Министерство образования РФ

ФГАУВО Нижегородский государственный университет им. Н. И. Лобачевского

Институт Информационных Технологий Математики и Механики

Отчет

По лабораторной работе №1

**Сортировка Шелла**

Выполнил:

Студент группы

381506-3

Таламанов А. Е.

Проверил:

Доцент

кафедры МОСТ

Сысоев А. В.

Н. Новгород

2018

Оглавление

[Введение 3](#_Toc514615764)

[Постановка задачи 4](#_Toc514615765)

[Метод решения 5](#_Toc514615766)

[Последовательный алгоритм 5](#_Toc514615767)

[Параллельный алгоритм 7](#_Toc514615768)

[Generator 7](#_Toc514615769)

[Checker 7](#_Toc514615770)

[Результаты экспериментов 8](#_Toc514615771)

[Заключение 9](#_Toc514615772)

[Литература 10](#_Toc514615773)

# Введение

Раздел программирования **«Сортировка»** очень важен. Он касается свойств упорядоченных и неупорядоченных данных, а также помогает решить задачу разными способами. От порядка, в котором хранятся элементы в памяти компьютера, во многом зависит скорость выполнения и простота алгоритмов для их обработки.

Что такое сортировка? Когда элементы отсортированы, их проще найти, производить с ними различные операции. Легче определить пропущенные элементы. Легче найти общие элементы двух массивов. Сортировка влияет на скорость алгоритма, в котором нужно обратиться к определённому элементу массива.

Сортировка заслуживает серьезного изучения с точки зрения ее практического использования. Появление изощренных алгоритмов сортировки говорит о том, что она сама по себе интересна как объект исследования.

Данная работа содержит описание сортировки Шелла.

# Постановка задачи

Задача формулируется следующим образом: требуется разработать программу, сортирующую массив элементов с помощью сортировки Шелла с использованием методов параллельного программирования. Для этого требуется разработать последовательный и параллельный алгоритм данной сортировки. Затем сравнить время выполнения на нескольких потоках, определить ускорение.

Программа должна использовать библиотеки, предоставляющие API для параллельных вычислений: OpenMP и TBB.

**Выполнение задачи включает:**

1. Освоение темы (постановка задачи).
2. Изучение метода решения.
3. Разработку тестовой версии, включающей solver, generator, checker и набор тестов для проверки корректности параллельных версий.
4. Разработку OpenMP версии.
5. Разработку TBB версии.
6. Проведение экспериментов на 1, 2, 4 и 8 потоках.
7. Подготовку отчета с анализом результатов экспериментов.

**В программе необходимо:**

1. Выполнить проверку совпадения результатов последовательной и параллельной реализаций (checker).
2. Продемонстрировать корректность работы алгоритма на задаче/задачах малой размерности.
3. Обеспечить генерацию данных для задач произвольной размерности (generator).
4. Вывести время работы последовательного и параллельного алгоритмов. Ускорение параллельного алгоритма.

# Метод решения

Сортировка Шелла — алгоритм сортировки, являющийся усовершенствованным вариантом сортировки вставками. Идея метода Шелла состоит в сравнении элементов, стоящих не только рядом, но и на определённом расстоянии друг от друга. Иными словами — это сортировка вставками с предварительными «грубыми» проходами.

При сортировке Шелла сначала сравниваются и сортируются между собой значения, стоящие один от другого на некотором расстоянии *d*. После этого процедура повторяется для некоторых меньших значений *d*, а завершается сортировка Шелла упорядочиванием элементов при *d=1* (то есть обычной [сортировкой вставками](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%B2%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%BE%D0%B9)). Эффективность сортировки Шелла в определённых случаях обеспечивается тем, что элементы «быстрее» встают на свои места (в простых методах сортировки, например, [пузырьковой](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BC_%D0%BF%D1%83%D0%B7%D1%8B%D1%80%D1%8C%D0%BA%D0%B0), каждая перестановка двух элементов уменьшает количество [инверсий](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F_(%D0%BF%D0%B5%D1%80%D0%B5%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0)) в списке максимум на 1, а при сортировке Шелла это число может быть больше).

Невзирая на то, что сортировка Шелла во многих случаях медленнее, чем [быстрая сортировка](https://ru.wikipedia.org/wiki/%D0%91%D1%8B%D1%81%D1%82%D1%80%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0), она имеет ряд преимуществ:

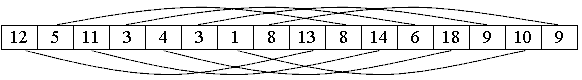
* отсутствие потребности в памяти под стек;
* отсутствие деградации при неудачных наборах данных — быстрая сортировка легко деградирует до O(n²), что хуже, чем худшее гарантированное время для сортировки Шелла.

## Последовательный алгоритм

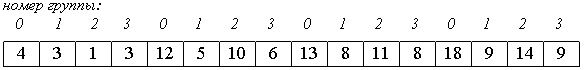
Рассмотрим алгоритм для целых положительных чисел массива a[0]...a[15].

http://algolist.manual.ru/sort/gif/11.gif

1. Вначале сортируем простыми вставками каждые 8 групп из 2-х элементов (a[0], a[8[), (a[1], a[9]), ... , (a[7], a[15]).

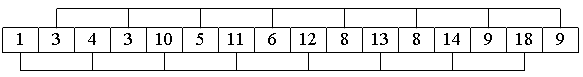


2. Потом сортируем каждую из четырех групп по 4 элемента (a[0], a[4], a[8], a[12]), ..., (a[3], a[7], a[11], a[15]).



В нулевой группе будут элементы 4, 12, 13, 18, в первой - 3, 5, 8, 9 и т.п.

3. Далее сортируем 2 группы по 8 элементов, начиная с (a[0], a[2], a[4], a[6], a[8], a[10], a[12], a[14]).



4. В конце сортируем вставками все 16 элементов.

http://algolist.manual.ru/sort/gif/15.gif

Очевидно, лишь последняя сортировка необходима, чтобы расположить все элементы по своим местам.

Все предыдущие сортировки продвигают элементы максимально близко к соответствующим позициям, так что в последней стадии число перемещений будет весьма невелико. Последовательность и так почти отсортирована. Ускорение подтверждено многочисленными исследованиями и на практике оказывается довольно существенным.

Единственной характеристикой сортировки Шелла является *приращение* - расстояние между сортируемыми элементами, в зависимости от прохода. В конце приращение всегда равно единице - метод завершается обычной сортировкой вставками, но именно последовательность приращений определяет рост эффективности.

Использованный в примере набор ..., 8, 4, 2, 1 - неплохой выбор, особенно, когда количество элементов - степень двойки. Однако гораздо лучший вариант предложил Р.Седжвик. Его последовательность имеет вид

http://algolist.manual.ru/sort/gif/14_1.gif

При использовании таких приращений среднее количество операций: O(n7/6), в худшем случае - порядка O(n4/3).

Обратим внимание на то, что последовательность вычисляется в порядке, противоположном используемому: inc[0] = 1, inc[1] = 5, ... Формула дает сначала меньшие числа, затем все большие и большие, в то время как расстояние между сортируемыми элементами, наоборот, должно уменьшаться. Поэтому массив приращений inc вычисляется перед запуском собственно сортировки до максимального расстояния между элементами, которое будет первым шагом в сортировке Шелла. Потом его значения используются в обратном порядке. При использовании формулы Седжвика следует остановиться на значении inc[s-1], если 3\*inc[s] > size.

## Параллельный алгоритм

1. Каждый поток получает на обработку часть массива и выполняет последовательную сортировку.
2. Отсортированные части сливаются между собой параллельно.

## Generator

Генератор создает массив и записывает его в бинарный файл.

## Checker

Берется массив из бинарного файла, сортируется и проверяется на отсортированность, используя std::is\_sorted().

# Результаты экспериментов

Проведя серию экспериментов, можно заключить, что параллельная сортировка работает гораздо быстрее, чем последовательная на всех данных. Наилучшее время работы достигается при распараллеливании на 8 потоков с помощью TBB.

# Заключение

В результате работы была написана программа, в которой реализованы последовательная и параллельные версии сортировки Шелла. Было замерено время их работы.

Следует заметить, что параллельная версия, с использованием как OpenMP, так и TBB, работает быстрее последовательной версии во всех случаях. Но TBB версия работает немного быстрее OpenMP версии.

# Литература

1. Сиднев А.А., Сысоев А.В., Мееров И.Б. “Лабораторная работа №3 Сортировки”
2. Кормен Т., Лейзерсон Ч., Ривест Р. "Алгоритмы. Построение и анализ"