МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Институт информационных технологий математики и механики**

**ЛАБОРАТОРНАЯ РАБОТА**

**«Поиск кратчайших путей в графе. Алгоритм Дейкстра на основе массива меток и D-куч»**

**Выполнил**: студент группы 081506-1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Трифонов А.Н.

Подпись

**Проверил:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Малышев Д.С.

Подпись

Нижний Новгород

2017

**Содержание**

* **Введение………………………………………………………………...3**
* **Алгоритм Дейкстры…………………………………………………...5**
* **D-кучи……………..…………………………………………………….6**
* **Вычислительный эксперимент………………………………………8**
* **Заключение……………………………………………………….…….9**

**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.

**Алгоритм Дейкстры**

Алгоритм Дейкстрыпредназначен для решения задачи поиска кратчайшего пути на графе. Для заданного ориентированного взвешенного графа с неотрицательными весами алгоритм находит кратчайшие расстояния от выделенной вершины-источника до всех остальных вершин графа.

**Описание алгоритма**

Пусть задан граф G=(V,E)с весами рёбер f(e)и выделенной вершиной-источником а.

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.

**Доказательство корректности**

Пусть l(v) — длина кратчайшего пути из вершины a в вершину v. Докажем по индукции, что в момент посещения любой вершины d(z)=l(z)}.

База. Первой посещается вершина a. В этот момент d(a)=l(a)=0.

Шаг. Пусть мы выбрали для посещения вершину z ≠ a. Докажем, что в этот момент d(z)=l(z). Для начала отметим, что для любой вершины v всегда выполняется d(v) ≥ l(v) (алгоритм не может найти путь короче, чем кратчайший из всех существующих). Пусть P — кратчайший путь из a в z. Вершина z находится на P и не посещена. Поэтому множество непосещённых вершин на P непусто. Пусть y — первая непосещённая вершина на P, x — предшествующая ей (следовательно, посещённая). Поскольку путь P кратчайший, его часть, ведущая из a через x в y, тоже кратчайшая, следовательно, l(y)=l(x)+w(xy). По предположению индукции, в момент посещения вершины x выполнялось d(x)=l(x), следовательно, вершина y тогда получила метку не больше чем d(x)+w(xy)=l(x)+w(xy)=l(y). Следовательно, d(y)=l(y). Если z=y, то индукционный переход доказан. Иначе, поскольку сейчас выбрана вершина z, а не y, метка z минимальна среди непосещённых, то есть d(z) ≤d(y)=l(y) ≤l(z). Комбинируя это с d(z) ≥l(z), имеем d(z)=l(z), что и требовалось доказать.

Поскольку алгоритм заканчивает работу, когда все вершины посещены, в этот момент d=l для всех вершин.

**Сложность алгоритма Дейкстры**

Сложность алгоритма Дейкстры зависит от способа нахождения вершины v, а также способа хранения множества непосещённых вершин и способа обновления меток. Обозначим через n количество вершин, а через m — количество рёбер в графе G.

В простейшем случае, когда для поиска вершины с минимальным d[v] просматривается всё множество вершин, а для хранения величин d используется массив, время работы алгоритма есть O(n2). Основной цикл выполняется порядка n раз, в каждом из них на нахождение минимума тратится порядка n операций. На циклы по соседям каждой посещаемой вершины тратится количество операций, пропорциональное количеству рёбер m (поскольку каждое ребро встречается в этих циклах ровно дважды и требует константное число операций). Таким образом, общее время работы алгоритма O(n2+m), но, так как m≤ n(n-1) m, оно составляет O(n2).

**D-кучи**

Для улучшения сложности алгоритма Дейкстры необходимо использовать такую структуру данный в которой операции нахождения элемента с минимальным весом и изменение веса элемента выполняются за наименьшее время. В качестве такой структуры можно использовать D-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.

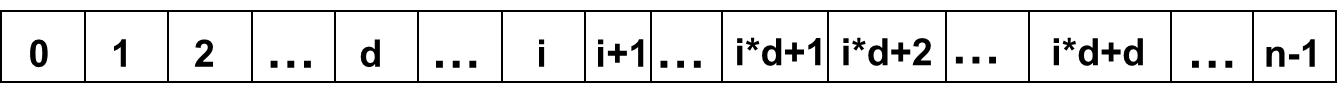
Завершенное d-арное дерево - корневое дерево.

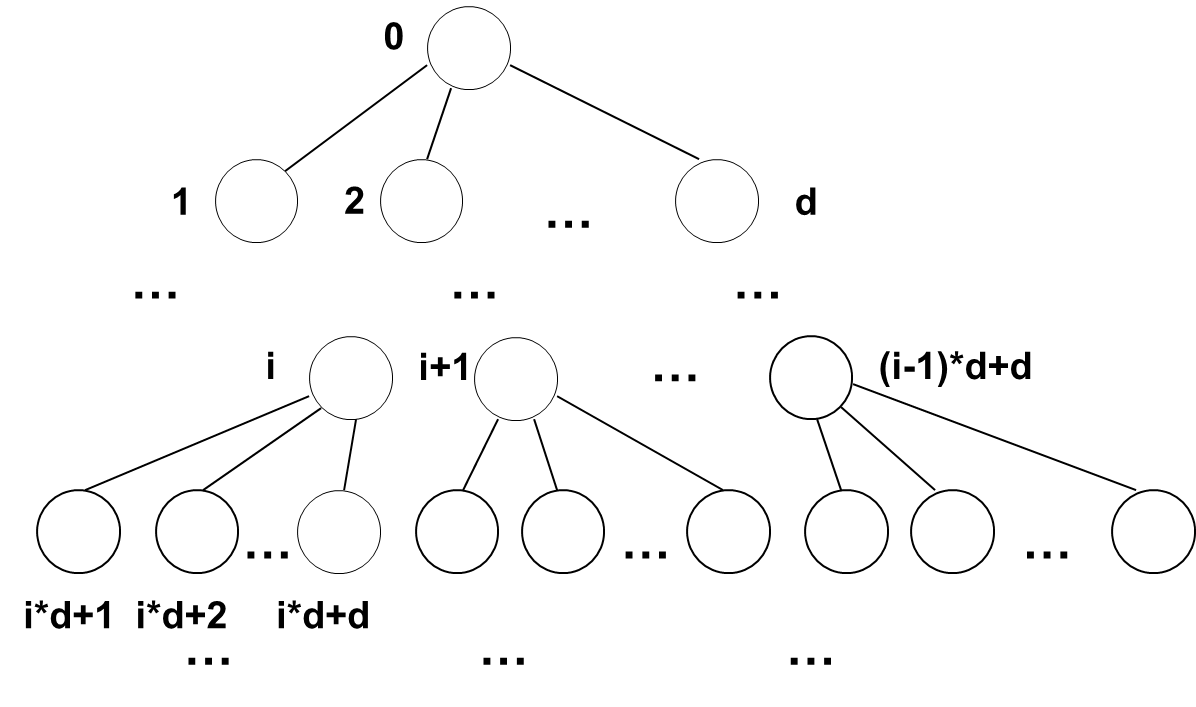
Свойства:

Каждый внутренний узел имеет ровно d потомков. Исключение - один узел, имеющий от 1 до d - 1 потомков.

На глубине i ровно di узлов (1≤k≤n-1), k - глубина дерева.

Количество узлов глубины k в дереве глубины k варьируется от 1 до dк.



****

**Основные операции над d-кучами:**

* Всплытие
* Погружение
* Вставка
* Удаление
* Уменьшение ключа
* Окучивание

Всплытие(i): Применяется для элемента x в узле i, нарушающего кучеобразный порядок, т.е. если ключ элемента меньше ключа родителя. Вычислительная сложность

Погружение(i): Применяется для элемента x в узле i, нарушающего кучеобразный порядок, т.е. если ключ элемента больше ключа потомка. Вычислительная сложность

Вставка(key\_insert): Добавление n+1 узла с номером n и применение операции Всплытие(n). Вычислительная сложность

Удаление(i):

Перенести последний элемент на место удаляемого элемента с номером i;

Если узел i имеет родителя с большим ключом, то применяется операции Всплытие(i), иначе операция Погружение(i)

Вычислительная сложность

Уменьшение ключа(i,k):

Уменьшить ключ элемента в узле i на заданную константу k = const ;

Выполнить операцию Всплытие(i).

Вычислительная сложность

Окучивание:

Применение операции Погружение(i) по очереди к узлам n-1…1.

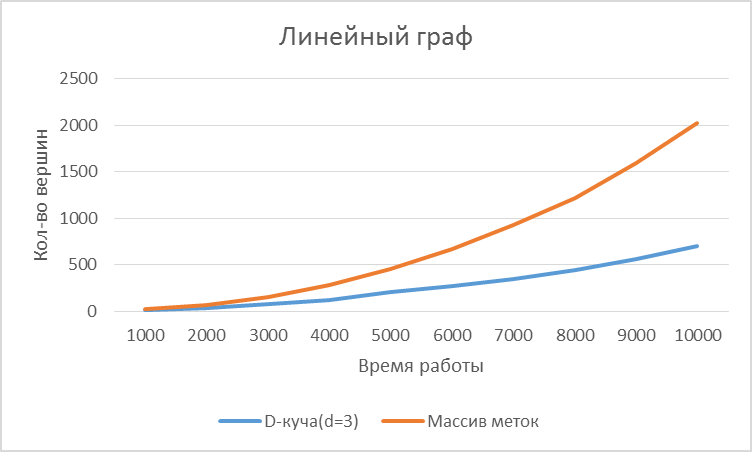
Вычислительная сложность

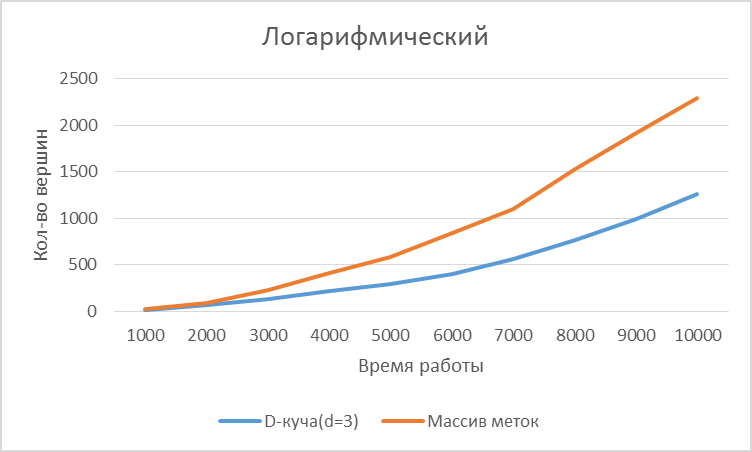
**Сложность алгоритма Дейкстры при использовании D-куч.**

Так как цикл выполняется порядка n раз, а сложность изъятия минимума , а количество релаксаций (смен меток) не больше m со сложностью , время работы такой реализации —

**Вычислительный эксперимент**

В ходе данной работы была реализована компьютерная программа для нахождения кратчайших путей в графе с помощью алгоритма Дейкстры с использованием массива меток и D-куч. Был проведен вычислительный эксперимент с целью проверки теоретической сложности алгоритма на графах различной плотности. Были использованы графы 3-х видов плотности: линейный(m=), логарифмический (m=), полный (m=)

****



**Заключение**

В ходе данной работы был изучен алгоритм Дейкстры для нахождения минимальный путей в графе и проанализирована его сложность при использовании различный структур данных. Была реализована компьютерная программа для нахождения кратчайших путей в графе с помощью алгоритма Дейкстры на основе массива меток и d-куч. Был проведен вычислительный эксперимент для сравнения теоретической и действительной сложности алгоритма. В результате, теоретическая сложность была подтверждена.