Алгоритм Дейкстры

Генов А.В.

Аннотация.
В отчёте описан алгоритм нидерландского учёного Эдсгера Дейкстры. Представлена реализация алгоритма на языке C++.

Об авторах:
Генов Александр Владимирович,
НИУ ВШЭ МИЭМ им. А.Н.Тихонова
ул. Мясницкая, 20, г.Москва, Россия, 101000
e-mail: avgenov@edu.hse.ru

Введение

Алгоритм Дейкстры (англ. Dijkstra's algorithm) — алгоритм на графах, изобретённый нидерландским учёным Эдсгером Дейкстрой в 1959 году. Находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса.

1. Формулировка задачи

Примеры

Вариант 1. Дана сеть автомобильных дорог, соединяющих города Московской области. Некоторые дороги односторонние. Найти кратчайшие пути от города Москвы до каждого города области (если двигаться можно только по дорогам).

Вариант 2. Имеется некоторое количество авиарейсов между городами мира, для каждого известна стоимость. Стоимость перелёта из A в B может быть не равна стоимости перелёта из B в A. Найти маршрут минимальной стоимости (возможно, с пересадками) от Копенгагена до Барнаула.

Формальное определение.
Дан взвешенный ориентированный граф $G(V, E)$ без дуг отрицательного веса. Найти кратчайшие пути от некоторой вершины a графа G до всех остальных вершин этого графа.

2. Неформальное объяснение

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.

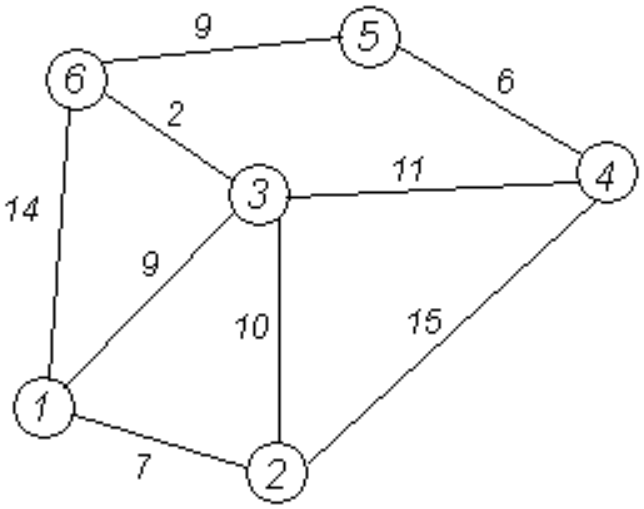

Рис. 1. example graph

3. Алгоритм

3.1. Обозначения

1. V - множество вершин графа;
2. E - множество рёбер графа;
3. $w[ij]$ - вес (длина) ребра ij;
4. a - вершина, расстояния от которой ищутся;
5. U - множество посещенных вершин;
6. $d[u]$ - по окончании работы алгоритма равно длине кратчайшего пути из a до вершины u;
7. $p[u]$ - по окончании работы алгоритма содержит кратчайший путь из a в u;
8. v - вершина графа;

3.2. Псевдокод

Присвоим $d[a] \leftarrow 0,\ p[a] \leftarrow 0$
Для всех $u \in V$ отличных от a
    присвоим $d[u] \leftarrow \infty$
Пока $\exists v \notin U$
    Пусть $v \notin U$ – вершина с минимальным $d[v]$
    занесём v в U
    Для всех $u \notin U$ таких, что $vu \in E$
        если $d[u] > d[v] + w[vu]$ то
            изменим $d[u] \leftarrow d[v] + w[vu]$
            изменим $p[u] \leftarrow (p[v], u)$

Рис. 2. code

4. Сложность алгоритма

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

1. 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 $O(n^2)$. 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 $O(n^2 + m)$, но, так как $m \leq n(n-1)$, оно составляет $O(n^2)$.

2. Для разреженных графов (то есть таких, для которых m много меньше n^2) непосещённые вершины можно хранить в двоичной куче, а в качестве ключа использовать значения $d[i]$, тогда время удаления вершины из $\overline{U}$ станет $\log n$ при том, что время модификации $d[i]$ возрастёт до $\log n$. Так как цикл выполняется порядка n раз, а количество релаксаций (смен меток) не больше m, время работы такой реализации — $O(n \log n + m \log n)$.

3. Если для хранения непосещённых вершин использовать фибоначчиеву кучу, для которой удаление происходит в среднем за $O(\log n)$, а уменьшение значения в среднем за $O(1)$, то время работы алгоритма составит $O(n \log n + m)$.

5. Полезные ссылки

Прикрепляю ссылку на репозиторий сайта github.com, содержащий данный отчёт и реализацию алгоритма на языке C++.

https://github.com/snowissquare/Dijkstra.