Алгоритм Дейкстры

Задача:

Для заданного взвешенного графа $G = (V, E)$ найти кратчайшие пути из заданной вершины s до всех остальных вершин. Веса всех рёбер неотрицательны.

Содержание

Алгоритм

В ориентированном взвешенном графе $G = (V, E)$, вес рёбер которого неотрицателен и определяется весовой функцией $w : E \to \mathbb{R}$, алгоритм Дейкстры находит длины кратчайших путей из заданной вершины s до всех остальных.
В алгоритме поддерживается множество вершин U, для которых уже вычислены длины кратчайших путей до них из s. На каждой итерации основного цикла выбирается вершина $u \notin U$, которой на текущий момент соответствует минимальная оценка кратчайшего пути. Вершина u добавляется в множество U и производится релаксация всех исходящих из неё рёбер.

Псевдокод

```
func dijkstra(s):
    for v ∈ V
        d[v] = ∞
        used[v] = false
    d[s] = 0
    for i ∈ V
        v = null
        for j ∈ V                        // найдём вершину с минимальным расстоянием
            if !used[j] and (v == null or d[j] < d[v])
                v = j
        if d[v] == ∞
            break
        used[v] = true
        for e : исходящие из v рёбра     // произведём релаксацию по всем рёбрам, исходящим из v
            if d[v] + e.len < d[e.to]
                d[e.to] = d[v] + e.len
```

Обоснование корректности

Теорема:

Пусть $G = (V, E)$ — ориентированный взвешенный граф, вес рёбер которого неотрицателен, s — стартовая вершина. Тогда после выполнения алгоритма Дейкстры $d(u) = \rho(s, u)$ для всех u, где $\rho(s, u)$ — длина кратчайшего пути из вершины s в вершину u

Доказательство:

$\triangleright$

Докажем по индукции, что в момент посещения любой вершины u, $d(u) = \rho(s, u)$.

- На первом шаге выбирается s, для неё выполнено: $d(s) = \rho(s, s) = 0$
- Пусть для n первых шагов алгоритм сработал верно и на $n+1$ шагу выбрана вершина u. Докажем, что в этот момент $d(u) = \rho(s, u)$. Для начала отметим, что для любой вершины v, всегда выполняется $d(v) \geqslant \rho(s, v)$ (алгоритм не может найти путь короче, чем кратчайший из всех существующих). Пусть P — кратчайший путь из s в u, v — первая непосещённая вершина на P, z — предшествующая ей (следовательно, посещённая). Поскольку путь P кратчайший, его часть, ведущая из s через z в v, тоже кратчайшая, следовательно $\rho(s, v) = \rho(s, z) + w(zv)$. По предположению индукции, в момент посещения вершины z выполнялось $d(z) = \rho(s, z)$, следовательно, вершина v тогда получила метку не больше чем $d(z) + w(zv) = \rho(s, z) + w(zv) = \rho(s, v)$, следовательно, $d(v) = \rho(s, v)$. С другой стороны, поскольку сейчас мы выбрали вершину u, её метка минимальна среди непосещённых, то есть $d(u) \leqslant d(v) = \rho(s, v) \leqslant \rho(s, u)$, где второе неравенсто верно из-за ранее упомянутого определения вершины v в качестве первой непосещённой вершины на P, то есть вес пути до промежуточной вершины не превосходит веса пути до конечной вершины вследствие неотрицательности весовой функции. Комбинируя это с $d(u) \geqslant \rho(s, u)$, имеем $d(u) = \rho(s, u)$, что и требовалось доказать.
- Поскольку алгоритм заканчивает работу, когда все вершины посещены, в этот момент $d(u) = \rho(s, u)$ для всех u.

$\triangleleft$

Оценка сложности

В реализации алгоритма присутствует функция выбора вершины с минимальным значением d и релаксация по всем рёбрам для данной вершины. Асимптотика работы зависит от реализации.

Пусть n — количество вершин в графе, m — количество рёбер в графе.

	Время работы			Описание
	Поиск минимума	Релаксация	Общее	
Наивная реализация	$O(n)$	$O(1)$	$O(n^2+m)$	n раз осуществляем поиск вершины с минимальной величиной d среди $O(n)$ непомеченных вершин и m раз проводим релаксацию за $O(1)$. Для плотных графов ($m \approx n^2$) данная асимптотика является оптимальной.
Двоичная куча	$O(\log n)$	$O(\log n)$	$O(m\log n)$	Используя двоичную кучу можно выполнять операции извлечения минимума и обновления элемента за $O(\log n)$. Тогда время работы алгоритма Дейкстры составит $O(n\log n+m\log n)=O(m\log n)$
Фибоначчиева куча	$O(\log n)$	$O(1)$	$O(n\log n+m)$	Используя Фибоначчиевы кучи можно выполнять операции извлечения минимума за $O(\log n)$ и обновления элемента за $O(1)$. Таким образом, время работы алгоритма составит $O(n\log n+m)$.

На практике удобно использовать стандартные контейнеры (например, **std::set** или **std::priority_queue** в C++).
При реализации необходимо хранить вершины, которые упорядочены по величине d, для этого в контейнер можно помещать пару — расстояние-вершина. В результате будут храниться пары, упорядоченные по расстоянию.

Изначально поместим в контейнер стартовую вершину s. Основной цикл будет выполняться, пока в контейнере есть хотя бы одна вершина. На каждой итерации извлекается вершина с наименьшим расстоянием d и выполняются релаксации по рёбрам из неё. При выполнении успешной релаксации нужно удалить из контейнера вершину, до которой обновляем расстояние, а затем добавить её же, но с новым расстоянием.
В обычных кучах нет операции удаления произвольного элемента. При релаксации можно не удалять старые пары, в результате чего в куче может находиться одновременно несколько пар расстояние-вершина для одной вершины (с разными расстояниями). Для корректной работы при извлечении из кучи будем проверять расстояние: пары, в которых расстояние отлично от $d[v]$ будем игнорировать. При этом асимптотика будет $O(m\log m)$ вместо $O(m\log n)$.

Источники информации

- Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн Алгоритмы: построение и анализ — 2-е изд. — М.: «Вильямс», 2007. — с. 459. — ISBN 5-8489-0857-4
- MAXimal :: algo :: Нахождение кратчайших путей от заданной вершины до всех остальных вершин алгоритмом Дейкстры (http://e-maxx.ru/algo/dijkstra)
- Википедия — Алгоритм Дейкстры (https://ru.wikipedia.org/wiki/Алгоритм_Дейкстры)
- Wikipedia — Dijkstra's algorithm (https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm)

- Эта страница последний раз была отредактирована 16 января 2019 в 22:35.