



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 №4 по дисциплине "Анализ алгоритмов"

Тема Исследование многопоточности

Студент Малышев И. А.

Группа ИУ7-51Б

Оценка (баллы) \_\_\_\_\_

Преподаватель: Волкова Л. Л.

Москва — 2021 г.

# Содержание

|                                                      |           |
|------------------------------------------------------|-----------|
| <b>Введение</b>                                      | <b>2</b>  |
| <b>1 Аналитическая часть</b>                         | <b>3</b>  |
| 1.1 Стандартный алгоритм . . . . .                   | 3         |
| 1.2 Параллельный алгоритм . . . . .                  | 3         |
| 1.3 Вывод . . . . .                                  | 3         |
| <b>2 Конструкторская часть</b>                       | <b>4</b>  |
| 2.1 Схемы алгоритмов . . . . .                       | 4         |
| 2.2 Вывод . . . . .                                  | 5         |
| <b>3 Технологическая часть</b>                       | <b>6</b>  |
| 3.1 Средства реализации . . . . .                    | 6         |
| 3.2 Реализация алгоритмов . . . . .                  | 6         |
| 3.3 Тестирование . . . . .                           | 8         |
| 3.4 Вывод . . . . .                                  | 8         |
| <b>4 Исследовательская часть</b>                     | <b>9</b>  |
| 4.1 Технические характеристики . . . . .             | 9         |
| 4.2 Время выполнения реализаций алгоритмов . . . . . | 9         |
| 4.3 Вывод . . . . .                                  | 10        |
| <b>Заключение</b>                                    | <b>11</b> |
| <b>Список литературы</b>                             | <b>12</b> |

# Введение

Поток, или поток выполнения, – базовая упорядоченная последовательность инструкций, которые могут быть переданы или обработаны одним ядром процессора. Вычисления разбиваются на части, за каждую из которых отвечает отдельный поток. Если в системе больше одного процессора, то использование многопоточности фактически позволяет выполнять несколько действий одновременно.

Многие алгоритмы (операции над матрицами, алгоритмы сортировки и т. д.) допускают многопоточную реализацию. Поэтому **целью** данной работы является сравнить производительность обычной и многопоточной реализаций алгоритма вычисления среднего степенного для строк матрицы. Для достижения поставленной цели необходимо решить следующие задачи:

- изучить стандартный алгоритм вычисления среднего степенного для строк матрицы и найти способ его распараллеливания;
- разработать и привести схемы алгоритмов;
- выбрать средства реализации;
- реализовать однопоточный и многопоточный алгоритм вычисления среднего степенного для строк матрицы;
- протестировать реализованные алгоритмы;
- провести сравнительный анализ алгоритмов на основе экспериментальных данных: время выполнения в зависимости от количества потоков.

# 1 | Аналитическая часть

В этом разделе приведён обзор и анализ алгоритмов вычисления среднего степенного для строк матрицы.

## 1.1 Стандартный алгоритм

Пусть дана прямоугольная матрица

$$A_{nm} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{l1} & a_{l2} & \dots & a_{nm} \end{pmatrix}, \quad (1.1)$$

тогда вектор  $B$

$$B_n = (b_1 \quad b_2 \quad \dots \quad b_n), \quad (1.2)$$

где

$$b_i = \sqrt[d]{\sum_{j=1}^m a_{ij}^d}, \quad (j = \overline{1, m}) \quad (1.3)$$

будет называться средним степенным строки матрицы  $A$  [1].

## 1.2 Параллельный алгоритм

Можно заметить, что каждый элемент вектора  $B$  вычисляется независимо от других и матрица  $A$  не изменяется. Тогда для параллельного вычисления среднего степенного строк достаточно просто равным образом распределить строки матрицы  $A$  между потоками [2].

## 1.3 Вывод

В данном разделе было выявлено, что обычный алгоритм получения среднего степенного от ряда чисел в строке матрицы независимо вычисляет элементы вектора-результата, что дает большое количество возможностей для реализации параллельного варианта алгоритма.

## 2 | Конструкторская часть

В этом разделе приводятся схемы алгоритмов вычисления среднего степенного для строк матрицы, приведённых в аналитической части.

### 2.1 Схемы алгоритмов

На рисунках 2.1 и 2.2 показаны схемы алгоритмов сортировки пузырьком, выбором и быстрой сортировки соответственно.

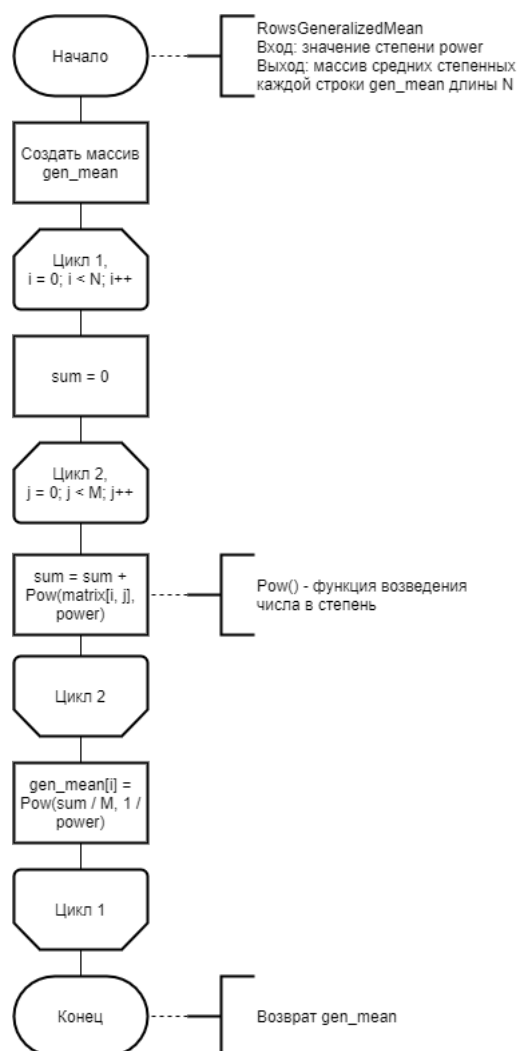


Рис. 2.1: Схема стандартного алгоритма

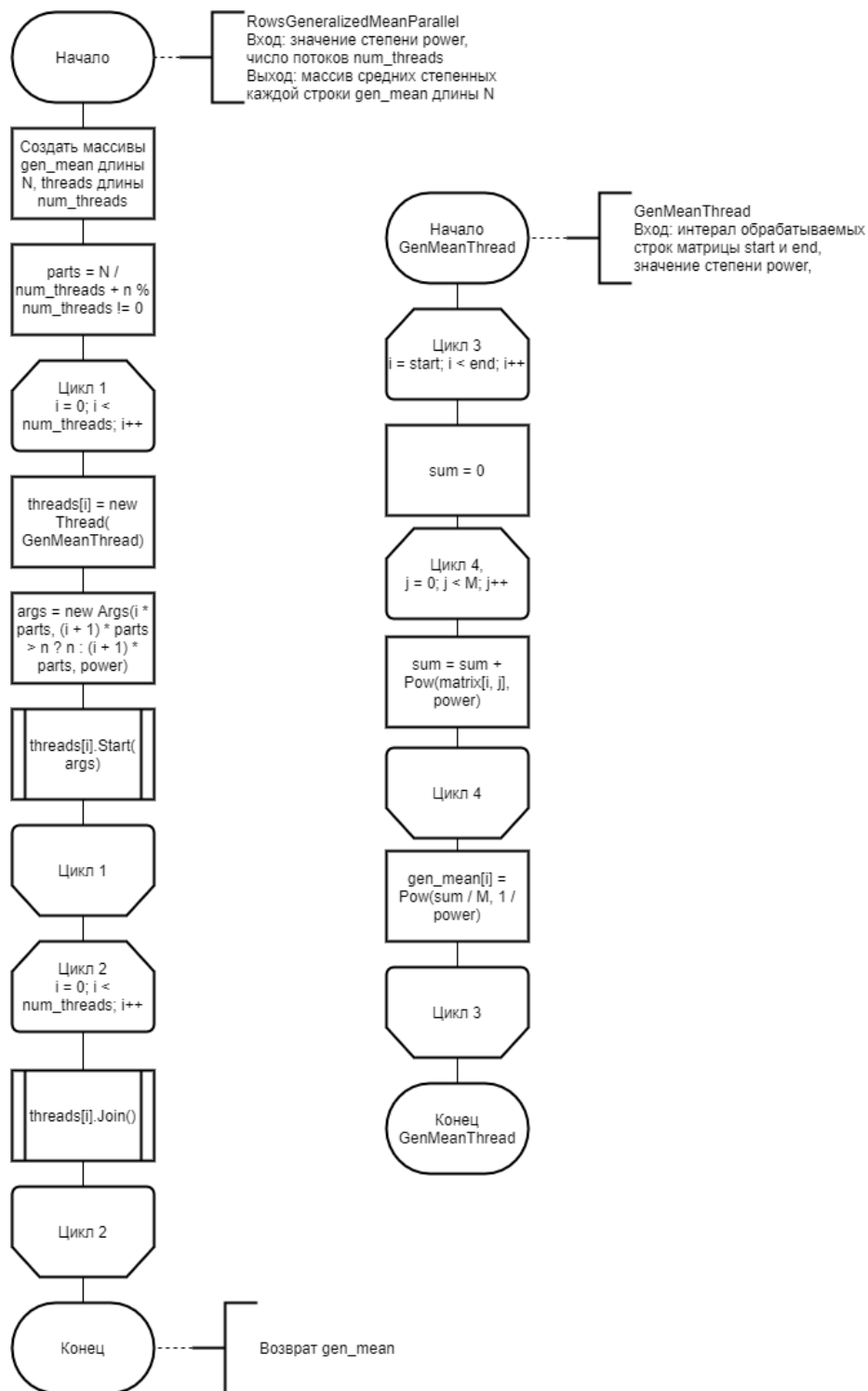


Рис. 2.2: Схема параллельной версии алгоритма

## 2.2 Вывод

На основе теоретических данных, полученных из аналитического раздела, были построены схемы алгоритмов вычисления среднего степенного для строк матрицы.

## 3 | Технологическая часть

В данном разделе приводится реализация алгоритмов, схемы которых были разработаны в конструкторской части. Кроме того, обосновывается выбор технологического стека и проводится тестирование реализованных алгоритмов.

### 3.1 Средства реализации

В качестве языка программирования был выбран C#, а среды разработки – Visual Studio, т. к. я знаком с данным языком и имею представление о тестировании программ в данном языке. Время работы алгоритмов было замерено с помощью библиотеки System.Diagnostics, класса Stopwatch, который имеет методы для расчёта процессорного времени [3].

### 3.2 Реализация алгоритмов

В листингах 3.1 и 3.2 приведена реализация стандартного и параллельного алгоритмов вычисления среднего степенного для строк матрицы.

Листинг 3.1: Стандартный метод вычисления среднего степенного для строк матрицы

```
1 public double[] RowsGenMean(double power)
2 {
3     double[] gen_means = new double[n];
4
5     for (int i = 0; i < n; i++)
6     {
7         double sum = 0;
8         for (int j = 0; j < m; j++)
9             sum += Math.Pow(matrix[i, j], power);
10
11         gen_means[i] = Math.Pow(sum / m, 1.0 / power);
12     }
13
14     return gen_means;
15 }
```

Листинг 3.2: Параллельный метод вычисления среднего степенного для строк матрицы

```

1 public double[] RowsGenMeanParallel(double power, int num_threads = 1)
2 {
3     Thread[] threads = new Thread[num_threads];
4     double[] gen_means = new double[n];
5
6     int parts = n / num_threads + (n % num_threads != 0 ? 1 : 0);
7
8     for (int i = 0; i < num_threads; i++)
9     {
10         threads[i] = new Thread(GenMeanThread);
11
12         Args args = new Args(i * parts, (i + 1) * parts > n ? n : (i + 1) *
13             parts, power);
14         threads[i].Start(args);
15     }
16
17     foreach (Thread t in threads)
18         t.Join();
19
20     return gen_means;
21
22     void GenMeanThread(object obj)
23     {
24         Args args = (Args)obj;
25         var (start, end) = (args.row_indx_start, args.row_indx_end);
26         double pow = args.power;
27
28         for (int i = start; i < end; i++)
29         {
30             double sum = 0;
31             for (int j = 0; j < m; j++)
32                 sum += Math.Pow(matrix[i, j], pow);
33
34             gen_means[i] = Math.Pow(sum / m, 1.0 / pow);
35         }
36     }
37 }

```



### 3.3 Тестирование

В таблице 3.1 приведены тесты для функций, реализующих стандартный и параллельный алгоритм вычисления среднего степенного для строк матрицы. Число потоков для многопоточной версии равно числу строк в матрице.

Таблица 3.1: Тестирование функций

| Матрица                                                             | Степень | Ожидаемый<br>результат | Фактический<br>результат<br>(стандартный) | Фактический<br>результат<br>(многопоточный) |
|---------------------------------------------------------------------|---------|------------------------|-------------------------------------------|---------------------------------------------|
| $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \end{pmatrix}$              | 2       | (2.1602 4.0825)        | (2.1602 4.0825)                           | (2.1602 4.0825)                             |
| $\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$             | 0.5     | (1.4571 3.4821 5.4886) | (1.4571 3.4821 5.4886)                    | (1.4571 3.4821 5.4886)                      |
| (2)                                                                 | 1       | (2)                    | (2)                                       | (2)                                         |
| $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ | 0       | (1 1 1)                | (1 1 1)                                   | (1 1 1)                                     |

Все тесты пройдены успешно.

### 3.4 Вывод

В данном разделе были реализованы стандартный и параллельный алгоритм вычисления среднего степенного для строк матрицы. Кроме того, реализации были успешно протестированы.

## 4 | Исследовательская часть

В данном разделе проводится сравнительный анализ реализованных алгоритмов по процессорному времени.

### 4.1 Технические характеристики

Ниже приведены технические характеристики устройства, на котором было проведено тестирование ПО:

- Операционная система: Windows 10 Home 64-bit [4];
- Оперативная память: 16 GB;
- Процессор: 4.0 GHz 4-ядерный процессор Intel Core i7-4790K [5].

### 4.2 Время выполнения реализаций алгоритмов

Для сравнительного анализа времени выполнения реализаций алгоритмов был проведен эксперимент. Для замеров были сформированы следующие данные:

- матрица со случайными числами размером 1024 на 1024;
- значение степени равно 2.

В таблице ?? представлены результаты замеров. Время измерялось 100 раз для каждого числа потоков, после усреднялось.

Таблица 4.1: Таблица времени выполнения (в тиках) алгоритмов при разном числе потоков

| Число потоков     | Время выполнения |
|-------------------|------------------|
| не распараллелено | 455 547          |
| 1                 | 549 188          |
| 2                 | 369 247          |
| 4                 | 221 466          |
| 8                 | 218 486          |
| 16                | 243 658          |
| 32                | 248 079          |

## 4.3 Вывод

На основе замеров процессорного времени было получено, что на конкретных данных параллельный алгоритм имеет наименьшее время работы при 8 потоках, что соответствует числу логических ядер компьютера, на котором проводилось тестирование. При числе потоков больше 8 увеличивается время работы алгоритма на тестируемом компьютере. Таким образом, рекомендуется использовать число потоков, равное числу логических ядер.

Также выявлено, что параллельная реализация при 8 потоках быстрее стандартной примерно в 2.1 раза. Хотя не распараллеленная реализация имеет меньшее время работы по сравнению с однопоточной, поскольку в однопоточной уходит время на создание потока.

# Заключение

В рамках данной лабораторной работы:

- был найден способ распараллеливания алгоритма вычисления среднего степенного для строк матрицы;
- были разработаны и приведены схемы алгоритмов;
- были реализованы однопоточный и многопоточный алгоритм вычисления среднего степенного для строк матрицы, а также протестированы;
- был проведён сравнительный анализ алгоритмов на основе экспериментальных данных: по времени выполнения в зависимости от количества потоков.

В результате исследования было выяснено, что параллельные реализации алгоритма вычисления среднего степенного для строк матрицы работают быстрее стандартной реализации. Наиболее эффективны данные алгоритмы при количестве потоков, совпадающем с количеством логических ядер компьютера. Так, на матрице размером 1024 на 1024 удалось улучшить время выполнения алгоритма в 2.1 раз (в сравнении со стандартной реализацией).

Поставленная цель была достигнута.

# Список литературы

1. Математическое просвещение (II). Выпуск 6. Математика, ее преподавание, приложения и история. Сборник. - М., ГИТТЛ, 1961. 376 с.
2. Kunle Olukotun. Chip Multiprocessor Architecture - Techniques to Improve Throughput and Latency. — Morgan and Claypool Publishers, 2007. — 154 p.
3. Свойство Process.UserProcessorTime [Электронный ресурс]. Режим доступа: <https://docs.microsoft.com/ru/dotnet/api/system.diagnostics.stopwatch?view=net-5.0>. Дата обращения: 26.10.2021
4. Windows 10 [Электронный ресурс]. Режим доступа: <https://www.microsoft.com/ru-ru/windows/get-windows-10>. Дата обращения: 26.10.2021
5. Процессор Intel Core I7-4790K [Электронный ресурс]. Режим доступа: <https://ark.intel.com/content/core-i7-4790k-processor-8m-cache-up-to-4-40-ghz.html>. Дата обращения: 26.10.2021