Алгоритм Левита

Историческая справка

Алгоритм предложил московский математик, профессор Борис Юльевич Левит.

Принцип работы

Пусть d_i — текущая длина кратчайшего пути до вершины i. Изначально, все элементы d, кроме s-го равны бесконечности; $d[s]=0$.
Разделим вершины на три множества:

- M_0 — вершины, расстояние до которых уже вычислено (возможно, не окончательно),
- M_1 — вершины, расстояние до которых вычисляется. Делится на 2 множества: M'_1 - основная очередь, M''_1 - срочная очередь.
- M_2 — вершины, расстояние до которых еще не вычислено.

Изначально все вершины, кроме s помещаются в множество M_2. Вершина s помещается в множество M_1.
Шаг алгоритма: выбирается вершина u из M_1. Если срочная очередь не пуста, то из неё, иначе из основной. Для каждого ребра $uv \in E$ возможны три случая:

- $v \in M_2$, то v переводится в конец очереди M'_1. При этом $d_v \leftarrow d_u + w_{uv}$ (производится релаксация ребра uv),
- $v \in M_1$, то происходит релаксация ребра uv,
- $v \in M_0$. Если при этом $d_v > d_u + w_{uv}$, то происходит релаксация ребра uv и вершина v помещается в M''_1; иначе ничего не делаем.

В конце шага помещаем вершину u в множество M_0.
Алгоритм заканчивает работу, когда множество M_1 становится пустым.

Временная сложность

Сложность в худшем случае – O(N * M)

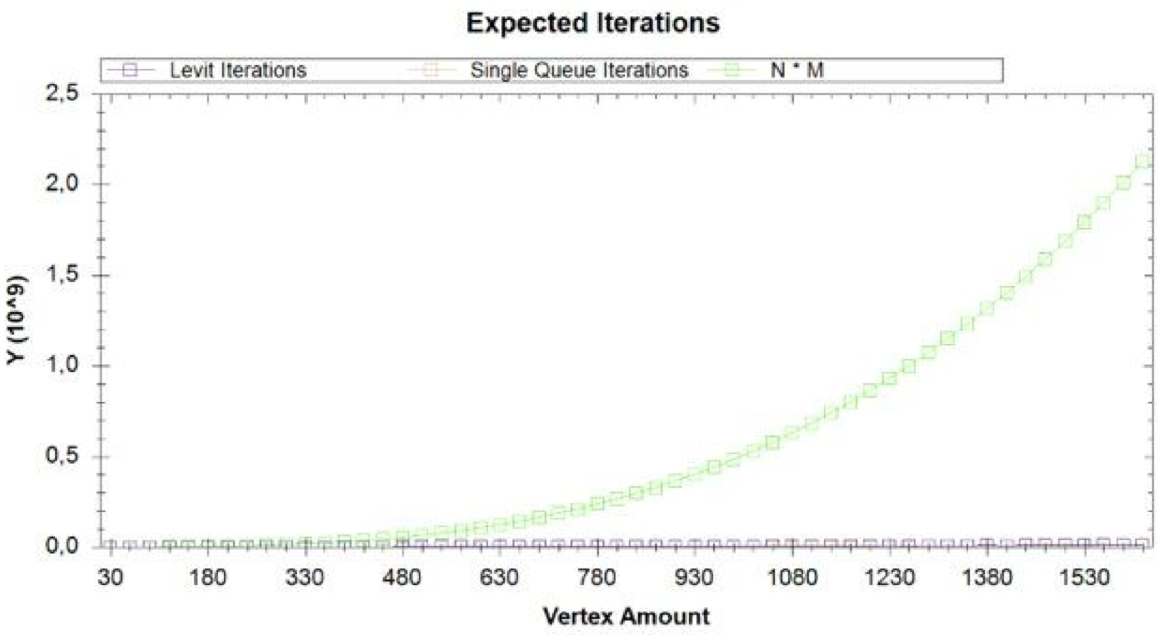
● Рассмотрим полный граф K_n с n вершинами и такими m рёбрами, идущими в лексикографическом порядке:

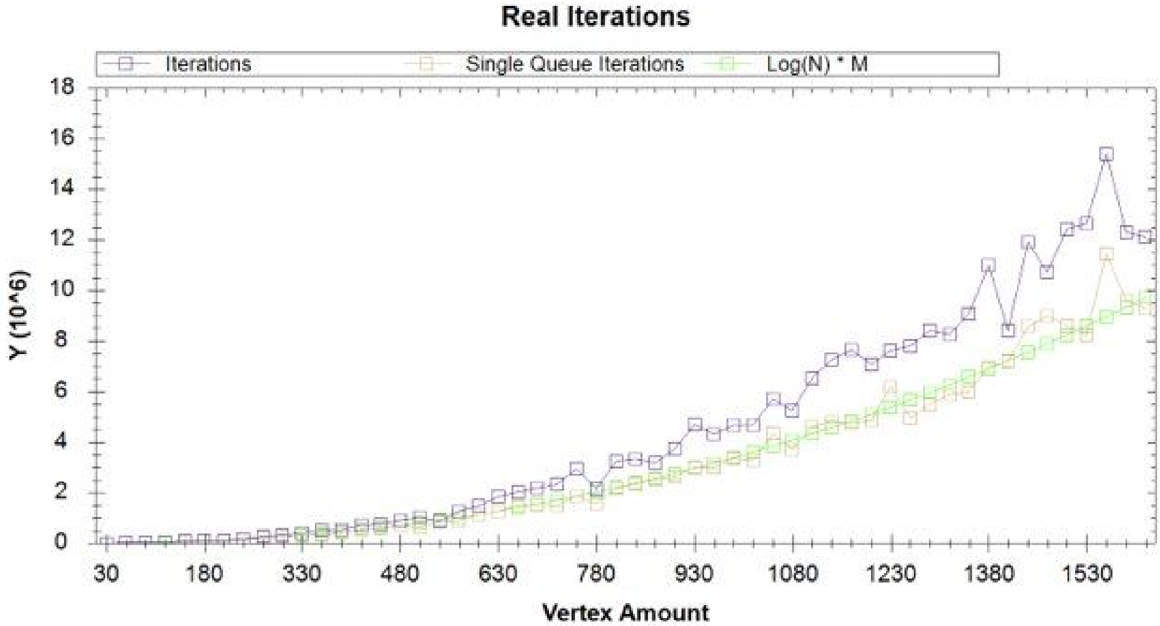
● для всех вершин $1<i<j\leq n$

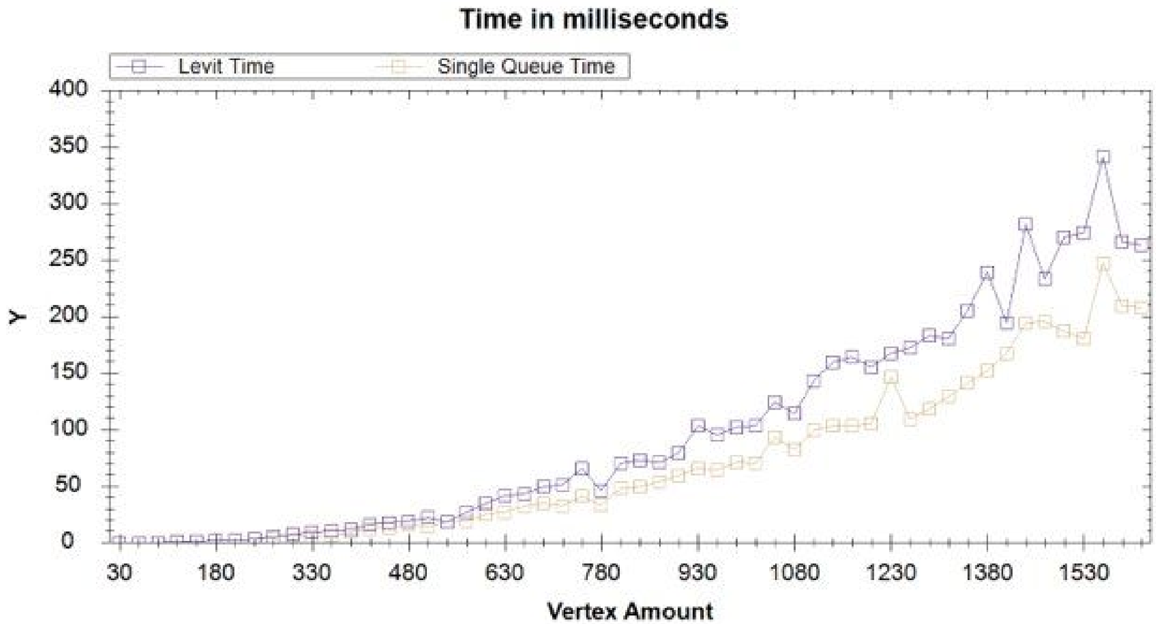
● вес ребра $(i,j)=j-i-1$, т.е. количество вершин между i и j; $w_{i,i+1}=0$, ребро $(1,n)$ веса 0, для всех вершин $1<i<n$ вес ребра $(1,i)=w_{1,i+1}+i-1$; от 1 до i вершины расстояние равно $\sum_{k=i-1}^{n-2} k$.

● Ясно, что кратчайший путь до каждой вершины равен 0, но в плохом случае алгоритм при подсчёте вершины i будет пересчитывать все вершины до неё (кроме первой). На 1 шаге в очередь положат вершины от 2 до n, причём вершину 1 из M_0 больше не достанут. На следующем шаге добавлений не произойдёт, так как вершины больше 2 уже в очереди. На 3 шаге алгоритм улучшит расстояние до вершины 2 на 1 (что видно из веса рёбер (1,2) и (1,3), равных $\sum_{k=1}^{n-2} k$ и $\sum_{k=2}^{n-2} k$ соответственно), так что её добавят в M_1'' и обработают на 4 шаге (релаксаций не происходит). На следующем шаге из обычной очереди достанут вершину 4, расстояние до неё, равное $\sum_{k=3}^{n-2} k$, на 2 меньше, чем расстояние до 2 и 3 вершин. Их добавят в срочную очередь, но так как $w_{24}-1=w_{34}$, то после подсчёта вершины 3 вершину 2 снова добавят в M_1''. Затем дойдёт очередь до вершины 5, что вызовет релаксацию предыдущих вершин 2,3,4, затем прорелаксируют вершины 2,3, и после вершина 2. Аналогично будут происходить релаксации всех вершин при обработке вершины i из очереди M_0. Таким образом, вершину i будут добавлять в срочную очередь $n-i$ раз (добавление вершин из очереди M_2 с номером больше i) + количество добавлений "старшей" вершины $i+1$. Количество добавлений вершины i составит $1+\sum_{k=1}^{n-i} k$, а сумма всех добавлений примерно составит $O(nm)$.

Графики







Выводы

- Лучше всего работает на графах геометрического происхождения.
- Работает с графами с отрицательным весом рёбер (кроме случая, когда есть отрицательные циклы)
- Модификация графа позволяет находить отрицательные циклы
- Простая реализация
- В среднем работает хуже, чем алгоритм Беллмана-Форда.

Список источников

1. Алгоритм Левита - Викиконспекты
2. Алгоритм Левита - Википедия
3. MAXimal - Алгоритм Левита нахождения кратчайших путей от заданной вершины до всех остальных вершин