Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное   
учреждение высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Факультет информационных технологий математики и механики

**Отчет по лабораторной работе**

**Сравнение алгоритмов поиска кратчайших путей**

**Выполнил**:студент группы 381606-1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кутовой В.Н.

Подпись

**Научный руководитель**:

Пр,д.ф.-м.н.,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Малышев Д.С

Подпись

Нижний Новгород

2018

Содержание

[Введение 3](#_Toc532161577)

[Описание алгоритмов 4](#_Toc532161578)

[Описание программы 7](#_Toc532161579)

[Эксперименты 9](#_Toc532161580)

[Заключение 11](#_Toc532161581)

# Введение

**Задача о кратчайшем пути** — задача поиска самого короткого пути (цепи) между двумя точками (вершинами) на графе, в которой минимизируется сумма весов рёбер, составляющих путь.

Задача о кратчайшем пути является одной из важнейших классических задач теории графов. Сегодня известно множество алгоритмов для её решения.

У данной задачи существуют и другие названия: задача о минимальном пути или, в устаревшем варианте, задача о дилижансе.

Значимость данной задачи определяется её различными практическими применениями. Например, в GPS-навигаторах осуществляется поиск кратчайшего пути между двумя перекрестками. В качестве вершин выступают перекрестки, а дороги являются ребрами, которые лежат между ними. Если сумма длин дорог между перекрестками минимальна, тогда найденный путь самый короткий.

В данной лабораторной работе мы рассмотрим два алгоритма: алгоритм Дейкстры на метках, и алгоритм Дейксты на d-куче.

# Описание алгоритмов

**Алгоритм дейкстры на метках**

Каждой вершине из **V** сопоставим метку — минимальное известное расстояние от этой вершины до **a**. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

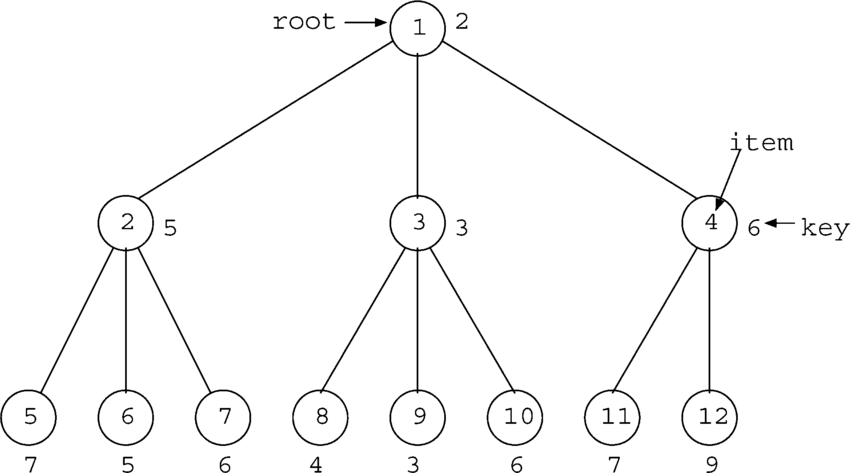
**Инициализация:** Метка самой вершины a полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от **a** до других вершин пока неизвестны. Все вершины графа помечаются как непосещённые.

**Шаг алгоритма:** Если все вершины посещены, алгоритм завершается. В противном случае, из ещё не посещённых вершин выбирается вершина **u**, имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, в которые ведут рёбра из u, назовём соседями этой вершины. Для каждого соседа вершины **u**, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки **u** и длины ребра, соединяющего **u** с этим соседом. Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину **u** как посещённую и повторим шаг алгоритма.

**Алгоритм Дейкстры на d-куче**

Алгоритм на d-куче не сильно отличается от классического алгоритма Дейкстры. Разница лишь в том, что мы изначально храним «серые» не помеченные вершины в куче. Наш выигрыш будет получаться на операциях поиска наименьшего элемента. В корне кучи мы будем держать как раз наименьший элемент. В остальном алгоритмы идентичны.

Пример дерева для 3-кучи:



Куча – представление взвешенного множества в виде корневого дерева, узлам которого ставятся во взаимно однозначное соответствие элементы рассматриваемого множества.

Соответствие между узлами дерева и элементами множества называется кучеобразным, если: ключ элемента, приписанного узлу, не превосходит ключей, приписанных его потомкам.

Завершенное d-арное дерево - корневое дерево.

Свойства:

* Каждый внутренний узел имеет ровно d потомков. Исключение - один узел, имеющий от 1 до d - 1 потомков
* На глубине i ровно узлов (1 <= k <= n -1),

k - глубина дерева;

* Количество узлов глубины k в дереве глубины k варьируется от 1 до d^k.

Глубина дерева – наибольшая глубина его узлов.

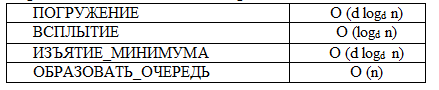
Глубина узла – число ребер в кратчайшем пути от корня до данного узла.

Высота узла - расстояние от него до наиболее далекого потомка.

Основные операции над d-кучами:

* Всплытие - применяется для элемента x в узле i, нарушающего кучеобразный порядок, т.е если ключ элемента меньше ключа родителя.
* Погружение - применяется для элемента x в узле i, нарушающего кучеобразный порядок, т.е если ключ элемента больше ключа потомка.
* Вставка - добавление n+1 узла с номером n и применение операции всплытие.
* Удаление - перенос последнего элемента на место удаляемого элемента i и если узел i имеет родителя с большим ключом, то применяется операции всплытие(i), иначе операция погружение(i).
* Уменьшение ключа уменьшить ключ элемента в узле i на заданную константу и выполнение операции всплытие(i).

Временные оценки:



# Описание программы

***Программа состоит из трех модулей****:*

1. DHeap.py – модуль, содержащий реализацию д-кучи
2. Main.py – модуль, содежащий реализацию алгоритмов
3. Scripts.py – модуль со вспомогательными скриптами

Реализация д-кучи представляет собой класс, содержащий в себе следующие методы:

ascend(self, i) - всплытие

min\_child(self, i) - погружение

first\_child(self, i) – первый ребенок вершины

last\_child(self, i) – последний ребенок вершины

father(self, i) – отец вершины

descend(self, i) - погружение

get\_min(self) – извлечение минимума

make(self) - окучивание

Класс содержит слеудующие поля:

key = [] – массив ключей

name = [] – массив имен вершин

index = [] – массив индексов

d = int()

n = int()

name1 = int()

key1 = int()

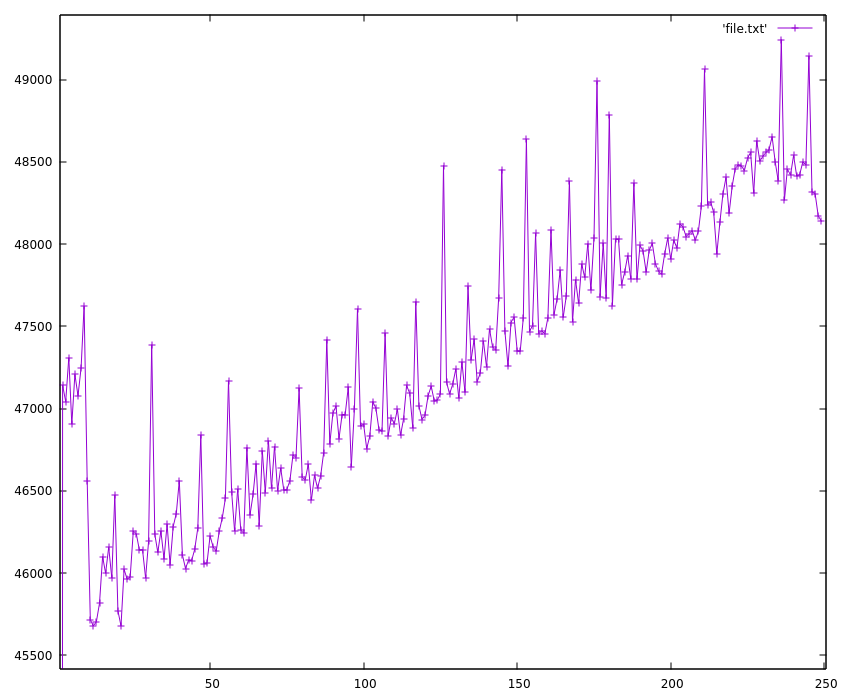
Представим d-кучу массивом имен **name[1..n]** и массивом ключей **key[1..n]** так, что **key[i]** является текущей оценкой длины кратчайшего пути от вершины **s** к вершине name[i]. В данном алгоритме также используется массив **index[1..n]**, который должен поддерживаться так, чтобы **index[name[i]]= i** при i=1, …, n.

Временная сложность алгоритма Дейкстры, реализованного на основе d-кучи, где , оценивается сверху величиной , так как алгоритм производит n изъятий и не более m всплытий, каждое из которых осуществляется за время .

# Эксперименты

В качестве первого вычислительного эксперимента для тестирования было проведено по 100 запусков алгоритма Дейкстры на d-куче для каждого значения параметра d от 2 до 250.

График зависимости времени работы алгоритма от значения параметра d:



Пики на графике предположительно вызваны переключением контекста процессора, чего избежать не удалось. Не смотря на это, на графике прослеживается закономерность: с увеличением значения параметра d –время работы алгоритма увеличивается.

С увеличением значения d параметра достигается незначительное ускорение выполнения операции «всплытие», так как уменьшается высота дерева, но происходит значительное замедление выполнения операции «погружение», так как родительскому узлу приходится выполнять больше сравнений с ключами потомков для выполнения операций.

Во втором эксперименте было проведено сравнение времени работы алгоритмов. Были получены следующие данные:

|  |  |  |
| --- | --- | --- |
| Кол-во вершин в графе | Дейкстра на d-куче (мкс) | Дейкстра на метках(мкс) |
| 100 | 15610 | 16658 |
| 500 | 62510 | 78124 |
| 1000 | 254156 | 294156 |
| 2000 | 1217445 | 1310141 |

# Заключение

В ходе выполнения лабораторной работы, мы ознакомились с методами нахождения кратчайших путей в графе от заданной вершины до остальных.

Разработали две структуры данных Graph и Dheap(Граф и d-куча). С минимальным необходимым функционалом для работы алгоритмов. При работе с d-кучей провели исследование для нахождения оптимального d параметра для работы алгоритма Дейкстры.

Выполнены реализации алгоритмов Дейкстры на метках и Дейкстры на куче.

Алгоритм Дейкстры метках показал худшие результаты по сравнению с алгоритмом на d-куче, но отставание было не радикальным, что легко объяясняется тем, что ускорение алгоритм на d-куче получает только при поиске минимальной вершины, потом же также приходится обходить все вершины для корректировки расстояний. Но не смотря на это с увеличением числа вершин алгоритм на куче получал все больший прирост производительности.

В результате работы все поставленые цели были достигнуты, эксперименты проведены и получены удовлетворительные результаты. Были приобретены навыки работы с графами любого размера