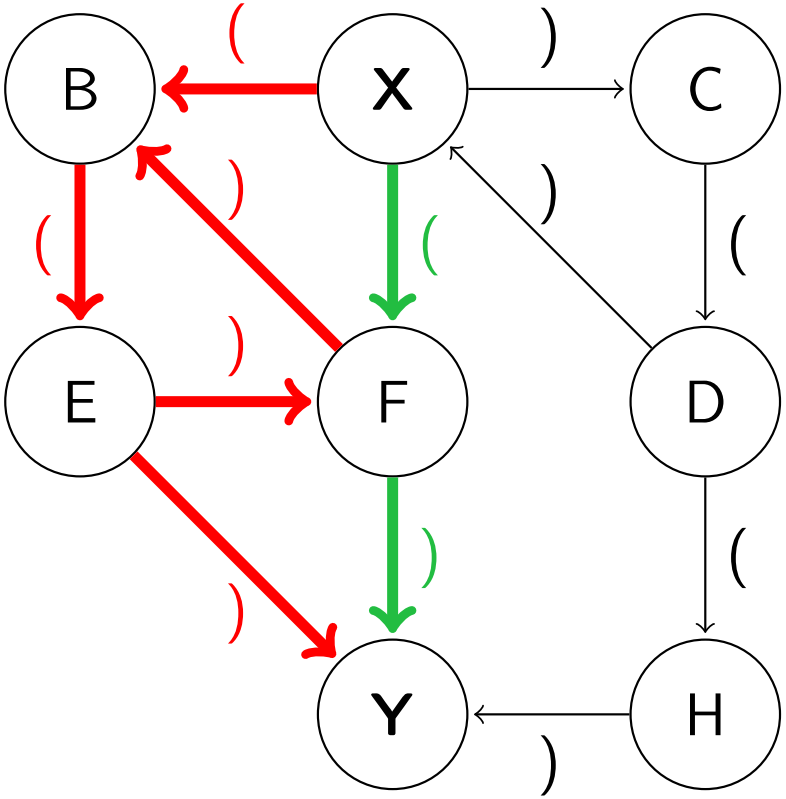
Поиск путей в графовых базах данных через тензорное произведение

Эпельбаум Илья Владимирович, 17.Б10-мм группа
Научный руководитель: к.ф.-м.н., доцент С.В. Григорьев

Кафедра системного программирования
Санкт-Петербургский государственный университет

14 мая 2020

Введение



Найти путь из **X** в **Y**, который является сбалансированной скобочной последовательностью

Ограничения — контекстно-свободный язык, заданный грамматикой:

$$S \to \varepsilon \mid (S)S$$

(()) ()

()

Введение

- Поиск путей выражается в терминах формальных языков
- Существующие алгоритмы не могут быть использованы в промышленных продуктах
  - Статья "An Experimental Study of Context-Free Path Query Evaluation Methods" Jochem Kuijpers et al, 2019 год
- Один из путей — линейная алгебра
  - Статья "Evaluation of the CFPQ Algorithm Based on Matrix Multiplication" Nikita Mishin, Iaroslav Sokolov... 2019 год

Обзор

- Алгоритм, предложенный Рустамом Азимовым
  - ✔ Операции на матрицах
  - ✖ НФХ
- Алгоритм, основанный на тензорном произведении
  - ✔ Рекурсивный автомат
  - ✔ Операции на матрицах
  - ✖ Матрицы больших размеров ($mn \times pq$)

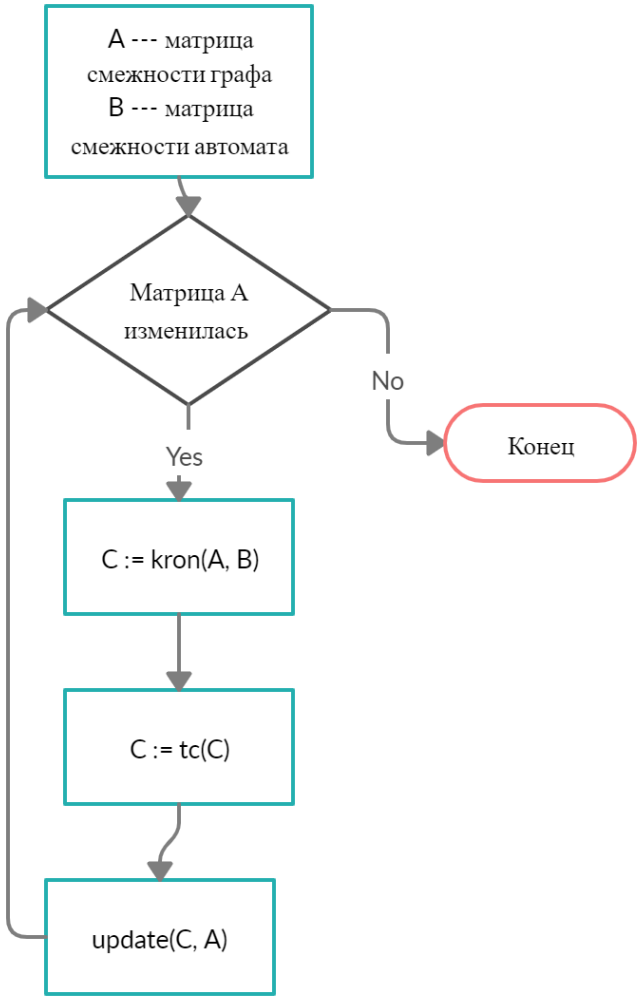
Постановка задачи

Цель — сравнить алгоритм, основанный на тензорном произведении, и алгоритм, основанный на матричном умножении

Задачи:

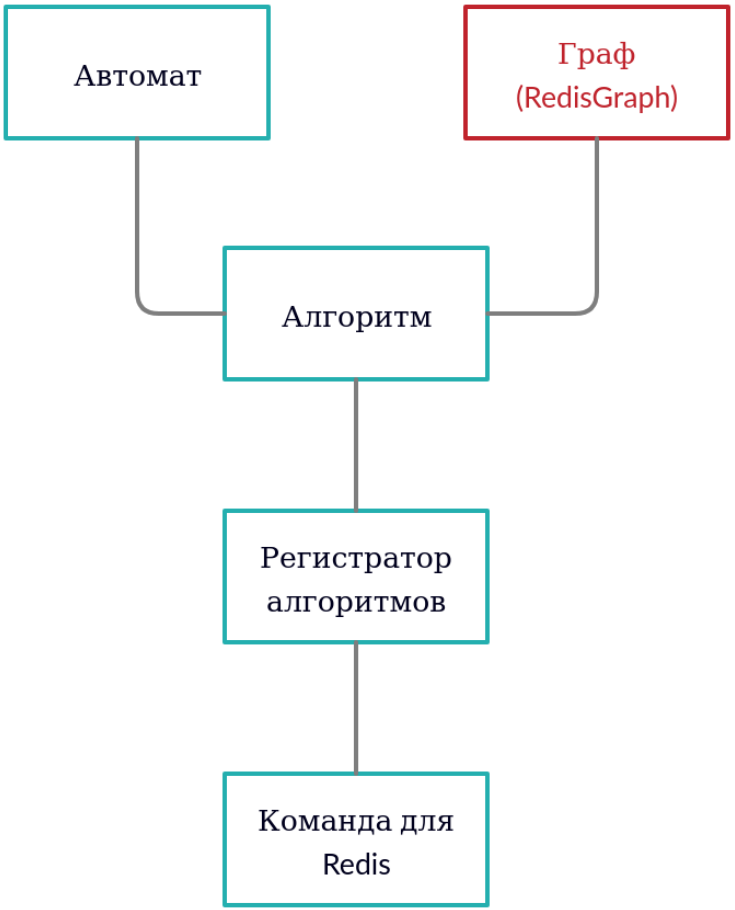
- Реализовать алгоритм, основанный на тензорном произведении
- Внедрить реализацию в СУБД RedisGraph
- Произвести экспериментальное исследование производительности реализации

Подход к реализации



- Матрица смежности графа
$$\begin{bmatrix} b & a \\ a & b \end{bmatrix} \Leftrightarrow a : \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, b : \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
- Тензорное произведение вычисляется в булевом полукольце:
$$C = A \otimes B \Leftrightarrow \widetilde{C} = \sum_i (A_i \otimes B_i)$$
- SuitSparse — реализация GraphBLAS API

Реализация



Алгоритм:

- RedisGraph — хранилище графов
- Использование булева полукольца

Команда для Redis:

- GRAPH.CFG
  название_алгоритма
  название_графа
  путь_до_автомата

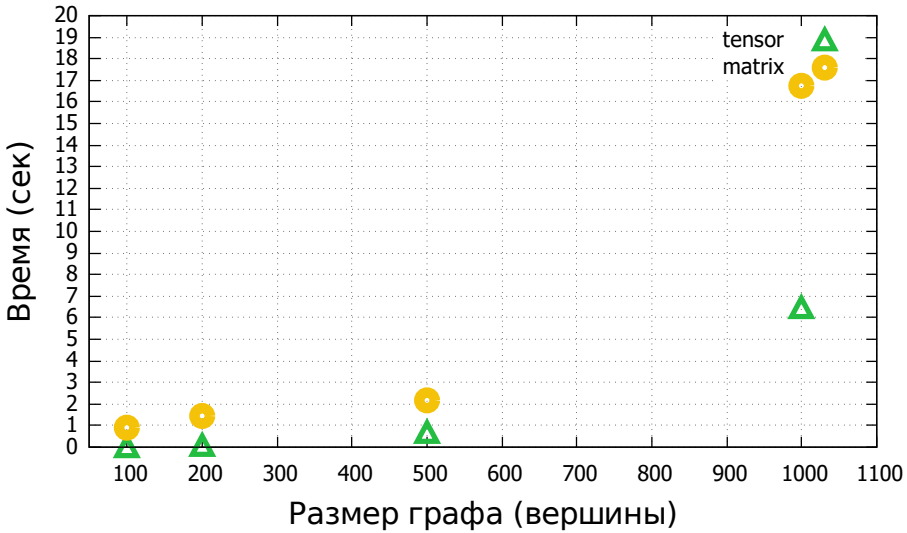
Эксперименты

- PC: Ubuntu 18.04. Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz CPU, DDR4 32 Gb RAM

Данные:

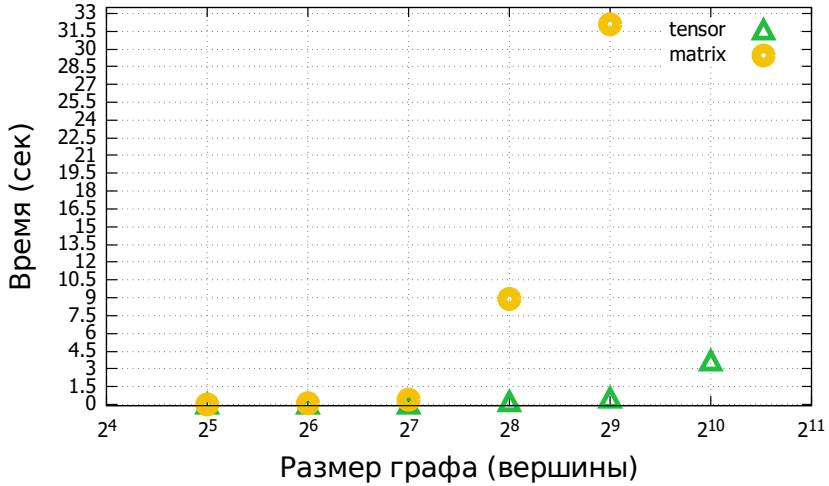
- WorstCase — теоретически наихудшая ситуация
- FullGraph — цикл, наихудшая ситуация для разреженных матриц
- RDF — реальные данные

Эксперименты



FullGraph

- Более чем в 2 раза быстрее
- $\pm\Delta = \pm 0,06$



Worstcase

- Более чем в 50 раз быстрее
- $\pm\Delta = \pm 0,06$

Эксперименты

Граф	V	E	Tensor	Matrix
pathways	6238	37196	0.008	0.009
go	272770	1068622	1.744	0.604
atom-primitive	291	685	0.011	0.016
eclass_514en	239111	1047454	0.329	0.067
foaf	256	815	0.001	0.002
funding	778	1480	0.005	0.006
go-hierarchy	45007	1960436	0.179	0.091

RDF

- $\pm\Delta = \pm 0,06$

Выводы

- Положительные результаты для RDF только на маленьких графах
- Результаты на Worstcase и FullGraph превосходят показатели матричного алгоритма

$\Rightarrow$ Реализацию необходимо улучшать

Результаты

- Реализован[1] алгоритм, основанный на тензорном произведении
- Реализация интегрирована в СУБД RedisGraph
- Проведены экспериментальные исследования на наборах данных:
  - WorstCase
  - FullGraph
  - RDF
- Результат принят на конференцию:
  - Статья "Context-Free Path Querying by Kronecker Product" на конференции ADBIS 2020

[1] Реализация: https://github.com/IlyaEp/RedisGraph/tree/IlyaEp_tensor