

Отчет по лабораторной работе №2
по курсу "Анализ алгоритмов"
по теме "Изучение алгоритмов умножения
матриц "

Студент: Барсуков Н.М. ИУ7-56
Преподаватель: Волкова Л.Л., Строганов Ю.В.

Содержание

1	Аналитическая раздел	3
1.1	Постановка задачи	3
1.2	Классический подход	3
1.3	Алгоритм Винограда	4
1.4	Оптимизированный алгоритм Винограда	4
1.5	Вывод	4
2	Конструкторский раздел	5
2.1	Алгоритмы	5
2.2	Модель трудоемкости	7
2.3	Расчет сложности алгоритмов	8
2.3.1	Классический алгоритм	8
2.3.2	Алгоритм Винограда	8
2.3.3	Мод. Алгоритм Винограда	8
2.3.4	Вывод	8
3	Технологический раздел	9
3.1	Минимальные требования	9
3.2	Выбор языка и среды разработки	9
3.3	Интерфейс	9
3.4	Листинг	9
3.5	Вывод	13
4	Исследовательский раздел	14
4.0.1	Debug	14
4.0.2	Release	15
4.1	Вывод	16
	Список литературы	17

Введение

Умножение матриц — это один из базовых алгоритмов, который широко применяется в различных численных методах, и в частности в алгоритмах машинного обучения. Многие реализации прямого и обратного распространения сигнала в сверточных слоях нейронной сети базируются на этой операции. Так порой до 90-95% всего времени, затрачиваемого на машинное обучение, приходится именно на эту операцию. Так же это один из немногих алгоритмов, который позволяет эффективно задействовать все вычислительные ресурсы современных процессоров и графических ускорителей. Поэтому не удивительно, что многие алгоритмы стараются свести к матричному умножению — дополнительная расходы, связанные с подготовкой данных, как правило с лихвой окупаются общим ускорением алгоритмов

1 Аналитическая раздел

В этом разделе описаны задачи, необходимые для достижения цели. Приведено математическое и детальное описание трех алгоритмов умножения матриц:

1. классический;
2. винограда;
3. винограда Модифицированный.

1.1 Постановка задачи

Цель: изучить и реализовать алгоритмы умножения матриц

Для достижения поставленной цели требуется решить следующие задачи:

1. изучить
 - (a) классический алгоритм умножения;
 - (b) алгоритм Винограда;
 - (c) модифицированный Винограда.
2. реализовать указанные выше алгоритмы;
3. разработать и реализовать оптимизированный алгоритм Винограда;
4. выбрать модель оценки трудоемкости;
5. сделать замеры алгоритмов;
6. сравнить теоритические и экспериментальные оценки трудоемкости;
7. сделать вывод.

В данном разделе приведено математическое описание алгоритмов перемножения матриц. Буду рассмотрены 3 подхода: классический, алгоритм Винограда и оптимизированный алгоритм Винограда.

1.2 Классический подход

Предположим, что необходимо получить матрицу $C_{(a,c)} = A_{(a,b)} * B_{(b,c)}$. Для нахождения значений элементов матрицы $C_{(a,c)}$ используют следующие выражение[1]

$$C_{i,j} = \sum_K (A_{i,k} B_{k,j}) \quad (1)$$

Классический алгоритм напрямую реализует эту формулу

1.3 Алгоритм Винограда

Можно заметить, что элементы из суммы выражения 1 можно переписать как:

$$A_{i,k-1}B_{k-1,j} + A_{i,k}B_{k,j} = (A_{i,k-1} + B_{k,j})(A_{i,k} + B_{k-1,j}) - A_{i,k-1}A_{i,k} - B_{k-1}B_{k,j} \quad (2)$$

т. е. как сумму произведения сумм и двух произведений. Учитывая, что упомянутые два произведения можно рассчитать заранее для обработки двух элементов матрицы теперь нужно не сложение и два умножения, а умножение и два сложения, что проще с точки зрения вычислений. Таким образом, алгоритм Винограда состоит в следующем:

1. совершить расчет заранее двух произведений для каждого ряда и столбца матрицы-результата (одно произведение считается для ряда, другое для столбца). Для хранения результатов используется промежуточный буфер;
2. по вышеприведённой формуле осуществить расчёт каждого элемента матрицы;
3. в случае, если в произведении $A_{(a,b) \times B_{b,c}}$ b – нечётное число, пройти во второй раз по матрице, дополняя элементы $C_{i,j}$ недостающим элементом (который не был описан вышеописанной суммой). Можно заметить, что пункт 3 необходимо выполнять только в некоторых случаях, но если это происходит, то получается существенное увеличение времени работы алгоритма.

1.4 Оптимизированный алгоритм Винограда

1. Внутри тройного цикла накапливать результат в буфер, а вне цикла сбрасывать его в ячейку матрицы.
2. Заменить $MulH[i] = MulH[i] + \dots$ на $MulH[i] += \dots$ (аналогично для $MulV$), где $MulH$ и $MulV$ – временные массивы для предварительного расчета сумм произведений

1.5 Вывод

В данном разделе детально описаны 3 алгоритма умножения матриц: Классический, Винограда, модифицированная версия Винограда.

2 Конструкторский раздел

В данном разделе представлены блок-схемы 2 алгоритмов умножения матриц: классический, Винограда, и схема вычисления rowFactor. Описана структура программы. Введена модель трудоемкости. Вычислены сложности алгоритмов.

2.1 Алгоритмы

В данном подразделе представлены схемы трех следующих алгоритмов умножения матриц: рисунок (1) Классический, рисунок (2) Винограда и схема вычисления rowFactor (3)

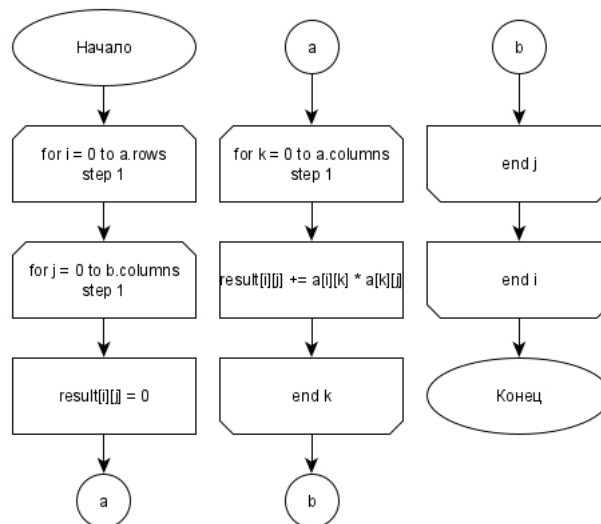


Рис. 1: схема классического алгоритма

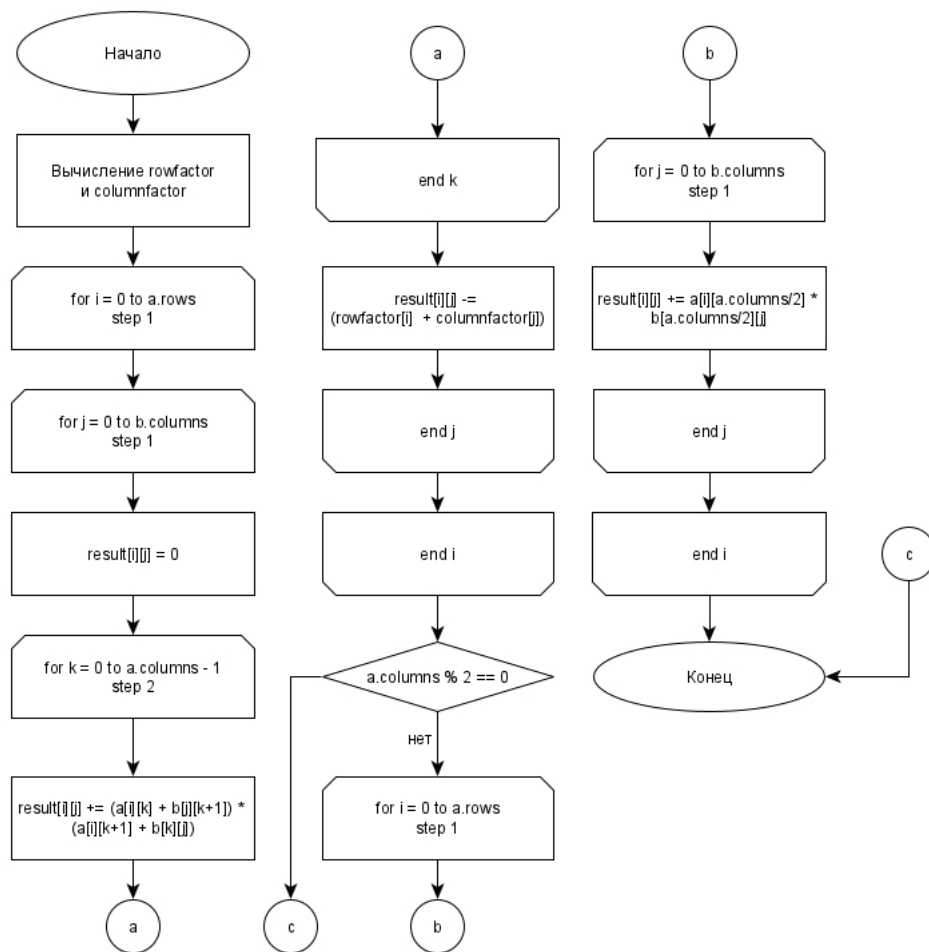


Рис. 2: схема алгоритма Винограда

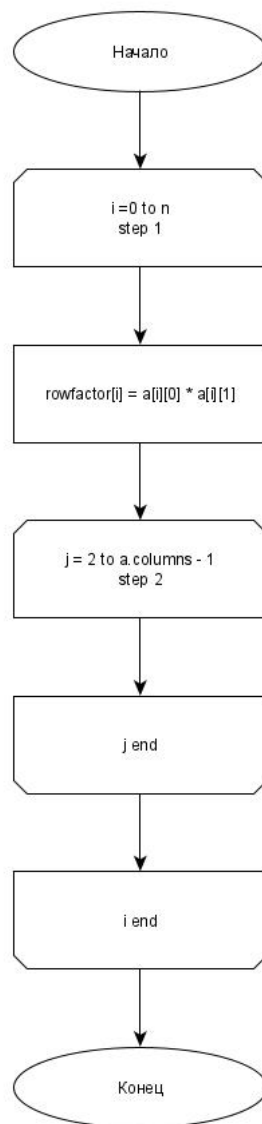


Рис. 3: Схема вычисления rowFactor

2.2 Модель трудоемкости

Основные правила оценки трудоемкости:

1. Операции единичной трудоемкости: $+$, $-$, $*$, $/$,
2. C - подобная модель оценки трудоемкости циклов:
 - (a) **for** (**int** $i = 0$; $i < N$; $i++$){}
 $A = 2 + N * (2 + A())$;
 - (b) **for** (**int** $i = 0$; $i < N + 1$; $i++$){}
 $A = 3 + N * (3 + A())$.

2.3 Расчет сложности алгоритмов

2.3.1 Классический алгоритм

$$\begin{aligned} C[NxM] &= A[NxQ] * B[QxM] \\ O(i, j,) &= 2 + 4N + 7NM + 12QMN \\ \text{Цикл i} &= 2 + 4N + 7NM + 12QMN \\ \text{Цикл j} &= 2 + 7M + 12QM \\ \text{Цикл индекс} &= 2 + 12Q \\ \text{Тело индекс} &= 10 \end{aligned} \tag{3}$$

2.3.2 Алгоритм Винограда

1. цикл 1:

$$2 + Q(12 + \frac{1}{2} * 13N) \tag{4}$$

2. цикл 2:

$$2 + M(12 + \frac{13}{2} * Q) \tag{5}$$

3. цикл 3:

$$2 + N(4 + M * (17 + 11 * Q)) \tag{6}$$

4. цикл 4 (Четное или нечетное):

$$2 + N * (4 + 13 * M) \tag{7}$$

2.3.3 Мод. Алгоритм Винограда

1. цикл 1:

$$2 + N * (6 + 5Q) \tag{8}$$

2. цикл 2:

$$2 + M * (6 + 5Q) \tag{9}$$

3. цикл 3:

$$2 + N * (4 + M * (11 + Q)) \tag{10}$$

4. цикл 4 (Четное или нечетное):

$$3 + N * (4 + 10 * M) \tag{11}$$

2.3.4 Вывод

В данном разделе приведены схемы алгоритмов, приведена модель трудоемкости, проведен расчет сложности алгоритмов.

3 Технологический раздел

В данном разделе указаны минимальные системные требования. Описан используемый язык и среда разработки. Представлено описание интерфейса и его скриншоты. Приведен листинг 3 алгоритмов умножения матрицы.

3.1 Минимальные требования

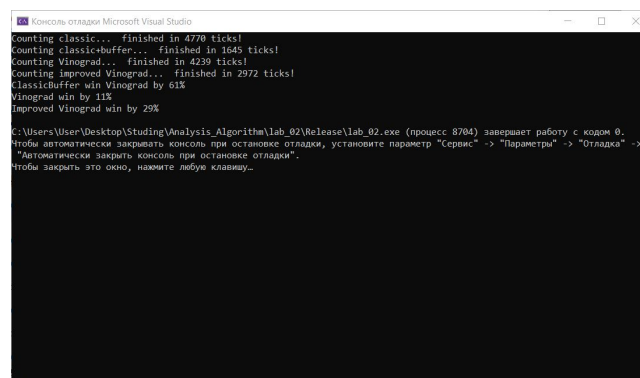
Минимальные системные требования: РС с операционной системой Windows XP/Vista/7/8/10. Требуется устройства ввода: клавиатура, мышь. Устройство вывода: монитор.

3.2 Выбор языка и среды разработки

Для решения данной поставленной задачи, мной был выбран язык с++ стандарта c11 по причине использования ООП. Так же использовалась среда Visual Studio 2019

3.3 Интерфейс

Интерфейс представляет из себя простую консольную команду в которой пользователь видит временные результаты умножения матриц и результаты сравнения в процентном соотношении. Ниже представлен скриншот выполнения (4). Данный интерфейс выбран из за простоты и удобства.



```
Консоль отладки Microsoft Visual Studio
Counting classic... finished in 4770 ticks!
Counting classic+buffer... finished in 1425 ticks!
Counting Vinograd... finished in 4229 ticks!
Counting improved Vinograd... finished in 2972 ticks!
ClassicBuffer win Vinograd by 61%
Vinograd win by 11%
Improved Vinograd win by 29%
C:\Users\User\Desktop\Studying\Analysis_Algorithm\lab_02\Release\lab_02.exe (процесс 8704) завершает работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" ->
"Автоматически закрывать консоль при остановке отладки".
Чтобы закрыть это окно, нажмите любую клавишу...
```

Рис. 4: Скриншот программы

3.4 Листинг

В данном подразделе приведены листинги трех различных алгоритмов умножения матриц:

1. классический 1
2. Винограда 2

3. Винограда модифицированный 3

Листинг 1: Классическое умножение матриц

```
1
2 Matrix Matrix::classicMult(const Matrix& other ,
   std::clock_t& time) {
3     if (columns != other.rows) {
4         throw std::logic_error("Sizes_don't_match!");
5     }
6     Matrix result(rows, other.columns);
7
8     std::clock_t start = std::clock();
9
10    for (unsigned i = 0; i < rows; ++i) {
11        for (unsigned j = 0; j < columns; ++j) {
12            data[i][j] = 0;
13            for (unsigned index = 0; index < columns;
14                ++index) {
15                data[i][j] = data[i][j] + data[i][index]
16                    * other.data[index][j];
17            }
18        }
19    }
20    time = std::clock() - start;
21    return result;
22 }
```

Листинг 2: Винограда

```
1 Matrix Matrix::vinogradMult(const Matrix& other ,
   std::clock_t& time) {
2     if (columns != other.rows) {
3         throw std::logic_error("Sizes_don't_match!");
4     }
5     Matrix result(rows, other.columns);
6
7     unsigned* rowFactor = new unsigned[rows];
8     unsigned* columnFactor = new unsigned[columns];
9
10    std::clock_t start = std::clock();
11
12    for (unsigned i = 0; i < rows; ++i) {
13        rowFactor[i] = data[i][0] * data[i][1];
14        for (unsigned j = 2; j < columns - 1; j += 2) {
15            rowFactor[i] = rowFactor[i] + data[i][j] *
16                data[i][j + 1];
17        }
18    }
19 }
```

```

18
19     for (unsigned i = 0; i < other.columns; ++i) {
20         columnFactor[i] = other.data[0][i] *
21             other.data[1][i];
22         for (unsigned j = 2; j < other.rows - 1; j += 2)
23             {
24                 columnFactor[i] = columnFactor[i] +
25                     other.data[j][i] * other.data[j + 1][i];
26             }
27     }
28
29     for (unsigned i = 0; i < rows; ++i) {
30         for (unsigned j = 0; j < other.columns; ++j) {
31             result.data[i][j] = 0;
32             for (unsigned k = 0; k < columns - 1; k +=
33                 2) {
34                 result.data[i][j] = result.data[i][j] +
35                     (data[i][k] + other.data[k + 1][j]) *
36                     (data[i][k + 1] + other.data[k][j]);
37             }
38             result.data[i][j] = result.data[i][j] -
39                 (rowFactor[i] + columnFactor[j]);
40         }
41     }
42
43     if (columns % 2 != 0) {
44         unsigned columnsDiv2 = columns / 2;
45         for (unsigned i = 0; i < rows; ++i) {
46             for (unsigned j = 0; j < other.columns; ++j)
47                 {
48                     result.data[i][j] = result.data[i][j] +
49                         data[i][columnsDiv2] *
50                         other.data[columnsDiv2][j];
51                 }
52         }
53
54     time = std::clock() - start;
55
56     delete[] columnFactor;
57     delete[] rowFactor;
58
59     return result;
60 }

```

Листинг 3: Винограда модифицированный

```

1 Matrix Matrix::vinogradImprMult(const Matrix& other,
2     std::clock_t& time) {
3     if (columns != other.rows) {

```

```

3         throw std::logic_error("Sizes_don't_match!");
4     }
5     Matrix result(rows, other.columns);
6
7     unsigned* rowFactor = new unsigned[rows];
8     unsigned* columnFactor = new unsigned[columns];
9
10    std::clock_t start = std::clock();
11
12    for (unsigned i = 0; i < rows; ++i) {
13        rowFactor[i] = data[i][0] * data[i][1];
14        unsigned buffer = rowFactor[i];
15        for (unsigned j = 2; j < columns - 1; j += 2) {
16            buffer += data[i][j] * data[i][j + 1];
17        }
18        rowFactor[i] += buffer;
19    }
20
21    for (unsigned i = 0; i < other.columns; ++i) {
22        columnFactor[i] = other.data[0][i] *
23            other.data[1][i];
24        unsigned buffer = columnFactor[i];
25        for (unsigned j = 2; j < other.rows - 1; j += 2) {
26            buffer += other.data[j][i] * other.data[j +
27                1][i];
28        }
29        columnFactor[i] = buffer;
30    }
31
32    for (unsigned i = 0; i < rows; ++i) {
33        for (unsigned j = 0; j < other.columns; ++j) {
34            unsigned buffer = 0;
35            for (unsigned k = 0; k < columns - 1; k +=
36                2) {
37                buffer += (data[i][k] + other.data[k +
38                    1][j]) *
39                    (data[i][k + 1] + other.data[k][j]);
40            }
41            result.data[i][j] = buffer;
42            result.data[i][j] -= (rowFactor[i] +
43                columnFactor[j]);
44        }
45    }
46
47    if (columns % 2 != 0) {
48        unsigned columnsDiv2 = columns / 2;
49        for (unsigned i = 0; i < rows; ++i) {
50            for (unsigned j = 0; j < other.columns; ++j)
51                {

```

```

46         result.data[i][j] +=
47             data[i][columnsDiv2] *
48             other.data[columnsDiv2][j];
49     }
50 }
51 time = std::clock() - start;
52
53 delete[] columnFactor;
54 delete[] rowFactor;
55
56 return result;
57 }

```

3.5 Вывод

В данном разделе представлены минимальные системные требования. Выбран язык разработки и среда. Описан интерфейс. Представлены листинги программ.

4 Исследовательский раздел

В данном разделе приведены характеристики машины на которых проводилось тестирование алгоритмов. Указаны замеры времени выполнения трех алгоритмов умножения матриц: Классического, Винограда и Винограда модифицированного.

1. Компьютер:

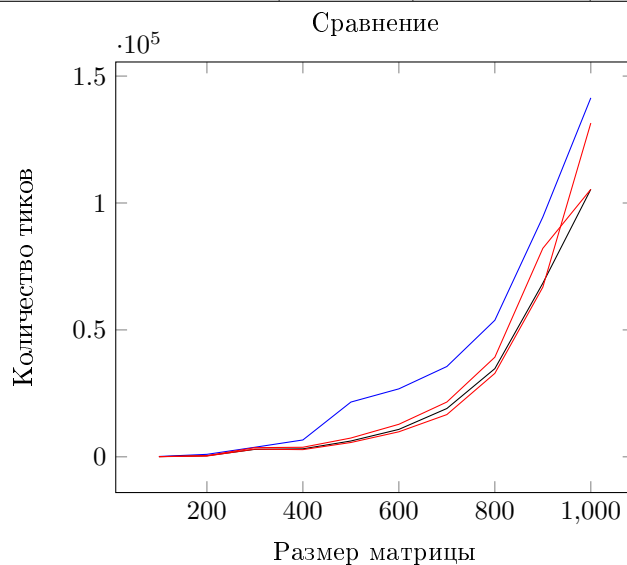
- (a) Тип компьютера Компьютер с ACPI на базе x64;
- (b) Операционная система Microsoft Windows 10 Pro.

2. Системная плата:

- (a) тип ЦП DualCore Intel Core i5-6200U, 2700 MHz (27 x 100);
- (b) системная плата HP 8079;
- (c) чипсет системной платы Intel Sunrise Point-LP, Intel Skylake-U;
- (d) системная память 8072 МБ (DDR4 SDRAM).

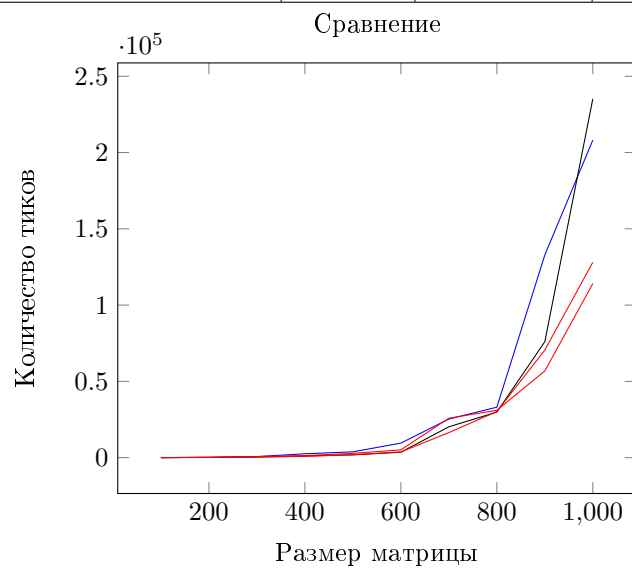
4.0.1 Debug

Размерность Алгоритм	Классик	Классик ул.	Винограда	Винограда ул.
100	140	78	78	62
200	982	422	452	312
300	3775	2949	3615	3026
400	6645	3145	3768	2839
500	21575	6240	7410	5648
600	26782	10811	12823	9865
700	35584	19036	21590	16645
800	53805	34741	39250	32885
900	94474	68406	82135	66846
1000	141414	105394	111638	131493



4.0.2 Release

Размерность Алгоритм	Классик	Классик ул.	Винограда	Винограда ул.
100	31	31	31	31
200	219	125	249	219
300	733	438	702	421
400	2544	1138	1373	845
500	3822	1826	2826	1966
600	9549	3562	5109	3764
700	25319	20239	25877	16430
800	33092	29881	31061	30251
900	132864	76041	56852	70562
1000	208147	235230	114093	128083



4.1 Вывод

Можно заметить значительно преимущество алгоритма Винограда над классическим алгоритмом, которое растет пропорционально размерности матриц. Улучшения для алгоритма Винограда дают чуть лучший результат, чем оригинальный алгоритм, но не всегда. Намного большую эффективность дает добавление флага оптимизации компилятора 03.

Заключение

В ходе работы были исследованы два алгоритма умножения матриц: классический и Винограда. На квадратных матрицах размера от 100 до 1000, заполненных произвольными числами, было проведено сравнение скорости этих алгоритмов между собой, а также с их оптимизированными программными методами вариантами. Исследования проводились при компиляции с оптимизациями и без. Установлено:

1. классический модифицированный алгоритм работает быстрее Винограда на 56%;
2. виноград работает быстрее на 27% чем Классический алгоритм;
3. виноград улучшенный работает на 44% быстрее Классического;

Список литературы

- [1] Дж. Макконнелл. Анализ алгоритмов. Активный обучающий подход.- М.: Техносфера, 2009.
- [2] Методика идентификации пассажира по установленным данным В. М . Черненький , Ю. Е . Гапанюк [Электронный ресурс]. – Режим доступа: <http://www.engjournal.ru/articles/89/89.pdf>, свободный – (07.10.2019)
- [3] Библиография в LaTeX [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/114997/>, свободный – (26.09.2019) <https://habr.com/ru/post/114997/>
- [4] The Listings Package [Электронный ресурс]. - Режим доступа: <http://texdoc.net/texmf-dist/doc/latex/listings/listings.pdf>, свободный - (11.12.2019)