Московский государственный университет
имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики
Кафедра математической кибернетики

Курсовая работа

«Тема работы»

студента группы 518

Иванова Андрея Александровича

Научный руководитель:

д.ф.-м.н., проф.

Алексеев В. Б.

Москва, 2025

Содержание

1 Введение **3**

2 Постановка задачи **4**

3 Основная часть **4**

4 Полученные результаты **6**

Список литературы **6**

1 Введение

Вычисление произведения матриц является крайне распространённой задачей. Эта операция широко применяется в таких областях как компьютерная графика, моделирование физических процессов, обучение нейронных сетей и т.д.

Пусть необходимо вычислить произведение двух матриц размеров $n \times n$. Получаемый непосредственно из определения алгоритм «строка на столбец» требует $O(n^3)$ операций. В 1969 году Ф. Штрассен предложил алгоритм [1], который требует всего $O(n^{\log_2 7})$ операций для того же вычисления ($\log_2 7 \approx 2.807 < 3$). В этой же работе было показано, что на основе быстрого алгоритма умножения матриц могут быть построены алгоритмы вычисления определителя и обратной матрицы с той же асимптотикой. Ключевым шагом алгоритма Штрассена является возможность умножить две матрицы 2×2 за 7 умножений вместо стандартных 8.

Пусть даны матрицы $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ и $B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$. Требуется вычислить $C = AB$. Согласно алгоритму Штрассена вначале вычисляются линейные комбинации элементов матриц $\alpha_k = \sum u_{ij}^{(k)} a_{ij}$, $\beta_k = \sum v_{ij}^{(k)} b_{ij}$. Затем вычисляются произведения этих линейных комбинаций $\gamma_k = \alpha_k \beta_k$. Ответ вычисляется как $C = \sum \gamma_k W_k^T$. Где W_k — 7 матриц 2×2 и $1 \leqslant i, j \leqslant 2$; $1 \leqslant k \leqslant 7$. Для того чтобы данная процедура корректно вычисляла произведение матриц необходимо и достаточно чтобы элементы матриц U_k, V_k, W_k удовлетворяли трилинейной системе уравнений $\sum u_{i_1 j_1}^{(k)} v_{i_2 j_2}^{(k)} w_{i_3 j_3}^{(k)} = \delta_{i_1 j_3} \delta_{i_2 j_1} \delta_{i_3 j_2}$, где суммы берутся по k, δ_{ij} — символы Кронекера и уравнения должны быть выполнены для любого набора $(i_1, j_1, i_2, j_2, i_3, j_3)$.

Алгоритмы такого вида получили название билинейных. Далее задача умножения матрицы размера $m \times n$ на матрицу размера $n \times p$ будет обозначаться как $\langle m, n, p \rangle$. Можно показать, что минимальное количество умножений в билинейном алгоритме не меняется при перестановке m, n, p, и что имея алгоритм билинейной сложности L для решения задачи $\langle m, n, p \rangle$ можно построить алгоритм умножения матриц произвольного размера N со сложностью $O(N^\omega)$, где $\omega = 3 \log_{(mnp)} L$.

В 1971 году Ш. Виноград доказал, что для матриц данного размера алгоритм Штрассена оптимален, то есть невозможно умножить две матрицы 2×2, использовав менее 7 умножений [2].

Для задач $\langle m, n, 1\rangle$ стандартный алгоритм сложности mn оптимален. На настоящий момент известны также точные оценки сложности задач $L(\langle 2, 2, 3\rangle) = 11$ [3], $L(\langle 2, 2, 4\rangle) = 14$ [4], $L(\langle 2, 3, 3\rangle) = 15$ [**?**]. Для бóльших размеров матриц известные нижние и верхние оценки уже не совпадают: $17 \leqslant L(\langle 2, 2, 5\rangle) \leqslant 18$ [**?**], $19 \leqslant L(\langle 3, 3, 3\rangle) \leqslant 23$ [5, **?**], $34 \leqslant L(\langle 4, 4, 4\rangle) \leqslant 49$ [**?**] [1].

Заметим, что и система на коэффициенты билинейного алгоритма, и её решение соответствующее алгоритму Штрассена симметричны относительно одновременного циклического сдвига U_k, V_k, W_k, а также что в алгоритме Штрассена один комплект (U_k, V_k, W_k) состоит из единичных матриц. Поэтому поиск решения часто ведётся именно в этом классе: один комплект коэффициентов представлен единичными матрицами, а остальные разбиваются на тройки, переводимые друг в друга циклическим сдвигом.

2 Постановка задачи

В рамках курсовой работы требовалось написать программу, которая путём полного перебора находит или подтверждает отсутствие решения сложности 19 задачи $\langle 3, 3, 3\rangle$ над $\mathbb{F}_2$ в указанном классе.

3 Основная часть

Общая система отвечающая билинейному алгоритму сложности 19 для задачи $\langle 3, 3, 3\rangle$, содержит $19 \cdot 3 \cdot 9 = 513$ неизвестных и $3^6 = 729$ уравнений. Зафиксировав один комплект коэффициентов, мы сокращаем количество неизвестных до $18 \cdot 3 \cdot 9 = 484$. Постулировав циклическую симметрию для коэффициентов мы сокращаем количество неизвестных до $6 \cdot 3 \cdot 9 = 162$ и уравнений до 249.

Перебор разбивается на два основных этапа. Вначале фиксируются элементы стоящие на диагоналях матриц коэффициентов. Всего $6 \cdot 3 \cdot 3 = 54$ элемента. В силу специфики системы имеется ряд симметрий, которые позволяют из нескольких эквивалентных наборов оставить только один, тем самым сократив перебор. Во-первых, циклическая

[1] Указаны нижние оценки для произвольного поля и верхние оценки для кольца $\mathbb{Z}$.

В 2022 году была получена оценка $L(\langle 4, 4, 4\rangle) \leqslant 47$ над $\mathbb{F}_2$ [6]

В 2025 году была получена оценка $L(\langle 4, 4, 4\rangle) \leqslant 48$ над $\mathbb{Z}[\frac{1}{2}, i]$ [7]

перестановка диагоналей внутри каждой тройки комплектов. Всего 3^6 симметрий. Во-вторых, произвольные перестановки комплектов. Ещё 6! симметрий. В-третьих, любые одновременные перестановки элементов внутри диагоналей. Ещё 3! симметрий. В-четвёртых, одновременная нечётная перестановка матриц внутри комплектов. Ещё 2 симметрии. Кроме того, имеется 11 уравнений, содержащих только диагональные элементы. Программа, реализующая первый этап перебора генерирует все наборы диагональных элементов, которые удовлетворяют этим 11 уравнениям и только их. Причём из каждого класса относительно указанных симметрий генерируется ровно один набор.

Перебор организован следующим образом. Каждая тройка диагоналей одного комплекта собирается в матрицу 3×3, которая кодируется единственным целым числом. Среди этих матриц выбираются лишь те, которые не содержат нулевых столбцов, т.к. матрицы содержащие нулевой столбец не вносят вклада в указанные уравнения. Составляется список кодов уникальных относительно циклического сдвига матриц (из класса эквивалентных выбирается матрица с наименьшим кодом). Та же операция производится с матрицами содержащими нулевой столбец, в дальнейшем они будут добавлены к сгенерированному набору в случае необходимости. Программа содержит функцию `unfold`, которая вычисляет вклад каждого комплекта в указанные уравнения, и функцию `apply_group_action`, которая по заданным кодам матрицы и элемента группы переставляющей её строки и столбцы вычисляет код результата применения этого элемента. Обе функции неоднократно вызываются в процессе работы программы. Т. к. они принимают и возвращают целые числа, а также количество возможных входов для них достаточно ограничено, применяется кэширование, для достижения максимальной эффективности.

Для оптимизации относительно перестановок комплектов генерируются только наборы упорядоченные по возрастанию кодов (одинаковые пары сокращаются из-за работы в характеристике 2, поэтому их добавление откладывается на более поздние этапы генерации). Вместе с каждым префиксом набора поддерживается множество его нетривиальных автоморфизмов. При добавлении очередного кода к префиксу, во-первых, рассматриваются только бóльшие коды, а, во-вторых, если какой-то нетривиальный автоморфизм переводит этот код в меньший по величине, то такой префикс точно не минимальный и этот код рассматривать не нужно. Однако для корректной работы

этого алгоритма необходимо добавлять коды пакетами, так чтобы коды внутри разных пакетов точно не могли переводиться друг в друга автоморфизмами. Для этого множество всех кодом разбивается на орбиты относительно действия группы автоморфизмов, и в действительности коды перебираются в порядке возрастания номера орбиты, а не отдельного кода.

Когда добавление кодов закончено, происходит проверка, что в итоге получено решение системы из 11 уравнений. Если сгенерированный префикс является решением, то к нему (если его длина всё ещё меньше 6) добавляются всевозможные пары одинаковых кодов и/или коды матриц с нулевым столбцом при помощи модификации описанного выше алгоритма. В противном случае префикс отбрасывается.

В итоге программа генерирует 4612257 уникальных наборов. Целью этого этапа является зафиксировать достаточно большое множество значений переменных для того, чтобы из полученных уравнений было удобно извлекать следствия, вместе с тем минимизировать количество вариантов, используя симметрии системы уравнений.

На втором этапе перебора строится полная система уравнений, которая переводится в представление sat решателя. Для каждого из построенных на предыдущем этапе варианта фиксируется соответствующее множество значений переменных. Затем sat решатель проверяет существование решения при данных предположениях. Для ускорения процесса выполнения используется запоминание состояния решателя перед наложением дополнительных ограничений и запуск нескольких решателей параллельно.

Программа была проверена для случая $n = 2$. Решение соответствующее алгоритму Штрассена корректно находится.

При $n = 3$ было установлено отсутствие решений в описанном классе. Все вычисления были выполнены на персональном компьютере. Время выполнения около 98 часов.

Исходный код размещён по адресу `https://github.com/AIV5/cw2025`

4 Полученные результаты

Написана программа для перебора поиска билинейного алгоритма умножения матриц 3×3 над полем $\mathbb{F}_2$ сложности 19 в определённом классе. Путём полного перебора было установлено, что решения такого вида не существует.

Список литературы

[1] Strassen V. Gaussian elimination is not optimal // Numer. Math. 1969, № 13. p. 354–356.

[2] Winograd S. On multiplication of 2×2 matrices // Linear Algebra and Appl. 1971, № 4. p. 381–388.

[3] Alekseyev V. B. On the complexity of some algorithms of matrix multiplication // Journal of Algorithms. 1985, № 6. p. 71–85.

[4] Алексеев В. Б., Смирнов А. В. О точной и приближённой сложностях умножения матриц размеров 4×2 и 2×2 // Современные проблемы математики. 2013, Вып. 17. С. 135–152.

[5] Laderman J. D. A noncommutative algorithm for multiplying 3×3 matrices using 23 multiplications // Bull. Amer. Math. Soc. 1976, № 82 p. 126–128.

[6] Fawzi A. et al. Discovering faster matrix multiplication algorithms with reinforcement learning // Nature. 2022, Т. 610. – №. 7930. – С. 47–53.

[7] Novikov A. et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. [pdf]