Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №2 по курсу "Анализ алгоритмов"

Тема Алгоритмы умножения матриц

Студент Цветков И.А.

Группа ИУ7-53Б

Оценка (баллы)

Преподаватель Волкова Л. Л.

Москва — 2021 г.

Содержание

Введение **3**

1 Аналитическая часть **4**

1.1 Матрица 4

1.2 Стандартный алгоритм 4

1.3 Алгоритм Копперсмита-Винограда 5

2 Конструкторская часть **7**

2.1 Описание используемых типов данных 7

2.2 Сведения о модулях программы 7

2.3 Схемы алгоритмов 7

2.4 Модель вычислений 13

2.5 Трудоемкость алгоритмов 13

2.5.1 Стандартный алгоритм умножения матриц 13

2.5.2 Алгоритм Копперсмита-Винограда 14

2.5.3 Оптимизированный алгоритм Копперсмита-Винограда . 15

2.6 Классы эквивалентности при тестировании 16

2.7 Вывод 16

Список литературы **17**

Введение

В математике и программировании часто приходится прибегать к использованию матриц. Существует огромное количество областей их применения в этих сферах. Например, матрицы активно испольуются при выводе различных формул в физике:

- градиент;
- дивергенция;
- ротор.

Также часто применяются и операции над матрицами - сложение, возведение в степень, умножение. При различных задачах размеры матрицы могут достигать большtх значений. Поэтому оптимизация операций работы над матрицами является важной задачей в программировании. Об оптимизации операции умножения пойдет речь в данной лабораторной работе.

Целью данной работы является изучение, реализация и исследование алгоритмов умножения матриц - классический алгоритм, алгоритм Винограда и оптимизированный алгоритм Винограда. Для достижения поставленной цели необходимо выполнить следующие задачи:

- изучить и реализовать алгоритмы - классический, Винограда и его оптимизацию;
- провести тестирование по времени и по памяти для алгоритмов лабораторной работы;
- провести сравнительный анализ по времени классического алгоритма и алгоритма Винограда;
- провести сравнительный анализ по времени алгоритма Винограда и его оптимизации;
- описать и обосновать полученные результаты в отчете о выполненной лабораторной работе, выполненного как расчётно-пояснительная записка к работе.

1 Аналитическая часть

В этом рзаделе будут представлены классический алгоритм умножения матриц и алгоритм Винограда.

1.1 Матрица

Матрица [1] - математический объект, который представляет собой двумерный массив, в котором элементы располагаются по строкам и столбцам.

Пусть A - матрица, тогда $A_{i,j}$ - элемент этой матрицы, который находится на *i-ой* строке и *j-ом* столбце.

Можно выделить следующие операции над матрицами:

- матрицы одинакового размера можно складывать и вычитать;
- количество столбцов одной матрицы равно количеству строк другой матрицы - их можно перемножить, причем количество строк будет, как у первой матрицы, а столбцов - как у второй.

Замечание: операция умножения матриц некоммутативна - если A и B - квадтраные матрицы, а C - результат их перемножения, то произведение AB и BA дадут разный результат C.

1.2 Стандартный алгоритм

Пусть даны две матрицы

$$A_{lm} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{l1} & a_{l2} & \dots & a_{lm} \end{pmatrix}, \quad B_{mn} = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{pmatrix}, \tag{1.1}$$

тогда матрица C

$$C_{ln} = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{l1} & c_{l2} & \dots & c_{ln} \end{pmatrix}, \tag{1.2}$$

где

$$c_{ij} = \sum_{r=1}^{m} a_{ir} b_{rj} \quad (i = \overline{1,l}; j = \overline{1,n}) \tag{1.3}$$

будет называться произведением матриц A и B.

Стандартный алгоритм реализует данную формулу.

1.3 Алгоритм Копперсмита-Винограда

Алгоритм Копперсмита-Винограда [2] — алгоритм умножения квадратных матриц. Начальная версия имела асимптотическую сложность алгортма около $O(n^{2,3755})$, где n - размер стороны матрицы, но после доработки он стал обаладть лучшей асимптотикой среди всех алгоритмов умножения матриц.

Рассмотрим два вектора $V = (v_1, v_2, v_3, v_4)$ и $W = (w_1, w_2, w_3, w_4)$. Их скалярное произведение равно: $V \cdot W = v_1w_1 + v_2w_2 + v_3w_3 + v_4w_4$, что эквивалентно (1.4):

$$V \cdot W = (v_1+w_2)(v_2+w_1)+(v_3+w_4)(v_4+w_3)-v_1v_2-v_3v_4-w_1w_2-w_3w_4. \tag{1.4}$$

Прирост производительности заключается в идее предварительной обработки - полученное выражение требует большего количества операций, чем стандартное умножение матриц, но выражение в правой части крайнего равенства можно вычислить заранее и запомнить для каждой строки первой матрицы и каждого столбца второй матрицы. Это позволит выполнить лишь два умножения и пять сложений, при учете, что потом будет сложено только с двумя предварительно посчитанными суммами соседних элементов текущих строк и столбцов. Операция сложения выполняется быстрее, поэтому на практике алгоритм должен работать быстрее обычного алогритма перемножения матриц.

Но стоит упомянуть, что при нечетном значении размера матрицы нужно дополнительно добавить произведения крайних элементов соответствующих строк и столбцов.

Вывод

В данном разделе были рассмотрены алгоритмы умножения матриц - стандартного и Винограда, который имеет большую эффективность за счет предварительных вычислений.

Comming soon...

2 Конструкторская часть

В этом разделе будут представлено описание используемых типов данных, а также схемы алгоритмов перемножения матриц - стандартного, Винограда и оптимизации алгоритма Винограда.

2.1 Описание используемых типов данных

Comming soon...

2.2 Сведения о модулях программы

Программа состоит из двух модулей:

- *main.py* - файл, содержащий весь служебный код;
- *algorythms.py* - файл, содержащий код всех алгоритмов перемножения матриц.

2.3 Схемы алгоритмов

На рисунке 2.1 представлена схема алгоритма для стандартного умножения матриц. На рисунках 2.2-2.3 схема алгоритма Винограда умножения матриц, а на 2.4-2.5 - схема оптимизированного алгоритма Винограда.

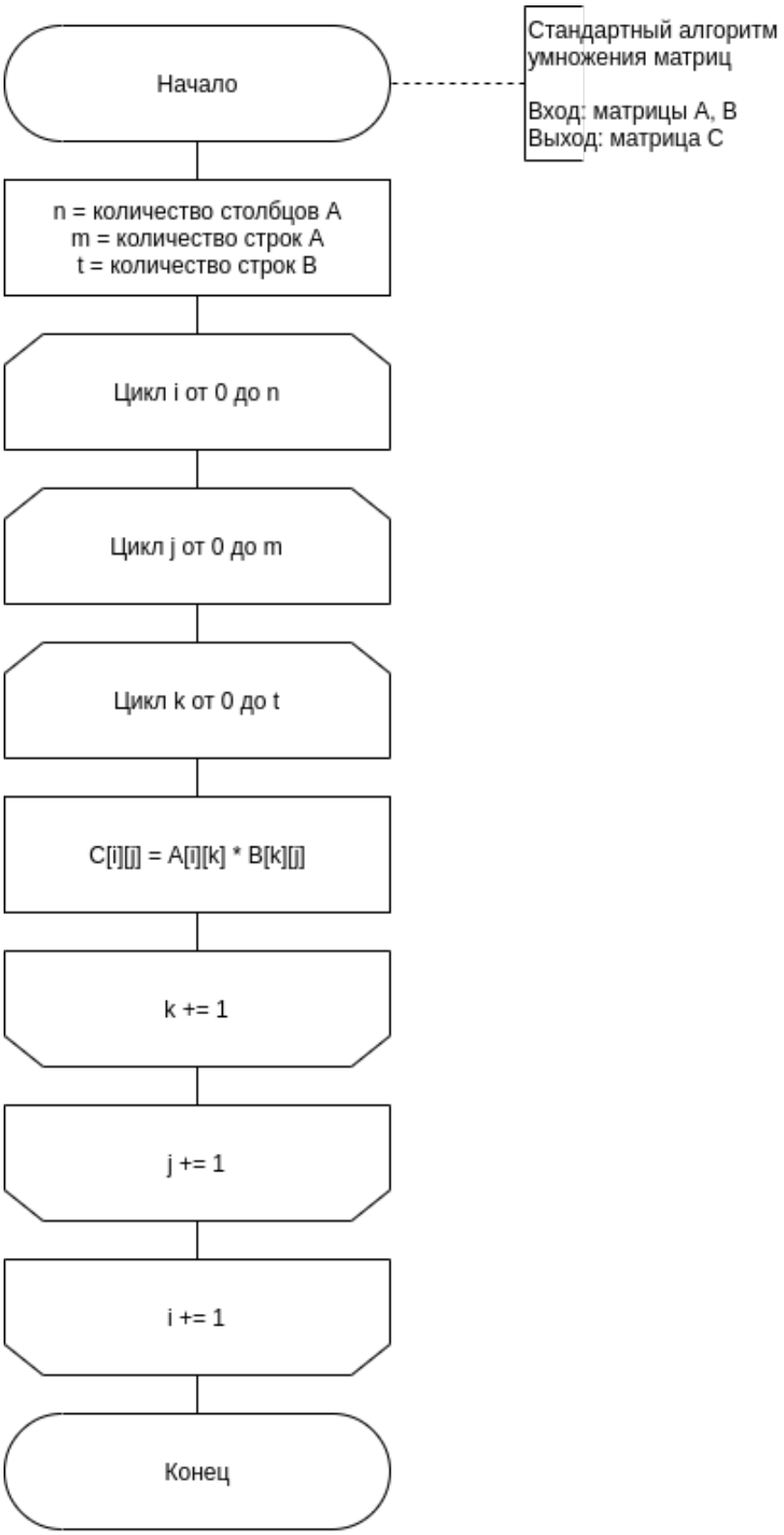

Рисунок 2.1 – Схема стандартного алгоритма умножения матриц

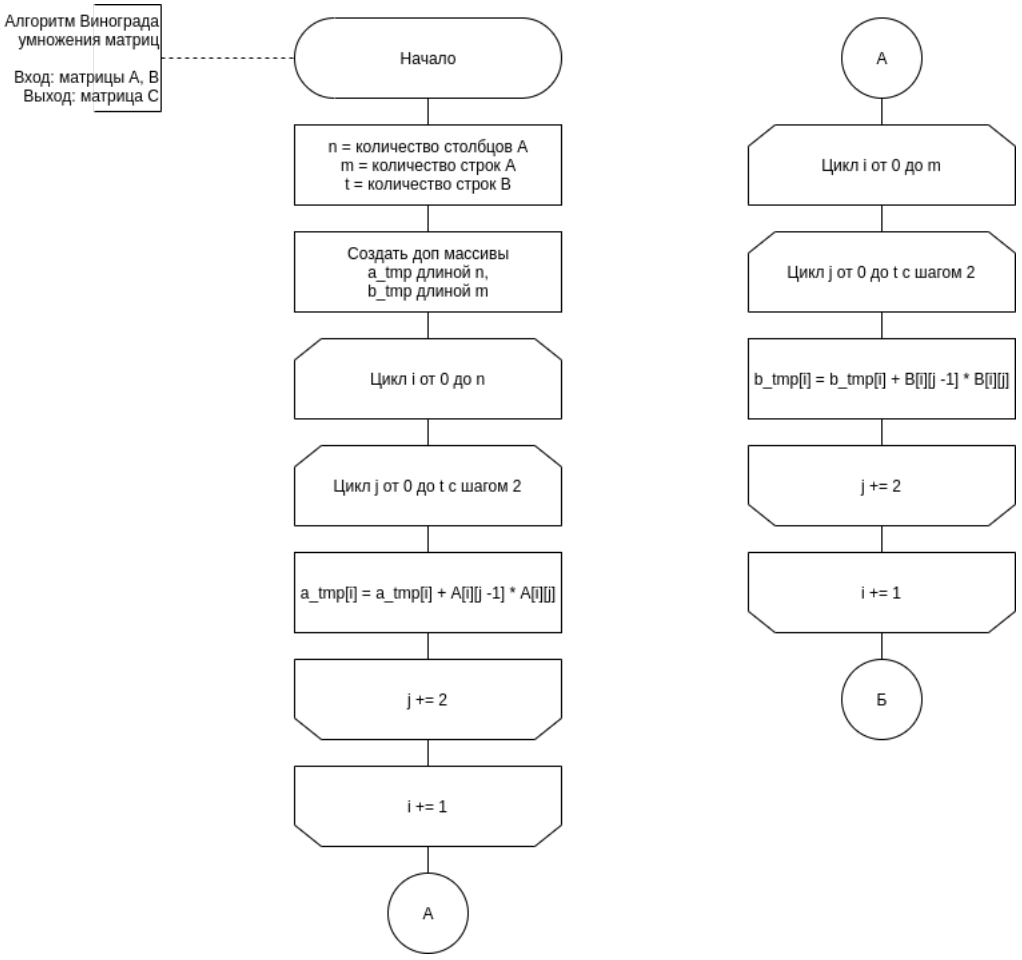

Рисунок 2.2 – Схема умножения матриц по алгоритму Винограда (часть 1)

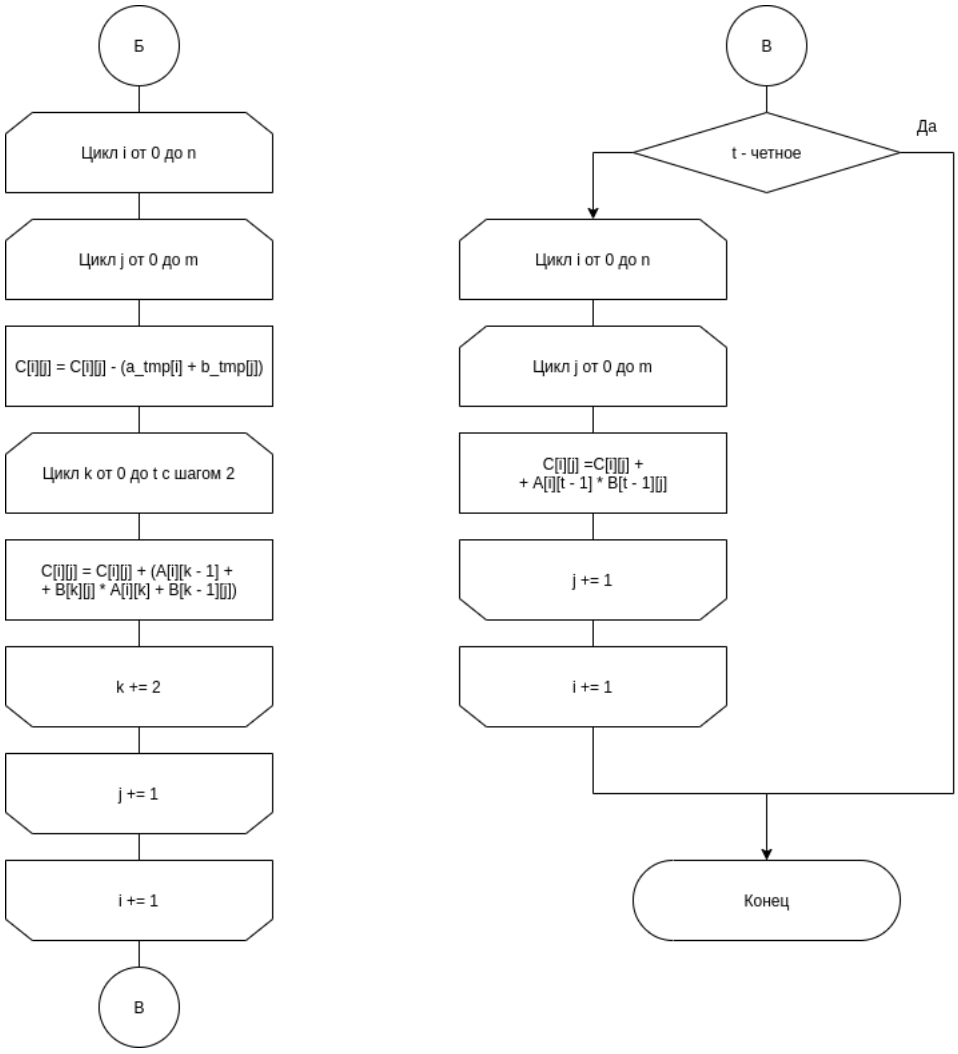

Рисунок 2.3 – Схема умножения матриц по алгоритму Винограда (часть 2)

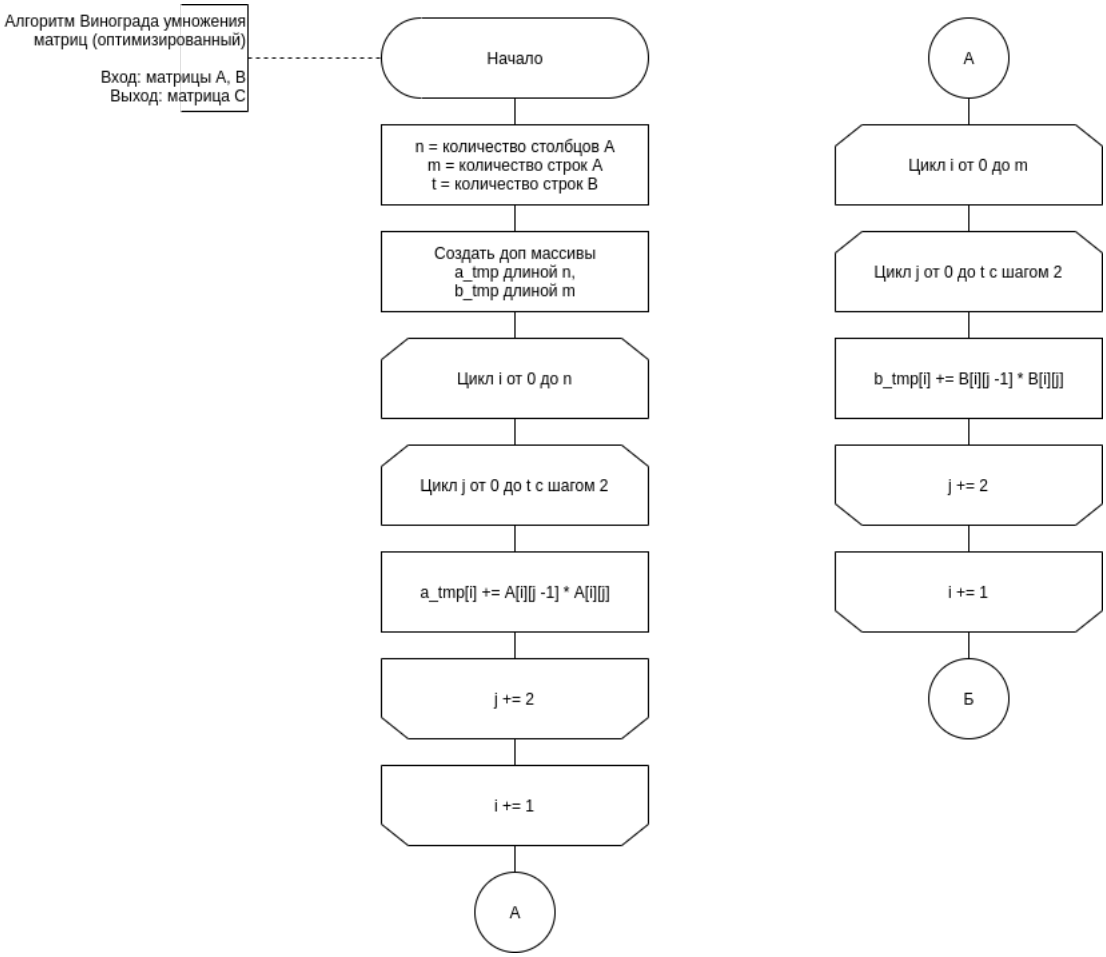

Рисунок 2.4 – Схема умножения матриц по оптимизированному алгоритму Винограда (часть 1)

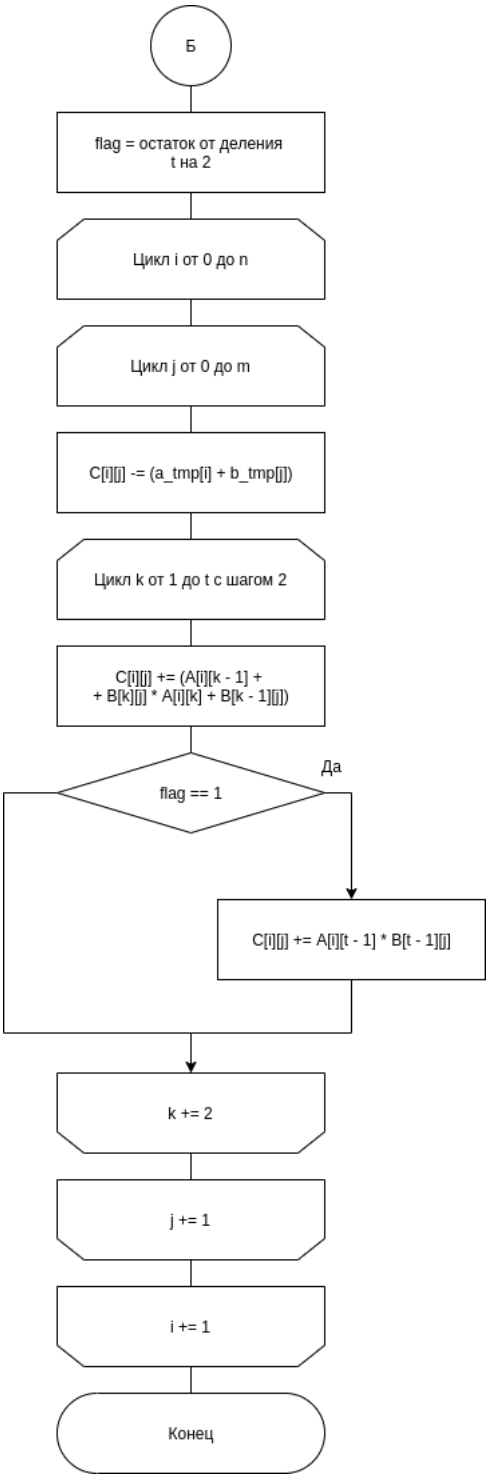

Рисунок 2.5 – Схема умножения матриц по оптимизированному алгоритму Винограда (часть 2)

2.4 Модель вычислений

Чтобы провести вычисление трудоемкости алгоритмов умножения матриц, введем модель вычислений [3]:

1. операции из списка (2.1) имеют трудоемкость 1;

$$+, -, *, /, \%, ==, !=, <, >, <=, >=, [], ++, --\tag{2.1}$$

2. трудоемкость оператора выбора `if условие then A else B` рассчитывается, как (2.2);

$$f_{if} = f_{\text{условия}} + \begin{cases} f_A, & \text{если условие выполняется,} \\ f_B, & \text{иначе.} \end{cases}\tag{2.2}$$

3. трудоемкость цикла рассчитывается, как (2.3);

$$f_{for} = f_{\text{инициализации}} + f_{\text{сравнения}} + N(f_{\text{тела}} + f_{\text{инкремента}} + f_{\text{сравнения}})\tag{2.3}$$

4. трудоемкость вызова функции равна 0.

2.5 Трудоемкость алгоритмов

Рассчитаем трудоемкость алгоритмов умножения матриц.

2.5.1 Стандартный алгоритм умножения матриц

Для стандартного алгоритма умножения матриц трудоемкость будет слагаться из:

- внешнего цикла по $i \in [1..M]$, трудоёмкость которого: $f = 2 + M \cdot (2 + f_{body})$;
- цикла по $j \in [1..N]$, трудоёмкость которого: $f = 2 + N \cdot (2 + f_{body})$;

- цикла по $k \in [1..K]$, трудоёмкость которого: $f = 2 + 10K$.

Поскольку трудоемкость стандартного алгоритма равна трудоемкости внешнего цикла, то:

$$f_{standard} = 2 + M \cdot (4 + N \cdot (4 + 10K)) = 2 + 4M + 4MN + 10MNK \approx 10MNK \tag{2.4}$$

2.5.2 Алгоритм Копперсмита-Винограда

Чтобы вычислить трудоемкость алгоритма Копперсмита-Винограда, нужно учесть следующее:

- создания и инициализации массивов a_tmp и b_tmp, трудоёмкость которых (2.5):

$$f_{init} = M + N; \tag{2.5}$$

- заполнения массива a_tmp, трудоёмкость которого (2.6):

$$f_{a\_tmp} = 2 + K(2 + \frac{M}{2} \cdot 11); \tag{2.6}$$

- заполнения массива b_tmp, трудоёмкость которого (2.7):

$$f_{b\_tmp} = 2 + K(2 + \frac{N}{2} \cdot 11); \tag{2.7}$$

- цикла заполнения для чётных размеров, трудоёмкость которого (2.8):

$$f_{cycle} = 2 + M \cdot (4 + N \cdot (11 + \frac{K}{2} \cdot 23)); \tag{2.8}$$

- цикла, который дополнительно нужен для подсчета значений при нечет-

ном размере матрицы, трудоемкость которого (2.9):

$$f_{last} = \begin{cases} 2, & \text{чётная,} \\ 4 + M \cdot (4 + 14N), & \text{иначе.} \end{cases} \tag{2.9}$$

Тогда для худшего случая (нечётный общий размер матриц) имеем (2.10):

$$f_{worst} = f_{a\_tmp} + f_{b\_tmp} + f_{cycle} + f_{last} \approx 11.5 \cdot MNK \tag{2.10}$$

Для лучшего случая (чётный общий размер матриц) имеем (2.11):

$$f_{best} = f_{a\_tmp} + f_{a\_tmp} + f_{cycle} + f_{last} \approx 11.5 \cdot MNK \tag{2.11}$$

2.5.3 Оптимизированный алгоритм Копперсмита-Виногр

Оптимизация заключается в:

- использовании побитового сдвига вместо деления на 2;
- цикл, который был вынесен при нечетном размере матрцы, был занесен в общий цикл, тем самым в общем цикле происходят дополнительные вычиления при нечетном размере матрицы;
- операции сложения и вычитания заменены на операции $+=$ и $-=$ соответственно.

Тогда трудоемкость оптимизированного алгоритма Копперсмита-Винограда состоит из:

- создания и инициализации массивов a_tmp и b_tmp (2.5);
- заполнения массива a_tmp, трудоёмкость которого (2.6);
- заполнения массива MV, трудоёмкость которого (2.7);
- цикла заполнения для чётных размеров, трудоёмкость которого (2.12):

$$f_{cycle} = 2 + M \cdot (4 + N \cdot (11 + \frac{K}{2} \cdot 18)); \tag{2.12}$$

- условие, которое нужно для дополнительных вычислений при нечетном размере матрицы, трудоемкость которого (2.13):

$$f_{last} = \begin{cases} 1, & \text{чётная,} \\ 4 + M \cdot (4 + 10N), & \text{иначе.} \end{cases} \tag{2.13}$$

Тогда для худшего случая (нечётный общий размер матриц) имеем (2.14):

$$f_{worst} = f_{MH} + f_{MV} + f_{cycle} + f_{last} \approx 9MNK \tag{2.14}$$

Для лучшего случая (чётный общий размер матриц) имеем (2.15):

$$f_{best} = f_{MH} + f_{MV} + f_{cycle} + f_{last} \approx 9MNK \tag{2.15}$$

2.6 Классы эквивалентности при тестировании

Comming soon...

2.7 Вывод

В данном разделе были построены схемы алгоритмов умножения матриц рассматриваемых в лабораторной работе, были описаны классы эквивалентности для тестирования, модули программы, а также проведена теоретическая оценка трудоемкости алогритмов.

Список литературы

[1] Матрица [Электронный ресурс]. Режим доступа: https://terme.ru/termin/matrica.html (дата обращения: 23.10.2021).

[2] Умножение матриц [Электронный ресурс]. Режим доступа: http://algolib.narod.ru/Math/Matrix.html (дата обращения: 23.10.2021).

[3] М. В. Ульянов Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ. 2007.