Отчет об экспериментальном сравнении реализаций

Автор:

Шапошников Алексей, СПБГУ

23 сентября 2020 г

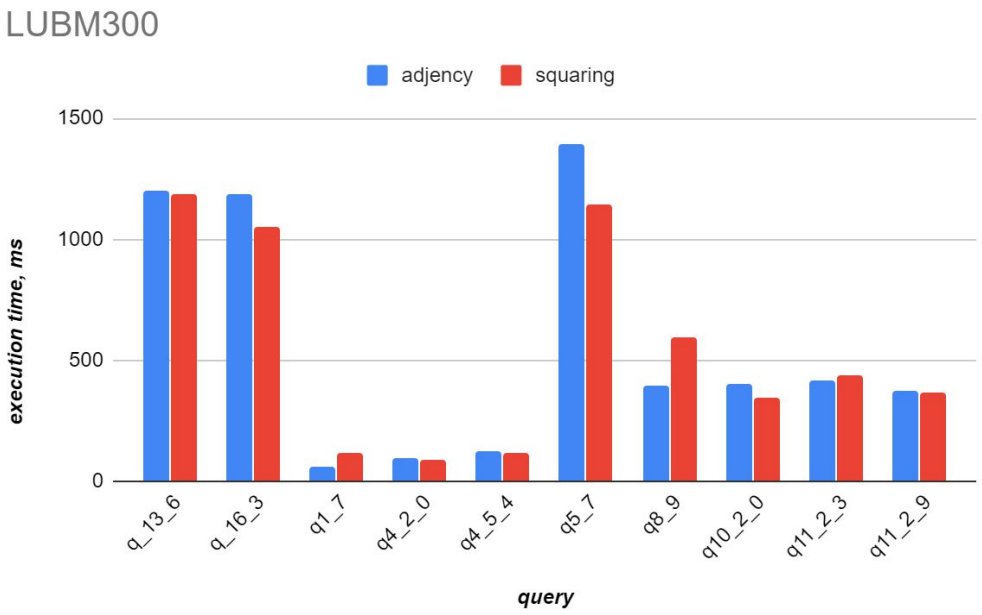
Экспериментальный анализ производительности

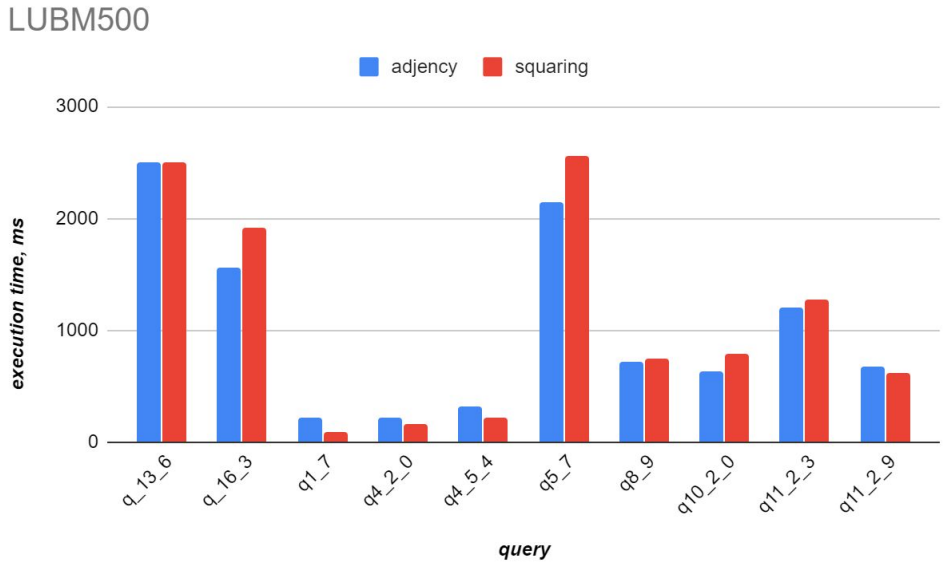
При проведении эксперимента использовался компьютер с Windows 10 -> WSL-Ubuntu 20.04; Intel Core i5-4690 CPU @ 3.50 GHz; 16 gb DDR4 RAM.

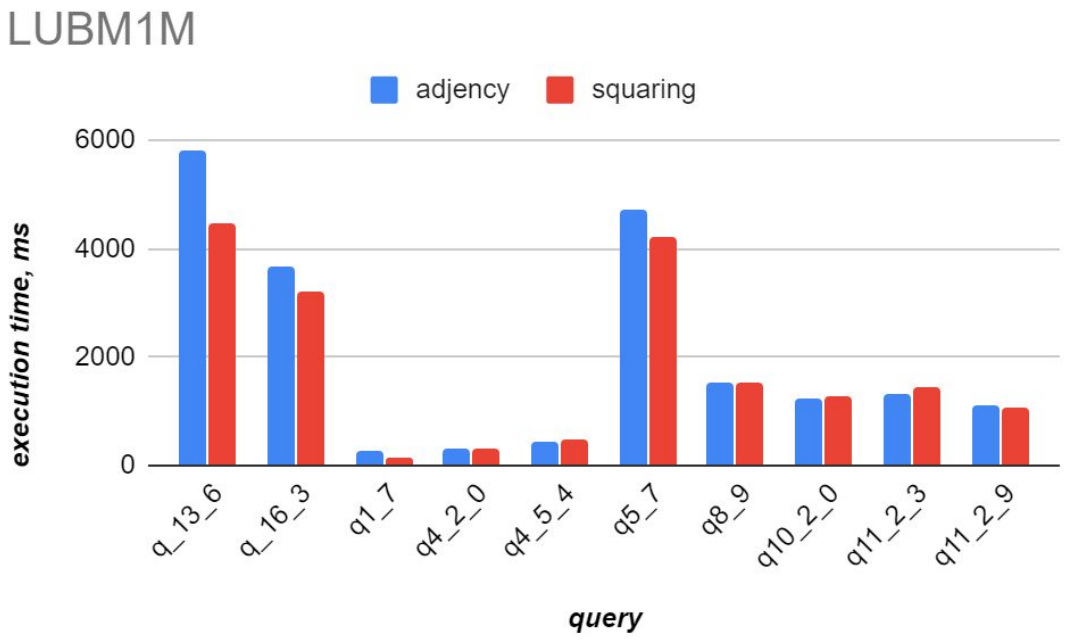
Для эксперимента был произведен анализ производительности вычисления регулярных запросов с помощью двух реализаций транзитивного замыкания: возведения в квадрат и умножения на матрицу смежности. В качестве графов были использованы следующие датасеты из пакета **refinedDataForRPQ**: *LUBM300*, *LUBM500*, *LUBM1M*, *LUBM1.5M*, *LUBM1.9M*, *proteomes*, *uniprotkb_archea_asgard_group_1935183_0*. Автоматы, полученные из вышеуказанных датасетов, пересекались с 10 случайно выбранными регулярными выражения из директории LUBM300/regexes.

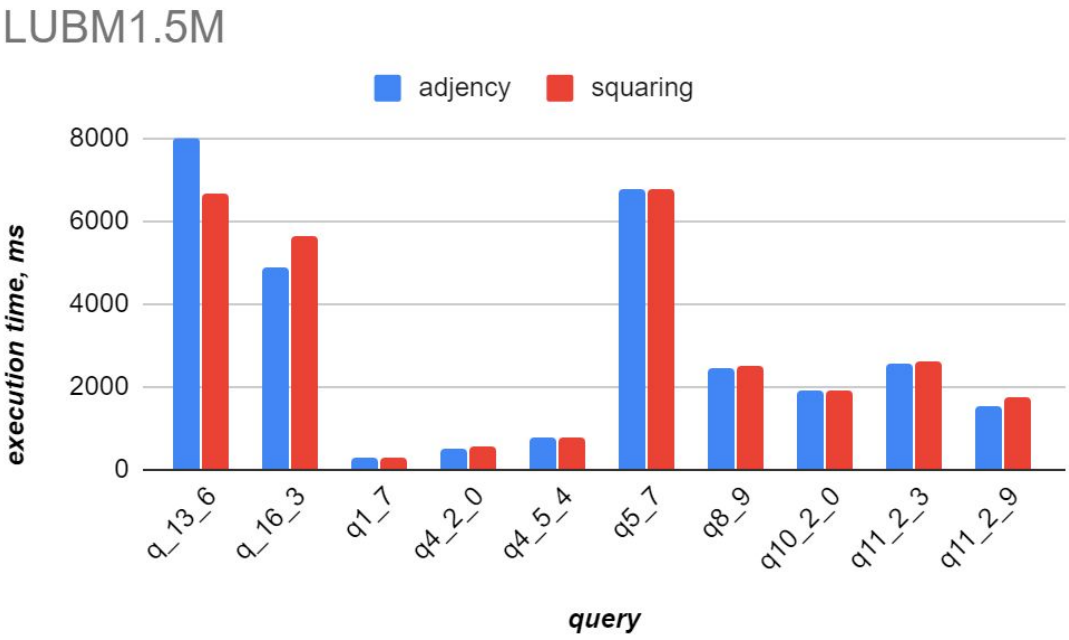
Время указано среднее из 5 замеров на запрос, данные для графиков можно посмотреть в src/tests/benchmark/{graph_name}_adjency.txt или src/tests/benchmark/{graph_name}_squaring.txt для умножения на матрицу смежности и возведения в квадрат соотвественно. Данные представлены в следующем виде: “$regex: $intersection_time $closure_time - $check_sum”, где **regex** - название файла с

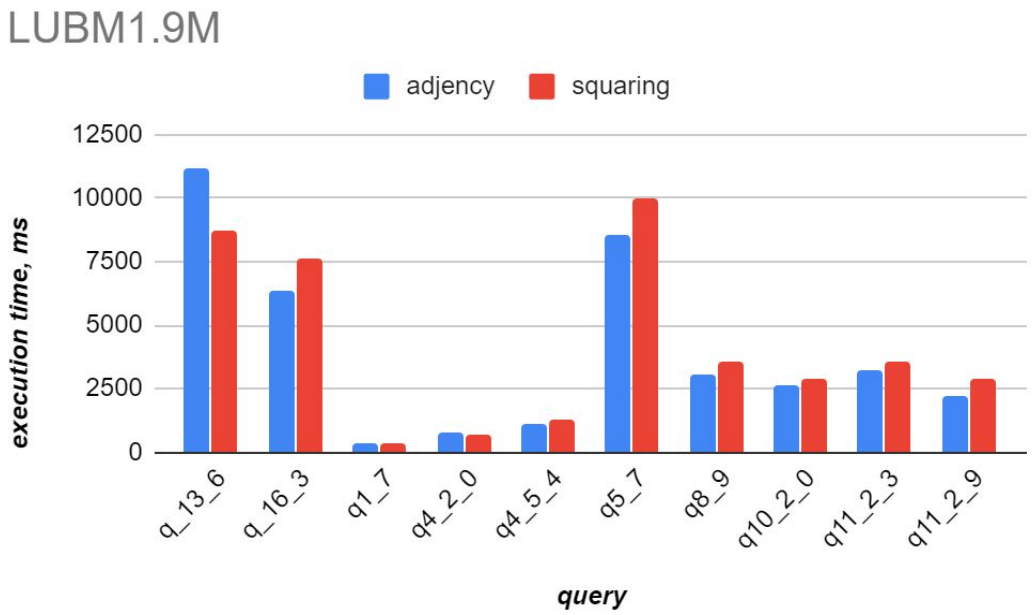
регулярным выражением, **intersection_time** - время, потраченное на пересечение графов, **closure_time** - время, потраченное на получение матрицы достижимости, **check_sum** - кол-во непустых ячеек в матрице достижимости. Для далее используемых данных контрольные значения обоих методов совпали.

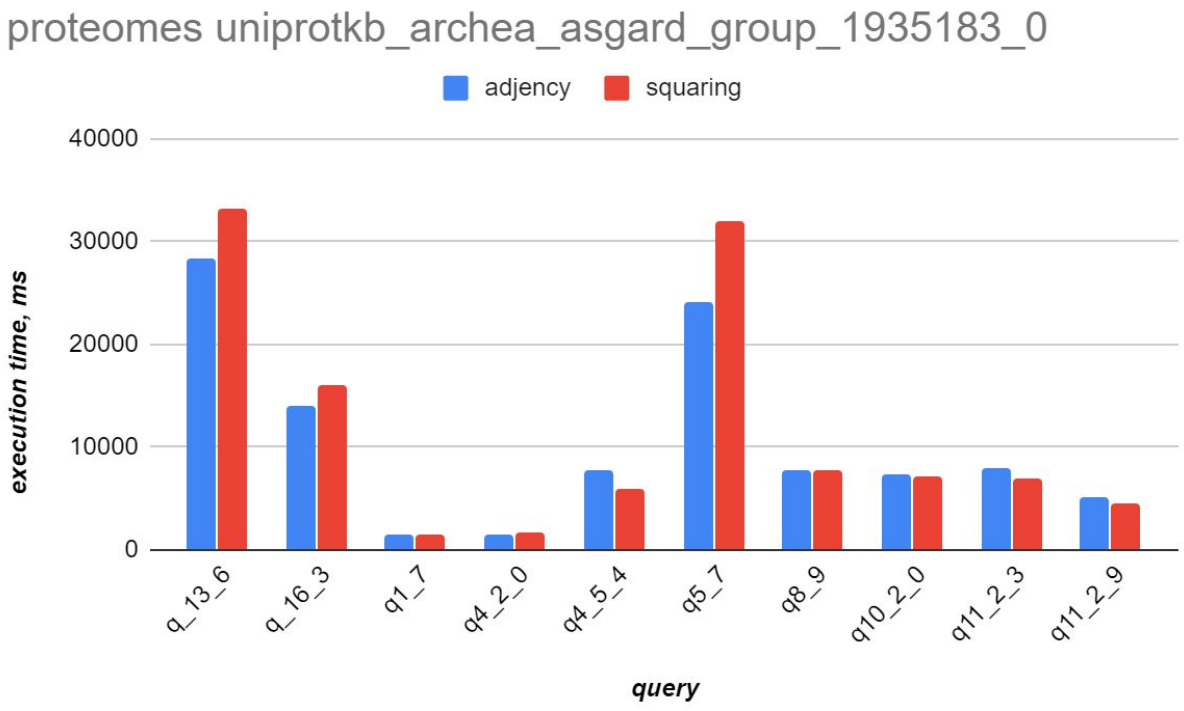





LUBM1M
adjency
squaring
6000
4000
2000
0
execution time, ms
q_13_6
q_16_3
q1_7
q4_2_0
q4_5_4
q5_7
q8_9
q10_2_0
q11_2_3
q11_2_9
query


LUBM1.5M
adjency
squaring
8000
6000
4000
2000
0
execution time, ms
q_13_6
q_16_3
q1_7
q4_2_0
q4_5_4
q5_7
q8_9
q10_2_0
q11_2_3
q11_2_9
query


LUBM1.9M
adjency
squaring
12500
10000
7500
5000
2500
0
execution time, ms
q_13_6
q_16_3
q1_7
q4_2_0
q4_5_4
q5_7
q8_9
q10_2_0
q11_2_3
q11_2_9
query


proteomes uniprotkb_archea_asgard_group_1935183_0
adjency
squaring
40000
30000
20000
10000
0
execution time, ms
q_13_6
q_16_3
q1_7
q4_2_0
q4_5_4
q5_7
q8_9
q10_2_0
q11_2_3
q11_2_9
query

Выводы

Первое тривиальное умозаключение в том, что увеличение вершин в графе прямо пропорционально времени исполнения запроса на нем. Что касается сравнения производительности транзитивного замыкания путем 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.