Санкт-Петербургский государственный университет

Факультет прикладной математики – процессов управления

**Анализ сложности алгоритма Беллмана-Форда**

Работу выполнил Панюшин Даниил Васильевич группа 19.Б12-пу

* *Описание алгоритма*

Алгоритм Беллмана-Форда ищет длины кратчайших путей от исходной вершины до всех остальных вершин в взвешенном (ориентированном или нет) графе. Алгоритм применим также и к графам, содержащим рёбра отрицательного веса. При наличии отрицательного цикла, кратчайшего пути от некоторых вершин может не существовать (так как вес кратчайшего пути должен быть равен минус бесконечности).

Алгоритм носит имя двух американских учёных: Ричарда Беллмана и Лестера Форда. Форд фактически изобрёл этот алгоритм в 1956 г. при изучении другой математической задачи, подзадача которой свелась к поиску кратчайшего пути в графе, и Форд дал набросок решающего эту задачу алгоритма. В 1958 г. Беллман опубликовал статью, посвящённую конкретно задаче нахождения кратчайшего пути, в которой он чётко сформулировал данный алгоритм в современном виде.

Алгоритм вычисляет кратчайшие пути снизу-вверх. Сначала он вычисляет самые короткие расстояния (длиной в одно ребро), а затем кратчайшие пути длиной не более двух ребер и так далее. В простом пути может быть максимум *|V|-1* ребер, поэтому внешний цикл выполняется именно *|V|-1* раз, где *|V|-* мощность множества вершин графа. Если вычислить кратчайший путь с не более чем *i* ребрами, то итерация по всем ребрам гарантирует получение кратчайшего пути с не более чем *i + 1* ребрами.

* Математический анализ алгоритма

Алгоритм выполняет *|V|-1* итерацию, на каждой из которых происходит релаксация рёбер *|E|* (релаксациейребра (*u*, *v*) называется уменьшение значения d[*v*] до d[*u*] + *w*, если второе значение меньше первого, где d[*x*] –расстояние от вершины-источника до ребра *x*, а *w* – расстояние (вес) ребра (*u*, *v*)). В итоге получаем *O(|V||E|)* операций в худшем случае (*|E|* - мощность множества ребер графа).

* Входные и выходные данные

Объём входных данных: *O(|V|+|E|)*

В данной работе используется количество вершин (*V*), равное от 10 до 500 (включительно) с шагом 10.

Затраты памяти: *O(|V|)*

Выходные данные (могут быть нескольких видов):

* + для каждой вершины *v* исходного графа – последнее ребро, лежащее на кратчайшем пути от вершины *u* к *v*, или соответствующая вершина *w*;
  + для каждой вершины *v* исходного графа – суммарный вес кратчайшего пути от от вершины *u* к *v*.

Таким образом объём выходных данных: *O(|V|).*

* Единицы измерения трудоёмкости

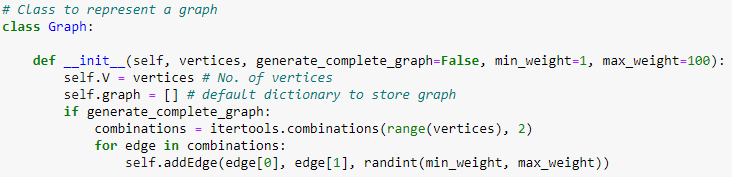
В рамках данной работы для измерения трудоемкости используется как время работы алгоритма, так и количество операций, которое он выполняет.

* Способ генерации входных данных

Используются полные графы со случайными весами на рёбрах (диапазон значений весов возможно изменить – по умолчанию от 1 до 100). Величины весов не влияют на сложность алгоритма.

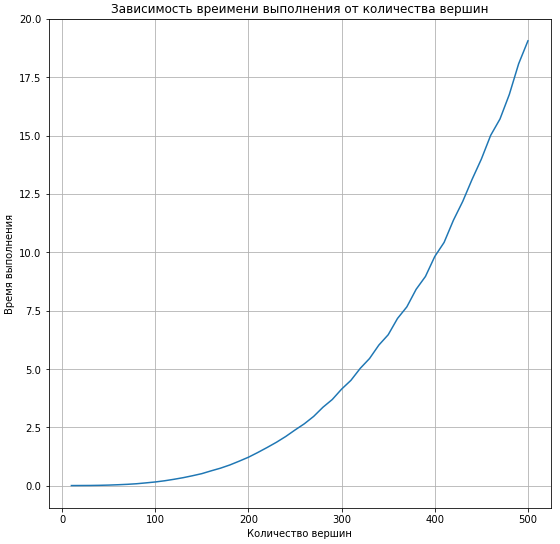
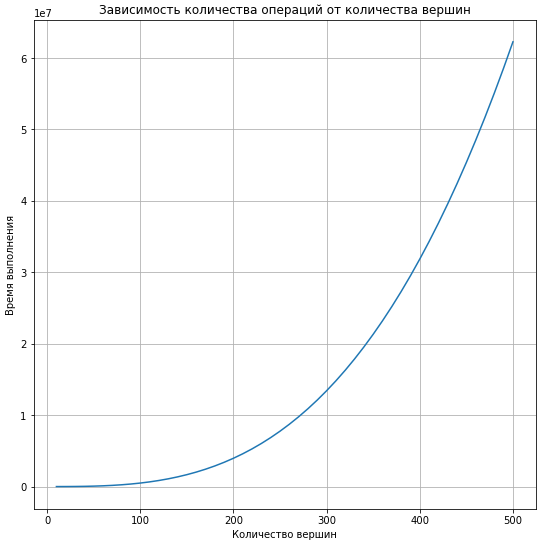
Так как при анализе используются полные графы (в которых *|V|-1* ребро), то сложность алгоритма *O(|V|2)*. Сложность не зависит от начальной точки, так как граф полный.

Генерация полного графа происходит в конструкторе класса, представляющего граф.

В каждом эксперименте создаётся новый граф с заданным количеством вершин.

* Код программы

Находится в репозитории [GitHub](https://github.com/DaniilDDDDD/algorithm_analysis)

* Эксперименты



Из данных, предоставленных на таблице видно, что при увеличении количества вершин вдвое, время выполнения и число операций увеличиваются в восемь раз.

*Результаты:*

Судя по полученным графикам и отношению результатов эксперимента при 2n и n вершинах можно сделать вывод, что время выполнения и количество операций алгоритма зависит от количества вершин графа нелинейно т.е. сложность алгоритма равна *O(n^3)*. Она отличается от теоретической сложности так как для экспериментов используются полные графы, а значит количество рёбер каждого графа равно n (n-1) / 2 (n – количество вершин). Таким образом сложность алгоритма равна *O(n^2 (n - 1) / 2) = O(n^3)*.

* Характеристики использованной среды и оборудования.
  + Вычисления произведены в среде Jupyther Notebook.
  + Версия Python: 3.7.11
  + Процессор: AMD Ryzen 7 3700X
* Список литературы
  + R. Bellman: On a Routing Problem // Quarterly of Applied Mathematics. 1958. Vol 16, No. 1. C. 87-90, 1958.
  + L. R. Ford, Jr., D. R. Fulkerson. Flows in Networks, Princeton University Press, 1962.
  + <https://e-maxx.ru/algo/ford_bellman>
  + <https://algowiki-project.org/ru/Алгоритм_Беллмана-Форда>
  + <https://habr.com/ru/company/otus/blog/484382/>