Государственное образовательное учреждение высшего профессионального образования

"Московский государственный технический университет имени Н.Э.Баумана"

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №4
курса «Анализ алгоритмов»
на тему
«Распараллеливание вычислений для алгоритма Винограда»

Студент: Ильясов И.М., группа ИУ7-53Б

Преподаватели: Волкова Л.Л., Строганов Ю.В.

Москва, 2019 г.

Содержание

Введение

С каждым годом все большее распространение получают задачи, связанные с обработкой матриц. Перемножение матриц — одна из стандартных и наиболее используемых операций над матрицами. Эту операцию можно выполнить, используя различные методы и алгоритмы. Одним из таких является алгоритм Винограда. Важную роль в решении таких задач играет скорость производимых вычислений. Многопоточность — один из способов улучшить данный показатель. В данной работе требуется на примере алгоритма Винограда изучить и применить на практике метод параллельных вычислений. Целью данной лабораторной работы является проведение сравнительного анализа скорости вычислений на одном потоке и множестве потоков. Для достижения поставленной задачи необходимо решить следующие задачи:

- реализовать ПО, включающее данный алгоритм;
- изучить распараллеливание вычислений и работу с потоками;
- реализовать распараллеливание вычислений;
- провести замеры времени выполнения алгоритмов на одном потоке и множестве потоков;

1 Аналитическая часть

В этом разделе содержатся теоретическое описание алгоритмов и указание области их применения.

1.1 Описание алгоритмов

Матрица — математический объект, эквивалентный двумерному массиву. Числа располагаются в матрице по строкам и столбцам. Если число столбцов в первой матрице совпадает с числом строк во второй, то эти матрицы можно перемножить. Произведением матриц $A(M * N)$ и $B(N * Q)$ является такая матрица $C(M * Q)$, что каждый ее элемент рассчитывается по формуле:

$$c_{ij} = \sum_{r=1}^{n} (a_{ir} * b_{rj})$$

Реализация умножения матриц стандартным алгоритмом имеет трудоемкость порядка $O(M * N * Q)$ [1]. Существуют алгоритмы умножения матриц, снижающие трудоемкость. Одним из таких является алгоритм Винограда. Данный алгоритм является одним из самых эффективных по времени алгоритмов умножения матриц. Особенность этого алгоритма заключается в том, что процесс умножения отдельных векторов преобразуется по формуле:

$$\begin{aligned} U * V = u_1 * v_1 + u_2 * v_2 + u_3 * v_3 + u_4 * v_4 = \\ = (u_1 + u_2) * (v_1 + v_2) + (u_3 + u_4) * (v_3 * v_4) - \\ - u_1 * u_2 - u_3 * u_4 - v_1 * v_2 - v_3 * v_4 \end{aligned}$$

При этом для каждых строки и столбца значения $u_i * u_{i+1}, v_i * v_{i+1}$ остаются неизменными, что позволяет вычислить их заранее. Оцененная трудоемкость такого алгоритма составляет порядка $O(N^{2.3727})$ для квадратных матриц [2]. Можно отметить, что вычисление каждый элемент искомой матрицы считается отдельно, и по этой причине можно увеличить скорость расчета результата за счёт распараллеливания вычислений.

1.2 Выводы

Умножение матриц — это очень важный и необходимый инструмент, который применяется в различных сферах человеческой жизни и вычисления которого можно ускорить, используя распараллеливание.

2 Конструкторская часть

В данном разделе будут приведены схема алгоритма и способы его распараллеливания.

2.1 Функциональная модель

На рисунке 1 представлена функциональная модель IDEF0 уровня 1.

Рисунок 1. Функциональная модель IDEF0 уровня 1

2.2 Разработка алгоритмов

Ниже представлена схема алгоритма умножения матриц по Винограду (рис. 2). Также приведена схема алгоритма умножения матриц по Винограду с возможностью распараллеливания (рис. 3).

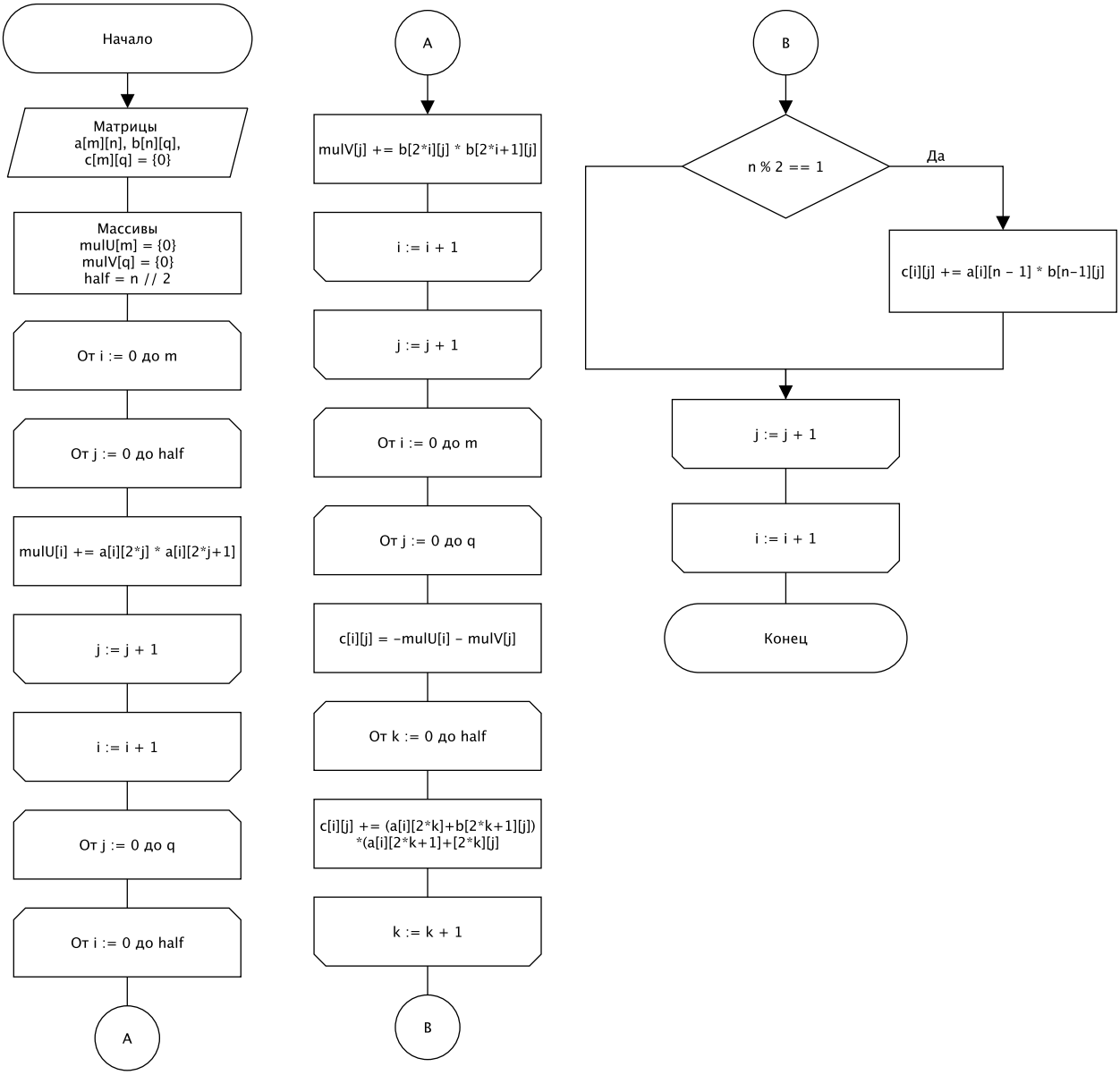

Рисунок 2: схема алгоритма Винограда вычисления произведения матриц

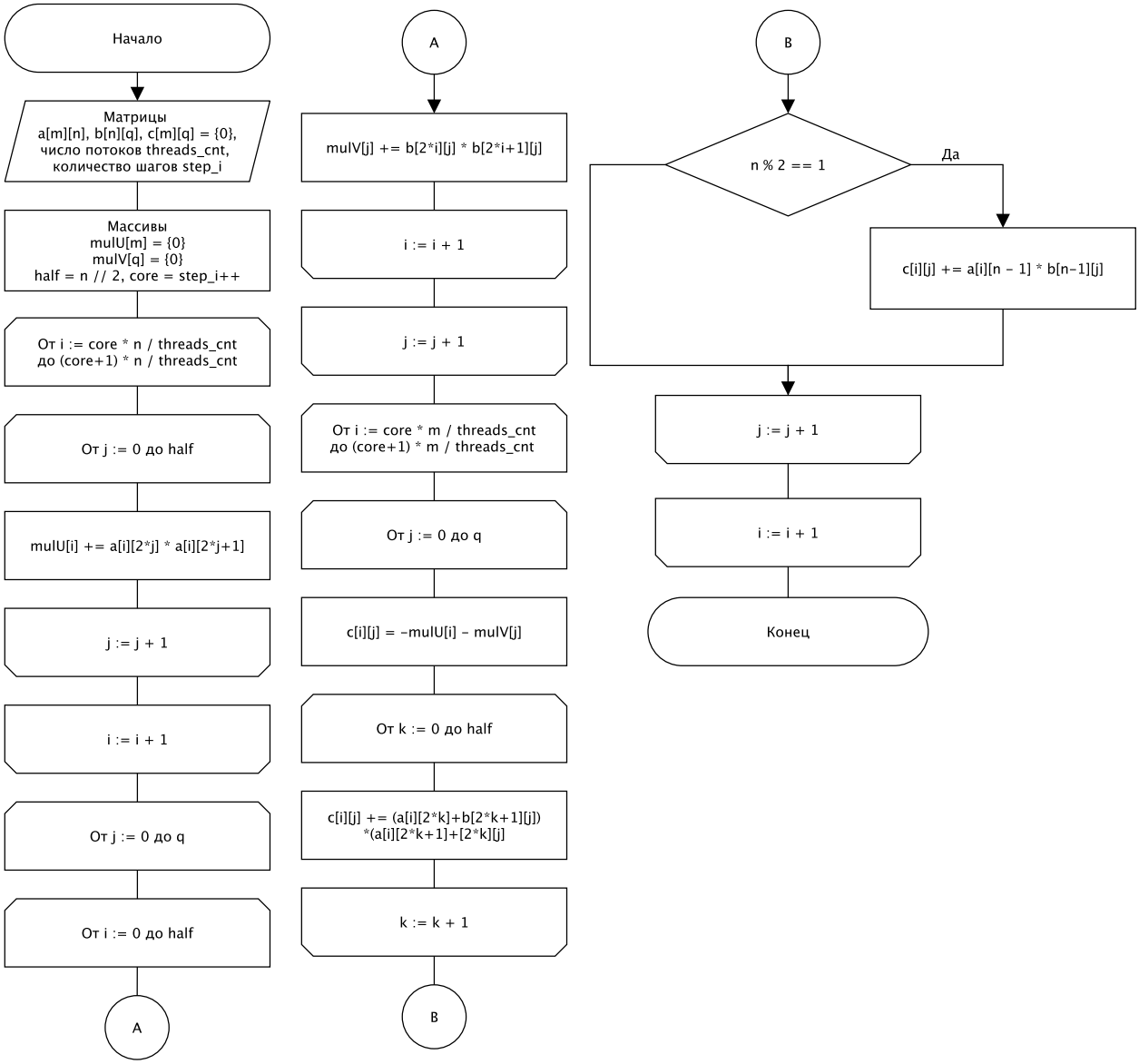

Рисунок 3: схема алгоритма Винограда вычисления произведения матриц с возможностью распараллеливания

Распараллеливание вычислений реализовано при помощи использования новых переменных threads_cnt, core, step_i, которые указывают на диапазон строк, отдаваемые под каждый поток и благодаря которым происходит разделение матрицы построчно между разными потоками. Это позволяет производить вычисления без использования мьютексов, так как каждый поток монополизирует рассчитанный диапазон в матрице. Эти переменные используются при расчете суммы произведений массивов mulU для строк, а также при расчете искомой матрицы во внешнем цикле, который отвечает за строки.

2.3 Выводы

Из-за возможности вычислять каждый элемент искомой матрицы отдельно, отдавая на каждый поток по диапазону строк из исходных матриц, удалось реализовать распараллеливание. После выполнения всех вычислений всеми потоками и соответственно расчета всех строк матрицы на выходе получается правильный результат.

3 Технологическая часть

В данном разделе представлены требования к разрабатываемому программному обеспечению, средства, использованные в процессе разработки для реализации поставленных задач, а также листинг кода программы.

3.1 Требования к программному обеспечению

ПО должно производить правильный расчет произведения матриц по Винограду и обеспечивать замер процессорного времени выполнения алгоритма при различном числе используемых потоков. Замеры производятся для квадратных матриц с размерностями от 100 до 1000 с шагом 100 и от 101 до 1001 с шагом 100.

3.2 Средства реализации

Для реализации поставленной задачи был использован язык программирования C++. Для измерения процессорного времени использовалась библиотека `chrono` [4]. Для распараллеливания вычислений была использована библиотека с классом нативных потоков `std::thread` [3].

3.3 Листинг кода

В приведенных ниже листингах представлены реализации алгоритма Винограда без возможности распараллеливания (листинг 1) и с возможностью распараллеливания (листинг 2).

Листинг 1: Алгоритм Винограда умножения матриц

```
void wino_algo(Matrix &c, Matrix& a, Matrix& b)
{
    int m = a.size(), n = b.size(), q = b[0].size();
    std::vector<int> mulU(m, 0);
    std::vector<int> mulV(q, 0);
    int half = n / 2;

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < half; j++) {
            mulU[i] += a[i][2 * j] * a[i][2 * j + 1];
        }
    }
    for (int j = 0; j < q; j++) {
```

```
        for (int i = 0; i < half; i++) {
            mulV[j] += b[2 * i][j] * b[2 * i + 1][j];
        }
    }
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < q; j++) {
            c[i][j] = - mulU[i] - mulV[j];
            for (int k = 0; k < half; k++) {
                c[i][j] += (a[i][2 * k] + b[2 * k + 1][j]) * (a[i]
[2 * k + 1] + b[2 * k][j]);
            }
            if (n % 2 == 1) {
                c[i][j] +=  a[i][n - 1] * b[n - 1][j];
            }
        }
    }
}
```

Листинг 2: Алгоритм Винограда умножения матриц с возможностью распараллеливания

```
void wino_algo_threads(Matrix &c, Matrix& a, Matrix& b, int threads_cnt)
{
    int core = step_i++;
    int m = a.size(), n = b.size(), q = b[0].size();
    std::vector<int> mulU(m, 0);
    std::vector<int> mulV(q, 0);
    int half = n / 2;

    for (int i = core*n/threads_cnt; i < (core+1)*n/threads_cnt; i++) {
        for (int j = 0; j < half; j++) {
            mulU[i] += a[i][2 * j] * a[i][2 * j + 1];
        }
    }
    for (int j = 0; j < q; j++) {
        for (int i = 0; i < half; i++) {
            mulV[j] += b[2 * i][j] * b[2 * i + 1][j];
        }
    }
    for (int i = core*m/threads_cnt; i < (core+1)*m/threads_cnt; i++) {
        for (int j = 0; j < q; j++) {
            c[i][j] = - mulU[i] - mulV[j];
            for (int k = 0; k < half; k++) {
                c[i][j] += (a[i][2 * k] + b[2 * k + 1][j]) * (a[i]
[2 * k + 1] + b[2 * k][j]);
            }
            if (n % 2 == 1) {
                c[i][j] +=  a[i][n - 1] * b[n - 1][j];
            }
        }
    }
}
```

3.4 Тестирование

Для тестирования программы были подготовлены следующие тесты в таблице 1.

Таблица 1

Тесты для алгоритма

Матрица №1	Матрица №2	Ожидаемый результат
2 3 4 5	1 0 0 1	2 3 4 5
1 1 1 2 2 2 3 3 3	1 1 1 2 2 2 3 3 3	6 6 6 12 12 12 18 18 18
1 1 1 2 2 2 3 3 3	1 2 3	6 12 18

3.5 Выводы

На основе схем алгоритмов, представленных в конструкторском разделе, в соответствии с указанными требованиями к реализации с использованием средств языка C++ было разработано программное обеспечение, содержащее реализации выбранного алгоритма. Для проверки правильности работы были заготовлены тесты.

4 Экспериментальная часть

В данном разделе будут приведены примеры работы программы, постановка эксперимента и сравнительный анализ алгоритмов на основе полученных данных.

4.1 Примеры работы

На приведенных ниже рисунках продемонстрирована работа алгоритмов.

```
└─ ./a.out
Input size of square matrix: 2
Input number of threads: 1
Time: 94

Output matrixes? (y/n) - y

First matrix:
  3   6
  9   1

Second matrix:
  2   9
  9   6

Result matrix:
 60  63
 27  87
```

Рисунок 4: пример работы программы с использованием одного потока.

```
└─ ./a.out
Input size of square matrix: 4
Input number of threads: 2
Time: 145

Output matrixes? (y/n) - y

First matrix:
  9   5   6   8
  0   0   0   8
  4   8   0   6
  2   0   5   7

Second matrix:
  8   1   6   5
  8   0   9   4
  3   1   2   2
  5   6   6   5

Result matrix:
170  63 159 117
 40  48  48  40
126  40 132  82
 66  49  64  55
```

Рисунок 5: пример работы программы с использованием двух потоков.

4.2 Результаты тестирования

Для тестирования были использованы тесты из таблицы 1. Результаты проведенного тестирования однопоточного и многопоточного алгоритмов приведены ниже.

Таблица 2

Результаты однопоточного алгоритма

Матрица №1	Матрица №2	Результат
2 3 4 5	1 0 0 1	2 3 4 5
1 1 1 2 2 2 3 3 3	1 1 1 2 2 2 3 3 3	6 6 6 12 12 12 18 18 18
1 1 1 2 2 2 3 3 3	1 2 3	6 12 18

Таблица 3

Результаты многопоточного алгоритма

Матрица №1	Матрица №2	Результат
2 3 4 5	1 0 0 1	2 3 4 5
1 1 1 2 2 2 3 3 3	1 1 1 2 2 2 3 3 3	6 6 6 12 12 12 18 18 18
1 1 1 2 2 2 3 3 3	1 2 3	6 12 18

Как видно в приведенных выше таблицах, все тесты пройдены успешно.

4.3 Постановка эксперимента по замеру времени

На рис. 6 представлены результаты замеров для варьирующихся размерностей матриц от 100x100 до 1000x1000 с шагом 100 при разном числе использованных потоков, на рис. 7 — для размерностей матриц от 101x101 до 1001x1001 с шагом 100. Один эксперимент ставился 5 раз, вычислялось среднее время работы.

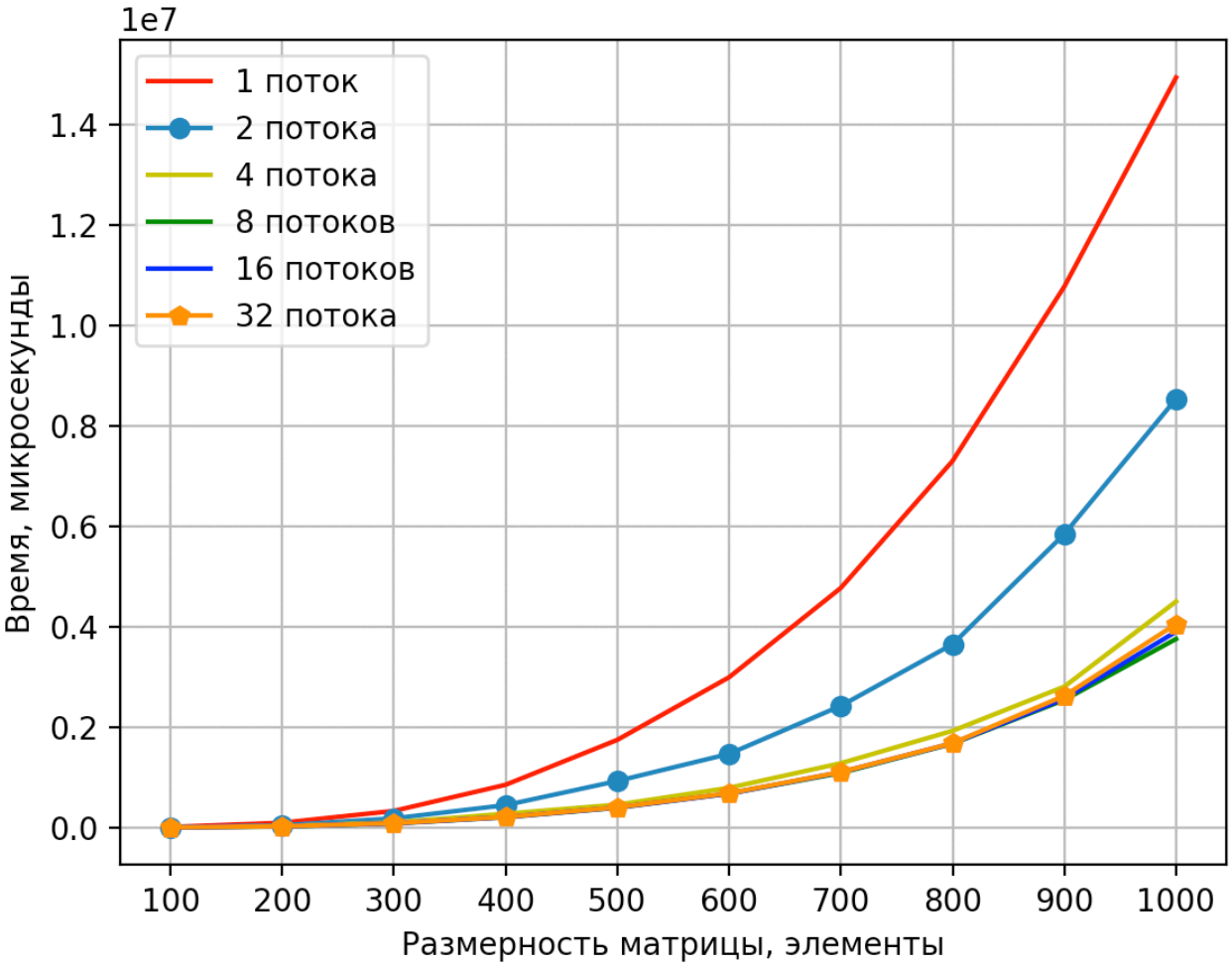

Рисунок 6: график зависимости времени работы алгоритма Винограда от четной размерности матриц при различном числе задействованных потоков.

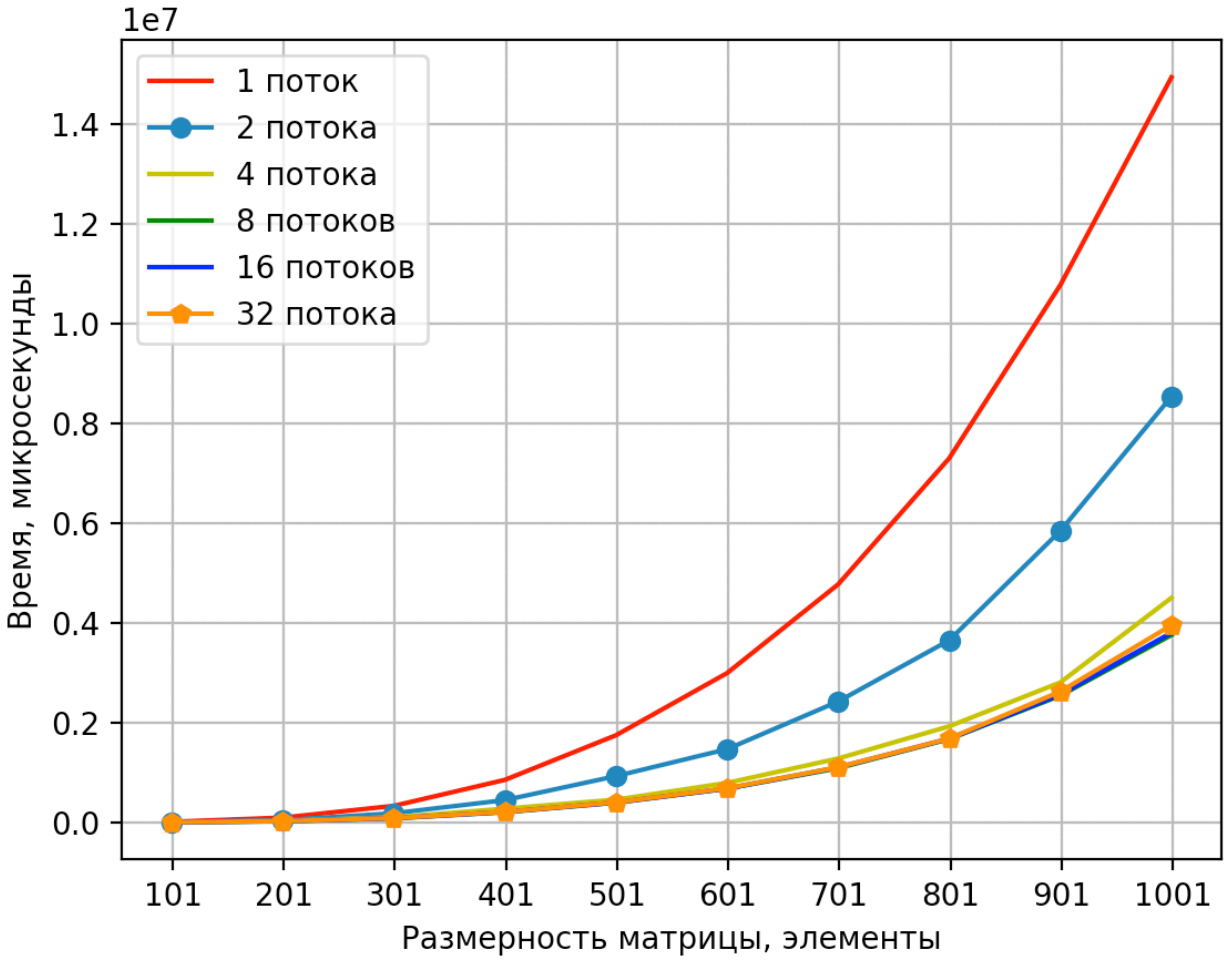

Рисунок 7: график зависимости времени работы алгоритма Винограда от нечетной размерности матриц при различном числе задействованных потоков.

4.4 Вывод

По графикам рис. 6, 7 видно, что на проведение вычислений на одном потоке затрачивается примерно в два раза больше времени, чем на нескольких потоках. Среди распараллеленных вычислений видно, что увеличение числа потоков дает небольшой прирост в производительность. Но стоит отметить, что при использовании числа потоков большего, чем логических ядер процессора, этот прирост уменьшается.

Заключение

В процессе выполнения лабораторной работы было проведено исследование работы алгоритма Винограда на нескольких потоках, и были сделаны следующие выводы:

- однопоточные вычисления проигрывают по времени многопоточным примерно в два раза;
- использование числа потоков, превышающее число логических ядер процессора, не дает выигрыша по времени, а также может работать медленнее.

Во время разработки программного обеспечения в соответствии с поставленными требованиями были получены практические навыки реализации алгоритма Винограда на нескольких потоках. При помощи разработанного программного обеспечения на материале замеров процессорного времени выполнения реализации на варьирующихся размерах матриц были экспериментально подтверждены различия во временной эффективности однопоточного и многопоточного алгоритмов вычисления произведения матриц по Винограду.

Список литературы

1. Корн Г., Корн Т. - «Алгебра матриц и матричное исчисление // Справочник по математике. — 4-е издание. — М: Наука, 1978. — С. 392—394»
2. Williams, Virginia - «Breaking the Coppersmith-Winograd barrier».
3. Документация по std::thread [Электронный ресурс]. - Режим http://www.cplusplus.com/reference/chrono/ Дата обращения: 19.10.2019
4. Документация по std::chrono [Электронный ресурс]. - Режим http://www.cplusplus.com/reference/chrono/ Дата обращения: 20.10.2019