Санкт-Петербургский государственный университет

Кафедра системного программирования

Разработка алгоритма для задачи достижимости с регулярными ограничениями

Порсев Денис Витальевич, 19.Б10-мм

Научный руководитель: к.ф.-м.н., Григорьев С.В., доцент кафедры информатики

Санкт-Петербург
2022

Графовые запросы

- Графовое представление данных
  - Метки на ребрах
  - Формальные ограничения
- Регулярные запросы к графовым БД
  - Достижимость и обход графа
  - Поиск путей в графе
  - Путь ограничен регулярной грамматикой

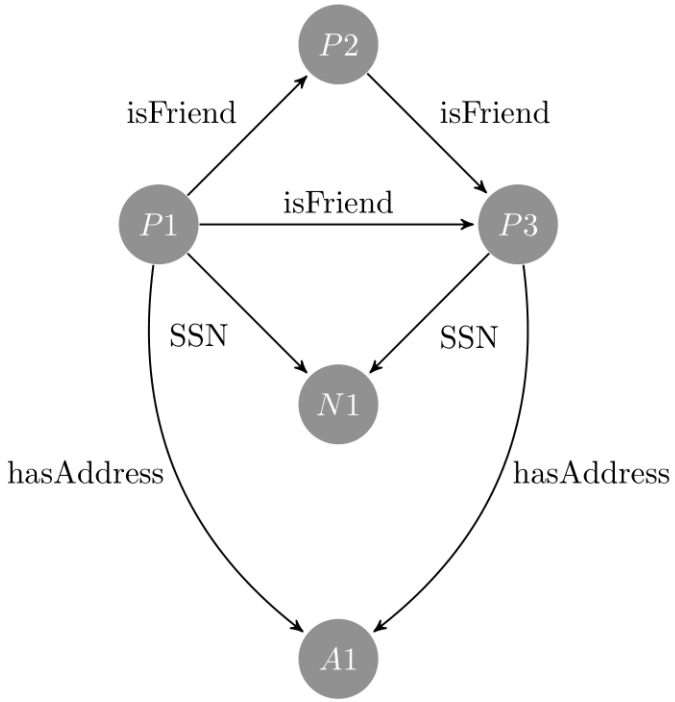


Рис.: Представление в виде графа

Мотивация создания алгоритма

- Регулярные запросы активно используются на практике
  - В языках запросов: SPARQL v1.1, Cypher и др.
  - Существующие подходы неэффективны на различных графах
- Новый подход: линейная алгебра
  - Использование разреженной матрицы смежности
  - Основные подходы к регулярным запросам используют BFS
  - BFS выражен в терминах линейной алгебры
  - SuiteSparse:GraphBLAS — реализация примитивов линейной алгебры
- Подготовка к запросам с контекстно-свободными ограничениями

Постановка задачи

Цель: разработать алгоритм для задачи достижимости с регулярными ограничениями (RPQ) для нескольких стартовых вершин

Задачи:

- Провести обзор алгоритмов RPQ и методов решения задачи
- Разработать матричный алгоритм, решающий задачу регулярной достижимости
- Реализовать разработанный алгоритм
- Провести экспериментальное исследование алгоритма

Задача достижимости с регулярными ограничениями

Определение (Задача регулярной достижимости в графе для нескольких стартовых вершин)

Имеется:

- *Регулярный язык* $\mathcal{R} = \langle Q, \Sigma, P, F, Q_{src} \rangle$
- *Направленный помеченный граф* $\mathcal{G} = \langle V, E, L \rangle$
- *Множество начальных вершин* $V_{src} \subset V$

Постановка задачи 1:

- *Найти множество* $\{w \mid \exists$ *путь* $\pi = (e_1, \ldots, e_n), e_k = (v_k, l_k, v_{k+1}),$ *т.ч.* $L(\pi) = (l_1 l_2 \ldots l_n) \in L(\mathcal{R}),\ v_1 \in V_{src}, w \in V\}$

Постановка задачи 2:

- *Найти множество пар* $\{(v, w) \mid \exists$ *путь* π, *т.ч.* $L(\pi) = (l_1 l_2 \ldots l_n) \in L(\mathcal{R}),\ v_1 \in V_{src}, w \in V\}$

Основные алгоритмы RPQ

Два основных подхода:

- Алгоритм α-RA, реляционная алгебра
- Использование FA, конечные автоматы

Основные алгоритмы RPQ: Использование *FA*

Идея:

- Регулярный запрос представляется в виде автомата $\mathcal{R}$
- Граф представляется в виде автомата $\mathcal{G}$
- Пересечение $\mathcal{R}$ и $\mathcal{G}$ содержит информацию о достижимых вершинах

Оптимизация:

- Матричный BFS синхронно по $\mathcal{R}$ и $\mathcal{G}$

$$Q_1 = (sameAs \cdot isLocated)^+$$

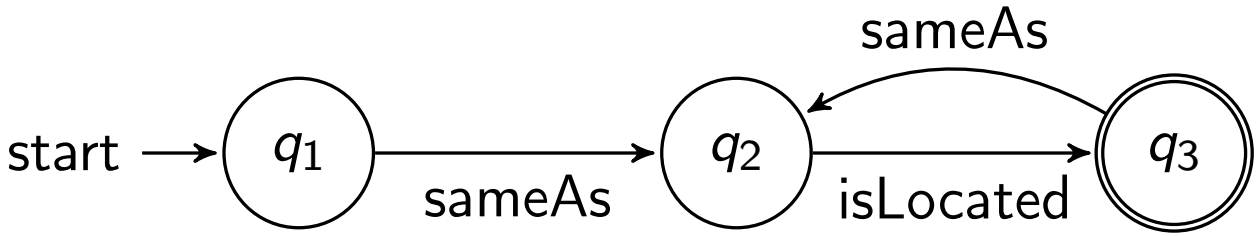


Рис.: Представление регулярного запроса в виде автомата

Разработанный алгоритм

Входные данные: $\mathcal{R}, \mathcal{G}, V_{src}$
Выходные данные: $\mathcal{P}$ - матрица достижимых пар вершин

- Посимвольно декомпозируем в булевые матрицы $Bool_{\mathcal{R}_a}$ и $Bool_{\mathcal{G}_a}$
- Синхронизируем обход с помощью следующей матрицы

$$\mathfrak{D} = \begin{bmatrix} Bool_{\mathcal{R}_a} & 0 \\ 0 & Bool_{\mathcal{G}_a} \end{bmatrix} \tag{1}$$

- Строим матрицу с начальными вершинами, которая будет хранить найденные во время обхода вершины

$$M^{k \times (k+n)} = \begin{bmatrix} Id_k & Matrix_{k \times n} \end{bmatrix} \tag{2}$$

- Пока M меняется:
  - Перемножаем $M \times \mathfrak{D}$ для получения новых вершин
  - Переставляем строчки в M для преобразования к начальному виду
  - Записываем найденные вершины в $\mathcal{P}$

Условия эксперимента

Оборудование:

- Intel Core i7-10750H 6 $\times$ 2.60 GHz
- 16 Gb DDR4 RAM

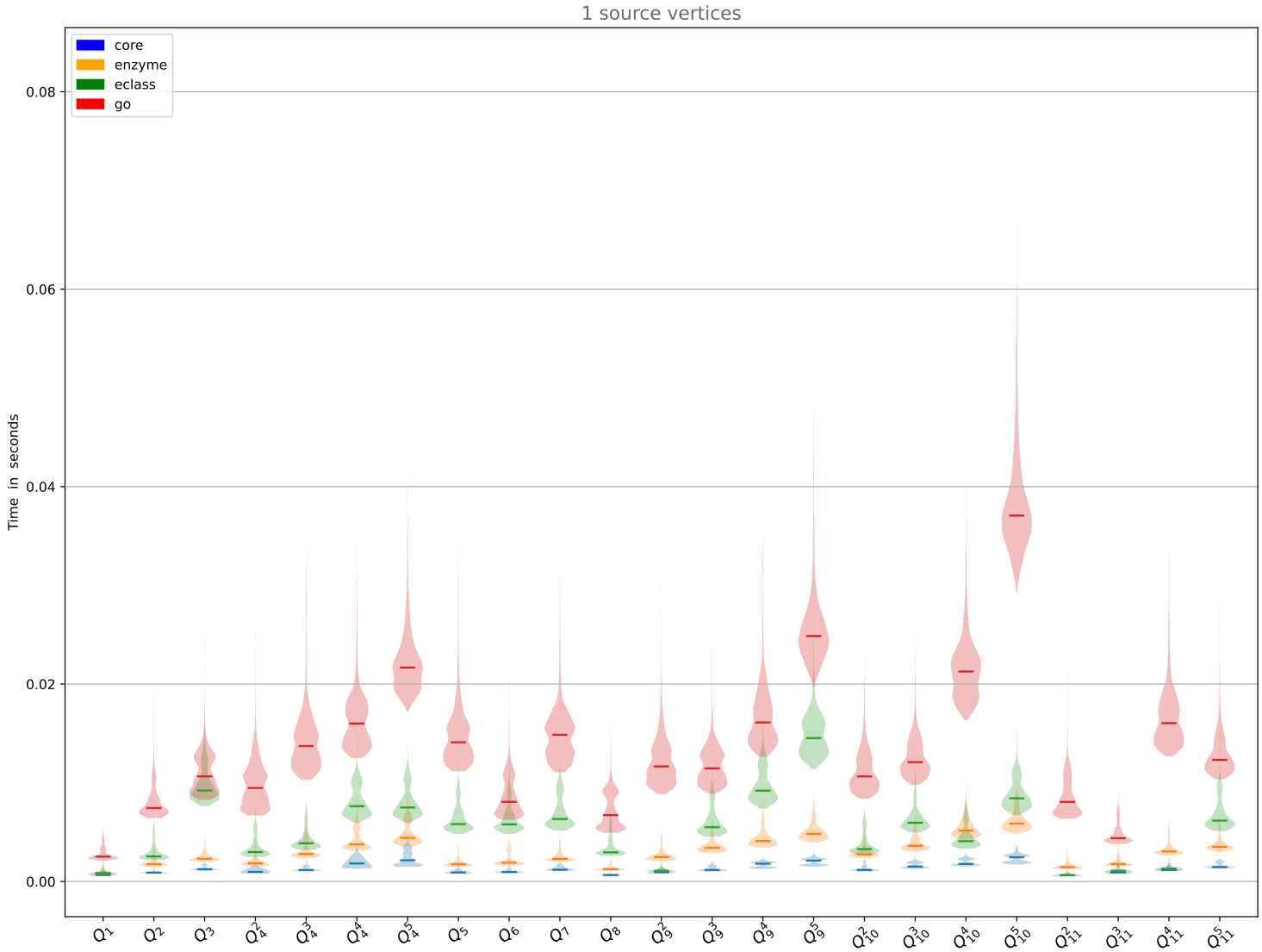
Graph	# Vertices	# Edges
core	1 323	2 752
enzyme	48 815	86 543
eclass	239 111	360 248
go	582 929	1 437 437

Таблица: *Графы*

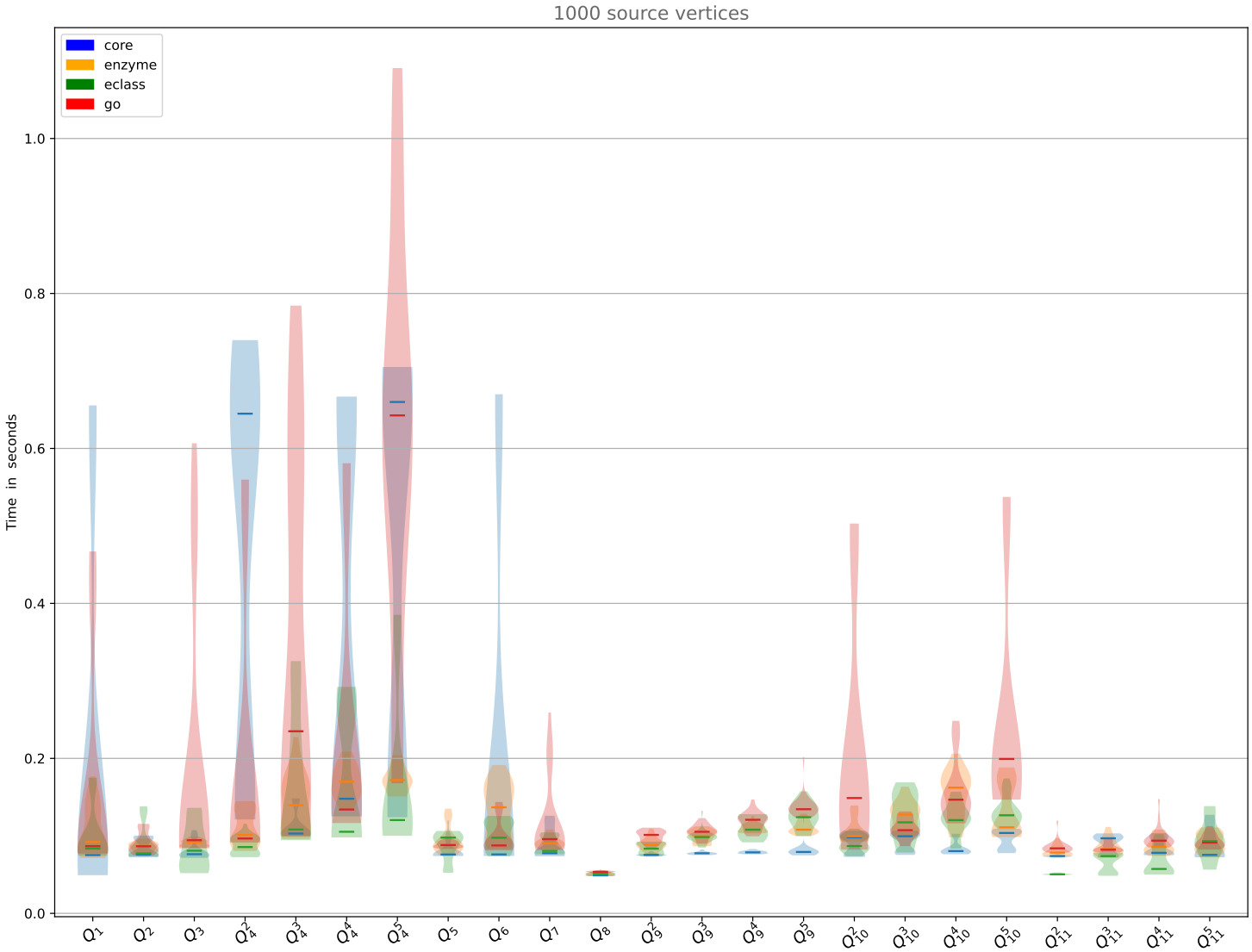
Name	Query
Q_1	a^*
Q_2	$a \cdot b^*$
Q_3	$a \cdot b^* \cdot c^*$
Q_4	$(a \mid b)^*$
Q_5	$a \cdot b^* \cdot c$
Q_6	$a^* \cdot b^* \cdot c$
Q_7	$a \cdot b \cdot c^*$
Q_8	$a? \cdot b^*$
Q_9	$(a \mid b)^+$
Q_{10}	$(a \mid b)^+ \cdot c^*$
Q_{11}	$a \cdot b$

Таблица: *Шаблоны запросов*

Экспериментальное исследование (SS)



Экспериментальное исследование (MS 1000)



Результаты

- Проведен обзор основных методов и алгоритмов решения задачи регулярной достижимости
- Разработан алгоритм, решающий задачу достижимости с регулярными ограничениями, в терминах линейной алгебры
- Реализован[1] разработанный алгоритм
- Проведено экспериментальное исследование алгоритма

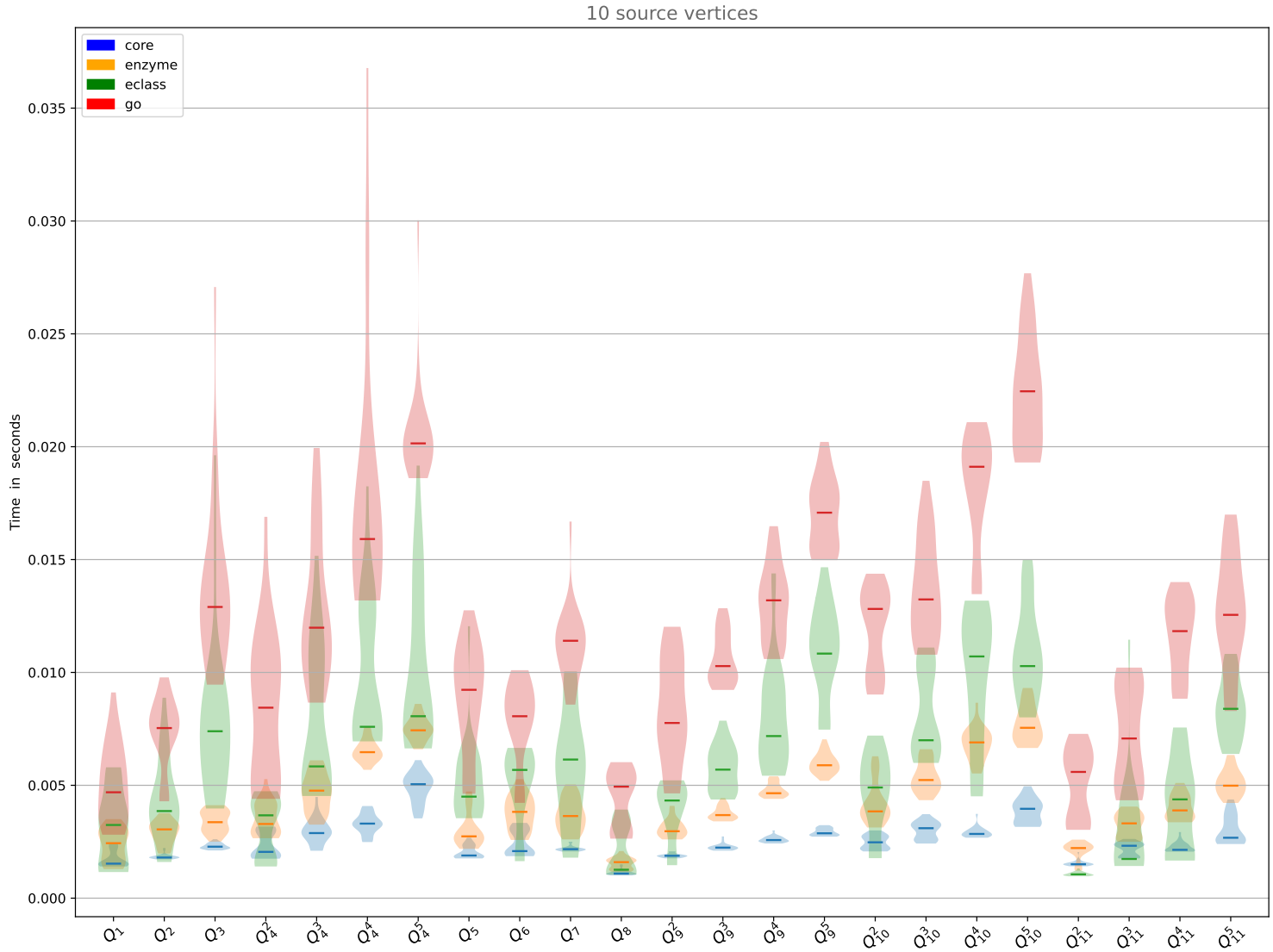
[1] https://github.com/JetBrains-Research/CFPQ_PyAlgo/pull/34

Алгоритм в псевдокоде

Algorithm 1 Алгоритм достижимости в графе с регулярными ограничениями на основе поиска в ширину, выраженный с помощью операций матричного умножения

1: **procedure** BFSBasedRPQ($\mathcal{R} = \langle Q, \Sigma, P, F, q\rangle, \mathcal{G} = \langle V, E, L\rangle, V_{src}$)
2: $\mathcal{P} \leftarrow$ Матрица смежности графа
3: $\mathfrak{D} \leftarrow Bool_{\mathcal{R}} \bigoplus Bool_{\mathcal{G}}$ ▷ Построение матриц $\mathfrak{D}$
4: $M \leftarrow CreateMasks(|Q|, |V|)$ ▷ Построение матрицы M
5: $M' \leftarrow SetStartVerts(M, V_{src})$ ▷ Заполнение нач. вершин
6: **while** Матрица M меняется **do**
7: $M \leftarrow M'\langle\neg M\rangle$ ▷ Применение комплементарной маски
8: **for all** $a \in (\Sigma \cap L)$ **do**
9: $M' \leftarrow M$ any.pair $\mathfrak{D}$ ▷ Матр. умножение в полукольце
10: $M' \leftarrow TransformRows(M')$ ▷ Приведение M' к виду M
11: $Matrix \leftarrow extractRightSubMatrix(M')$
12: $V \leftarrow Matrix.reduceVector()$ ▷ Сложение по столбцам
13: **for** $k \in 0 \ldots |V_{src}| - 1$ **do**
14: $W \leftarrow \mathcal{P}.getRow(k)$
15: $\mathcal{P}.setRow(k, V + W)$
16: **return** $\mathcal{P}$

Экспериментальное исследование (MS 10)



Экспериментальное исследование (MS 100)