

Отчет об экспериментальном сравнении реализаций

Автор:

Шапошников Алексей, СПбГУ

23 сентября 2020 г

Анализ

При проведении эксперимента использовался компьютер с Windows 10 ->

WSL-Ubuntu 20.04; Intel Core i5-4690 CPU @ 3.50 GHz; 16 gb DDR4 RAM.

Для эксперимента был произведен анализ производительности вычисления регулярных запросов с помощью двух реализаций транзитивного замыкания: возведения в квадрат и умножения на матрицу смежности. В качестве графов были использованы следующие датасеты из пакета **refinedDataForRPQ**: *LUBM300*, *LUBM500*, *LUBM1M*,

LUBM1.5M, *LUBM1.9M*, *proteomes*, *uniprotkb_archaea_asgard_group_1935183_0*.

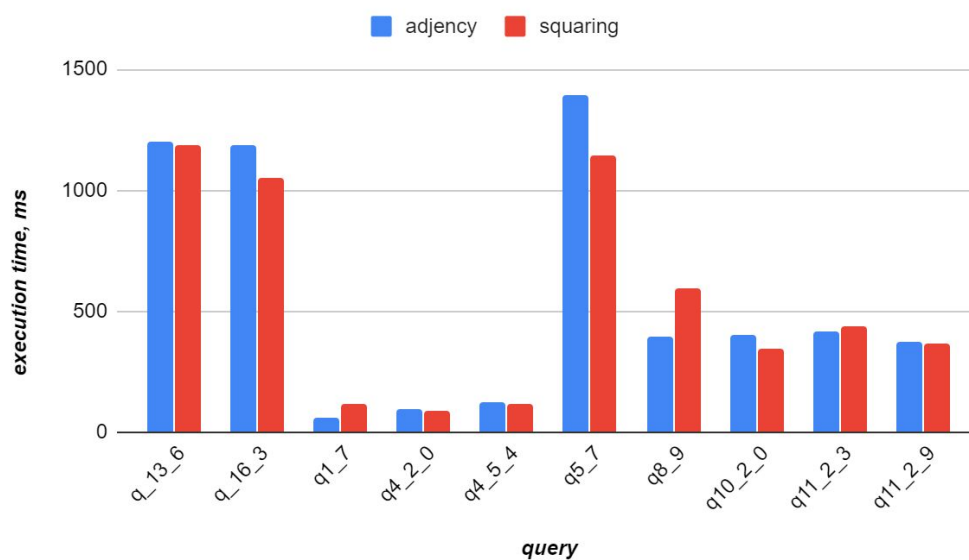
Автоматы, полученные из вышеуказанных датасетов, пересекались с 10 случайно выбранными регулярными выражения из директории *LUBM300/regexes*.

Время указано среднее из 5 замеров на запрос, данные для графиков можно посмотреть в `src/tests/benchmark/{graph_name}_adjency.txt` или `src/tests/benchmark/{graph_name}_squaring.txt` для умножения на матрицу смежности и возведения в квадрат соответственно. Данные представлены в следующем виде:

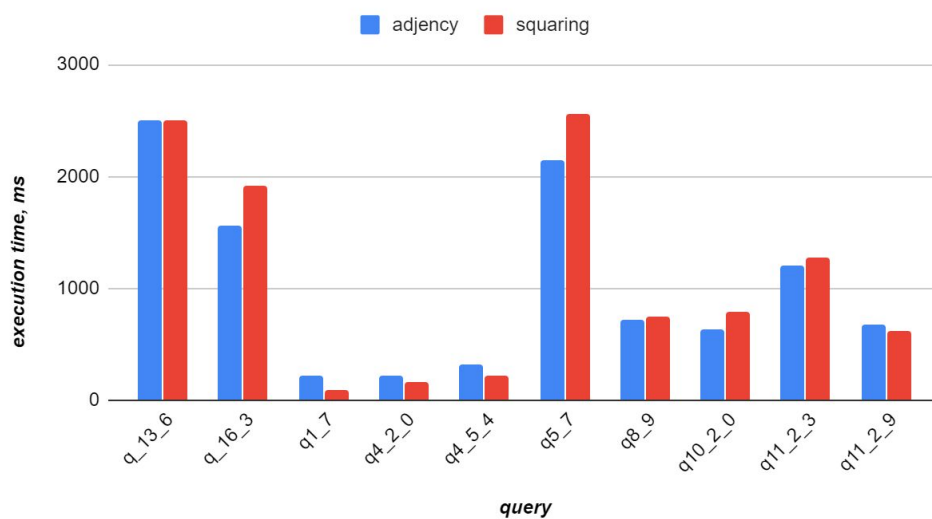
“\$regex: \$intersection_time \$closure_time - \$check_sum”, где **regex** - название файла с

регулярным выражением, **intersection_time** - время, потраченное на пересечение графов, **closure_time** - время, потраченное на получение матрицы достижимости, **check_sum** - кол-во непустых ячеек в матрице достижимости. Для далее используемых данных контрольные значения обоих методов совпали.

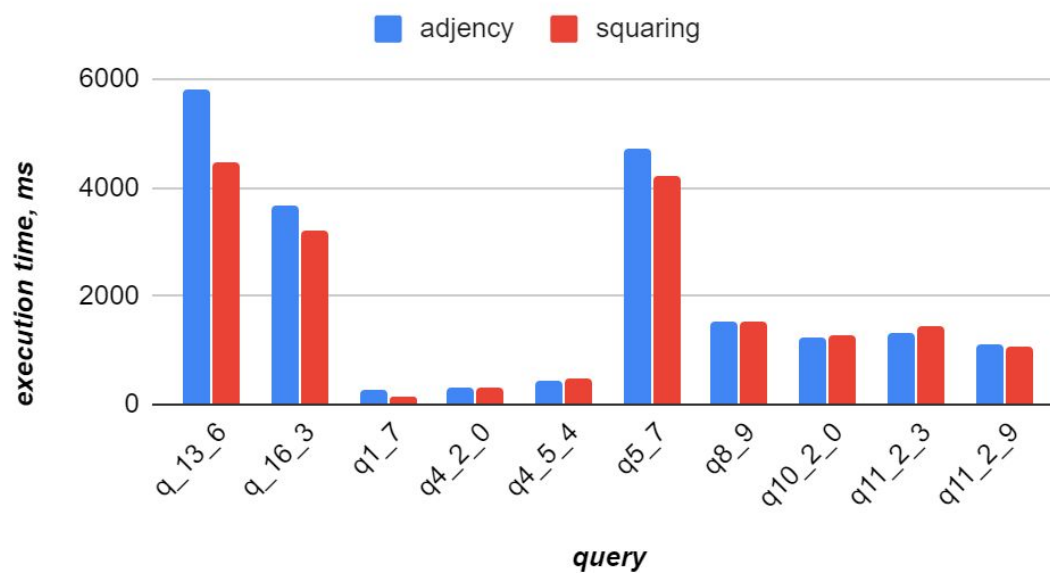
LUBM300



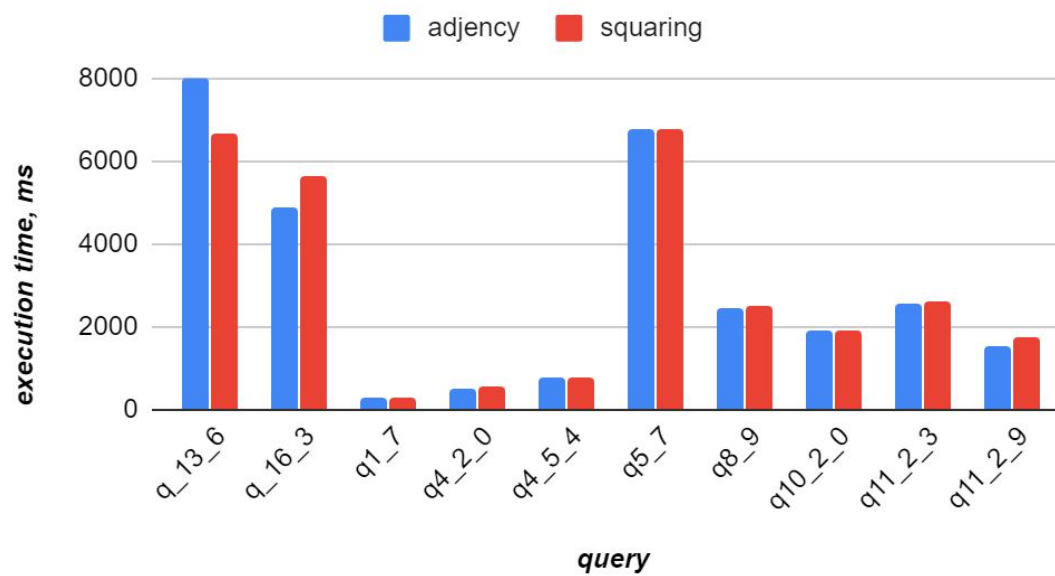
LUBM500



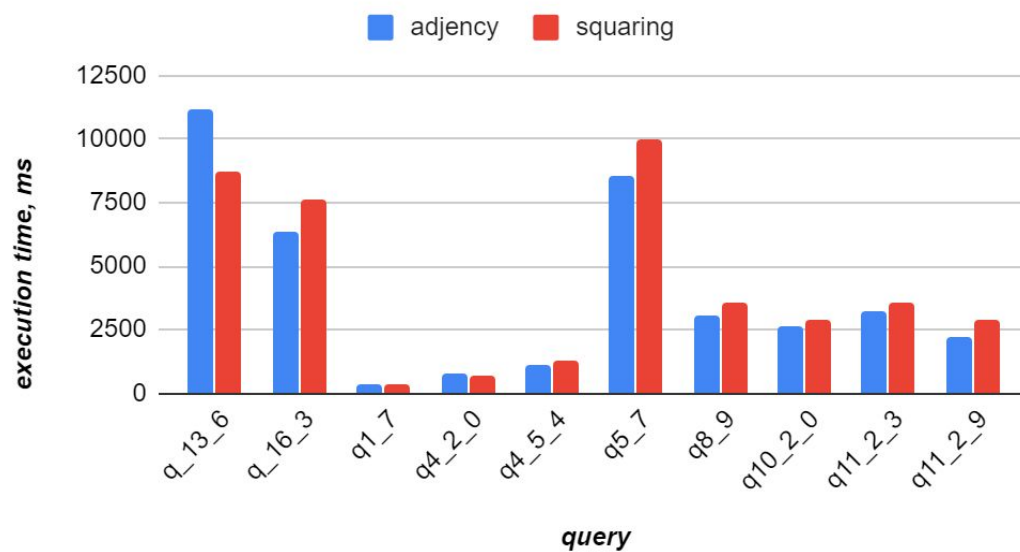
LUBM1M



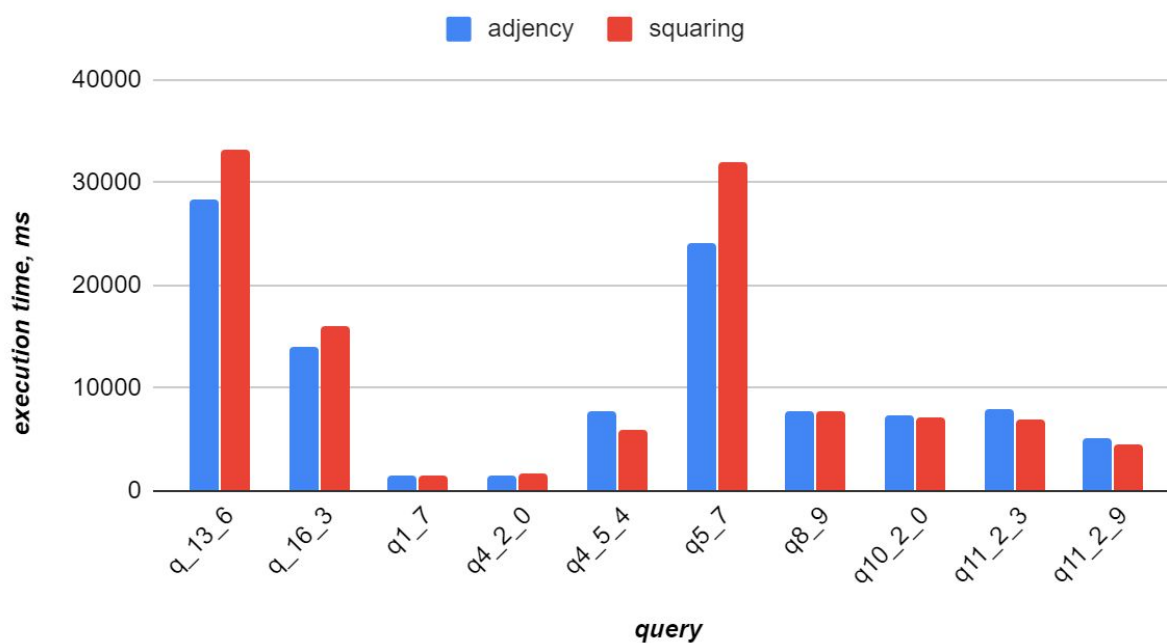
LUBM1.5M



LUBM1.9M



proteomes uniprotkb_archaea_asgard_group_1935183_0



Выводы

Первый тривиальное умозаключение в том, что увеличение вершин в графе прямо пропорционально времени исполнения запроса на нем. Что касается сравнения производительности методов умножения на матрицу смежности и возведения в квадрат, то видно, что на большей части тестов они очень близки по времени, при этом есть как тесты, в которых выигрывает одна реализация, так и тесты, в которых оказывается быстрее другая. Я думаю, это говорит о том, что малое количество операций умножения на плотную матрицу и большее количество умножений на разреженную по сложности похожи для внутренних вычислений `pygraphblas`, поэтому от конкретной ситуации зависит, что окажется быстрее, а предпочтение можно отдать любому способу.