное учреждение  
высшего образования

«Нижегородский государственный университет

им. Н.И. Лобачевского»

Институт информационных технологий математики и механики

**Отчёт по лабораторной работе**

**«Дороги: алгоритм Дейкстры»**

**Выполнила:**

студентка группы 0823-2

Кабалова А.А.

**Проверил:**

ассистент каф. МОСТ института ИТММ

Сиднев А.А.

Нижний Новгород

2016 г.

Оглавление

[Введение 3](#_Toc452904903)

[Постановка задачи 4](#_Toc452904904)

[Структура проекта 5](#_Toc452904905)

# Введение

С каждым днем автомобильные gps навигаторы получают все большее распространение среди людей. Нахождение кратчайшего пути — одна из главных функций этого устройства. Алгоритмов, отвечающих за работу навигаторов множество. Один из них – это алгоритм Дейкстры.

Пусть есть две точки: начало и конец маршрута. Также есть карта, представляющая собой большой массив векторов. Нужно использовать его и рассчитать оптимальный маршрут.

Как раз для таких случаев используется алгоритм Дейкстры для расчета оптимального пути между двумя узлами взвешенного графа или сети. Векторная карта представляет собой взвешенный граф, в качестве ребер которого выступают векторы, а их концы можно рассматривать в качестве узлов.

# Постановка задачи

Дан взвешенный неориентированный граф, каждое ребро которого представляет собой дорогу, а каждая вершина – перекрёсток. Таким образом, необходимо решить следующую задачу: реализовать приоритетную очередь с использованием структур данных d-куча и АВЛ-дерево. С помощью приоритетной очереди реализовать алгоритм Дейкстры поиска кратчайшего пути от стартовой вершины до финальной. Вычислить длину кратчайшего по суммарной длине маршрута и количество дорог, из которых состоит этот маршрут. Из всех оптимальных маршрутов нужно выбрать тот, который включает в себя наибольшее количество дорог. Разработать консольное приложение для демонстрации результатов эксперимента.

# Структура проекта

Описание структур данных

Приоритетная очередь — абстрактный тип данных в программировании, поддерживающий две обязательные операции — добавить элемент и извлечь максимум. Предполагается, что для каждого элемента можно вычислить его приоритет — действительное число или в общем случае элемент линейно упорядоченного множества.

Приоритетная очередь естественным образом используется в таких задачах, как сортировка элементов массива, поиск во взвешенном неориентированном графе минимального остовного дерева, поиск кратчайших путей от заданной вершины взвешенного графа до его остальных вершин, и во многих других.

Представление приоритетной очереди с помощью d-кучи

Представление приоритетной очереди с помощью d -кучи основано на использовании так называемых завершенных d -арных деревьев (d ≥ 2).

Завершенное d -арное дерево — это корневое дерево со следующими свойствами:

1. Каждый внутренний узел (то есть узел, не являющийся листом дерева), за исключением, быть может, только одного, имеет ровно d потомков. Один узел-исключение может иметь от 1 до d - 1 потомков.
2. Если k — глубина дерева, то для любого i такое дерево имеет ровно di узлов глубины i.
3. Количество узлов глубины k в дереве глубины k может варьироваться от 1 до dk. Это свойство является следствием первых двух.
4. Узлы завершенного d -арного дерева принято нумеровать следующим образом: корень получает номер 0, потомки узла с номером i получают номера: i \* d + 1, i \* d + 2, … , i \* d + d. Такая нумерация удобна тем, что позволяет разместить узлы дерева в массиве в порядке возрастания их номеров, при этом позиции потомков любого узла в массиве легко вычисляются по позиции самого узла. Так же легко по позиции узла вычислить позицию его родителя. Так, для узла, расположенного в позиции i, родительский узел располагается в позиции (i - 1) div d, где div — операция деления нацело.

В изображении завершенного d -арного дерева узлы одинаковой глубины удобно располагать на одном уровне, при этом потомки одного узла располагаются слева направо в порядке объявленных номеров. При таком рисовании нижний уровень заполняется, возможно, не полностью.

АВЛ-деревья

АВЛ-дерево — это прежде всего двоичное дерево поиска, ключи которого удовлетворяют стандартному свойству: ключ любого узла дерева не меньше любого ключа в левом поддереве данного узла и не больше любого ключа в правом поддереве этого узла. Особенностью АВЛ-дерева является то, что оно является сбалансированным в следующем смысле: для любого узла дерева высота его правого поддерева отличается от высоты левого поддерева не более чем на единицу.

В процессе добавления или удаления узлов в АВЛ-дереве возможно возникновение ситуации, когда высота, например, левого поддерева оказывается больше высоты правого поддерева больше, чем на 1, т.е. возникает расбалансировка поддерева. Для выправления ситуации применяются повороты вокруг тех или иных узлов дерева.

# Алгоритмы

Нахождение кратчайших путей в графе

Входные данные:

Граф G со взвешенными ребрами (под весами можно понимать длины ребер, если речь идет о геометрическом графе, или любые другие числовые характеристики ребер). Пусть L(i, j) — вес ребра ( i,j ).

Стартовая вершина s (вершина, от которой вычисляются расстояния до всех остальных вершин).

Выходные данные:

Массив dist [1…n], dist[i] — кратчайшее расстояние от вершины s до вершины i ).

Массив up[1…n], up[i] — предпоследняя вершина в кратчайшем пути из вершины s в вершину i ).

Приводимый ниже алгоритм Дейкстры корректно решает задачу для графов с неотрицательными весами вершин.

**Алгоритм Дейкстры**

1. Заполнить массив up[1…n] нулями.
2. Каждой вершине i приписать в качестве ключа dist[i] — максимально возможное число (оно должно быть больше, чем длина наибольшего из кратчайших путей в графе; в процессе вычислений это число будет уменьшаться и в итоге заменится на длину кратчайшего пути из вершины s в вершину i ).
3. Организовать приоритетную очередь из вершин графа, взяв в качестве ключей величины dist[i], i= 1, 2 … n.
4. Заменить ключ вершины s на 0.
5. Пока очередь не пуста, выполнять операции 6,7.
6. Выбрать (с удалением) из приоритетной очереди элемент r\_0 с минимальным ключом.
7. Для каждой вершины r, смежной с r\_0, выполнить операции 8, 9.
8. Вычислить величину delta = dist[r] – (dist[r0] + L (r0,r)).
9. Если delta > 0, то уменьшить ключ dist[r] элемента r на величину delta и заменить старое значение величины up[r] на r0.

# Заключение

В ходе работы была полностью реализована поставленная задача. Была написана программа, вычисляющая длину кратчайшего по суммарной длине маршрута и количество дорог, из которых состоит этот маршрут. Очередь с приоритетами реализована с помощью структур данных d-куча и АВЛ-дерево. Были получены результаты работы программы для четырех контрольных тестов. 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. Никлаус Вирт. Алгоритмы и структуры данных. Новая версия для Оберона. –ДМК Пресс, 2010
2. Использование материалов с сайта https://ru.wikipedia.org/wiki/ Информация о структурах данных d-куча и АВл-дерево.
3. Использование материалов с сайта https://habrahabr.ru/ Информация о структуре данных АВЛ-дерево, об алгоритме Дейкстры.
4. Использование материалов с сайта http://av-gps.com/articles/871-kratchajshij-put.html.
5. Использование материалов с сайта <http://www.intuit.ru/> Информация об абстрактном типе данных очередь с приоритетом и о структуре данных d-куча.