Федеральное государственное автономное образовательное учреждение профессионального образования   
«Нижегородский государственный университет им. Н.И.Лобачевского»

Институт информационных технологий математики и механики

Отчет по лабораторной работе

«Сортировка Хоара с четно-нечетным слиянием Бэтчера»

**Выполнил:**

Студент ИИТММ

группы 381506-3

Левкин.А.

**Проверил:**

Кандидат технических наук,

доцент каф. МОСТ, ИИТММ

Сысоев А.В.

Нижний Новгород

2018

Оглавление

[Постановка задачи 3](#_Toc514676661)

[Программная реализация 4](#_Toc514676662)

[Последовательная версия 4](#_Toc514676663)

[Параллельная версия 4](#_Toc514676664)

[Руководство пользователя 5](#_Toc514676665)

[Последовательная версия 5](#_Toc514676666)

[Параллельная версия 6](#_Toc514676667)

[Вычислительные эксперименты 6](#_Toc514676668)

[OpenMP 7](#_Toc514676669)

[TBB 8](#_Toc514676670)

[Заключение 9](#_Toc514676671)

# Постановка задачи

Сортировка Хоара – один из самых известных быстрых алгоритмов сортировки массивов на сегодняшний день. Его сложность в лучшем случае - среднем и в худшем случае .

Стоит отметить достоинства метода:

1. Быстрота
2. Простота реализации
3. Допускает удобное распараллеливание

Однако существуют и недостатки:

1. В худшем случае сложность
2. Реализация без дополнений (с двумя рекурсивными вызовами) может повлечь за собой ошибку переполнения стека.
3. Алгоритм неустойчив.

В данной лабораторной работе рассматривается сортировка Хоара с четно-нечетным слиянием Бэтчера. Четно-нечетное слияние Бэтчера заключается в том, что два упорядоченных массива, которые необходимо слить, разделяются на четные и нечетные элементы. Далее выполняется слияние четных и нечетных элементов массивов. Чтобы массив стал окончательно отсортированным, достаточно сравнить пары элементов, стоящие на нечетной и четной позициях. Первый и последний элементы массива проверять не надо, т.к. они являются минимальным и максимальным элементов массивов. Стоит отметить, что алгоритм с этой модификацией всё так же легко распараллелить, что я продемонстрирую ниже.

Параллельные версии реализованы с помощью библиотек OpenMP и Intel TBB.

# Программная реализация

## Последовательная версия

Для реализации требуемого алгоритма написаны и используются следующие функции:

* template<typename T> void swapMaxMin(T& x1, T& x2) – функция, меняющая элементы массива местами, если x1 > x2
* template<typename Tmass, typename Tint> void split(Tmass\* mass, Tint iStart, Tint iFinish, Tint halfSizeMass) – функция, осуществляющая разделение на чётные и нечётные элементы
* template<typename Tmass, typename Tint> void unsplit(Tmass\* mass, Tint iStart, Tint iFinish, Tint halfSizeMass) – функция, которая
* template<typename Tmass, typename Tint> void mergeMass(Tmass\* mass, Tint iStart, Tint iFinish) – функция, осуществляющая слияние массива
* template<typename Tmass, typename Tint> void qsortBatcher(Tmass\* mass, Tint iStart, Tint iFinish) – функция сортировки
* void qsortBatcher(int\* mass, int size) – функция-заглушка для удобного вызова предыдущей функции

Чтобы отсортировать массив, достаточно вызвать последнюю функцию.

## Параллельная версия

*Схема распараллеливания*:

1. Делим массив на части по числу потоков
2. Сортируем каждую часть на потоках
3. Производим слияние

# Руководство пользователя

## Последовательная версия

1. Запускаем командную строку Windows от имени администратора.

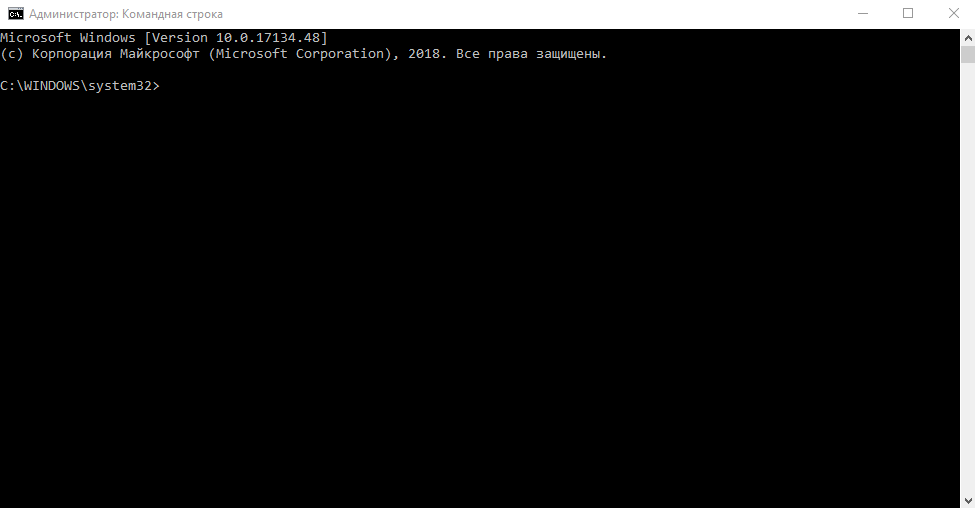
Нажмите кнопку «Поиск», затем введите “cmd” в найденном выберите «Командная строка», нажмите правой кнопкой мыши и выберите «Запустить от имени администратора».

Рис. 1 Запуск командной строки

1. Перейдите в каталог с исполняемым файлом программы. Введите в командной строке “cd” и путь до файла.

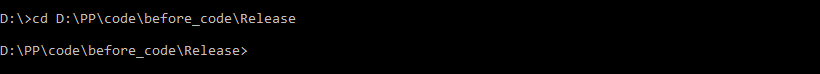


Рис. 2 Каталог с программой

1. Введите имя программы и укажите количество тестов, которые необходимо провести, после чего нажмите «Enter»

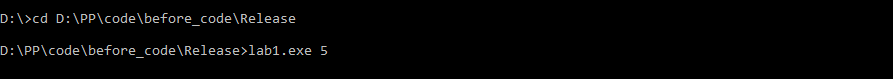


Рис. 3 Запуск программы

Результатом работы программы является файл ответов, в котором содержится отсортированный массив и время, которое потребовалось для сортировки.

## Параллельная версия

Для запуска параллельной версии, необходимо проделать все действия из вышестоящего руководства за исключением пункта 3. К аргументу командной строки, отвечающему за количество проведённых тестов, также добавляется второй аргумент – количество потоков, на котором будет запускаться программа.

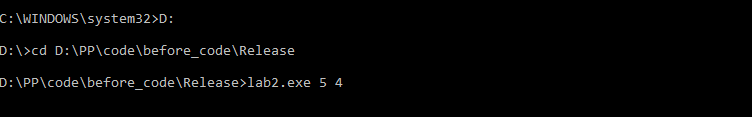


Рис. 4 Запуск параллельной программы

# Вычислительные эксперименты

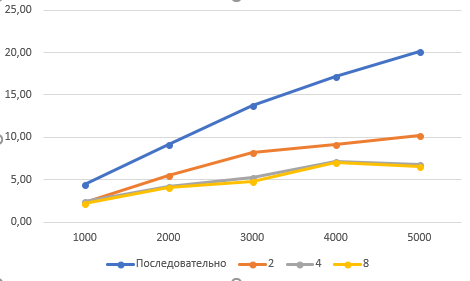
Для выяснения преимуществ распараллеленного алгоритма необходимо провести вычислительные эксперименты. Эксперимент будет проводиться на массивах больших размеров, иначе затраты на выполнение параллелизации будут перекрывать возможное ускорение.

Эксперименты проводятся на компьютере со следующими тех. характеристиками:

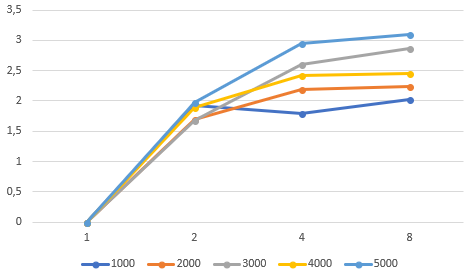
* Процессор: Intel Core i7 4510U (4 ядра)
* Оперативная память: 8GB
* ОС: Windows 10

Ниже будут представлены графики зависимостей времени выполнения на разном количестве потоков от количества элементов массива и ускорения от количества потоков.

## OpenMP

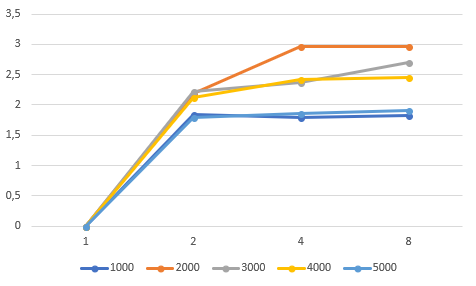


Граф. 1 Зависимость времени выполнения программы от количества элементов массива.

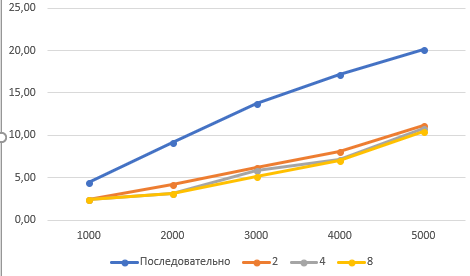


Граф. 2 Ускорение параллельного алгоритма на различном количестве потоков и точек

## TBB



Граф. 3 Ускорение параллельного алгоритма на различном количестве потоков и точек



Граф. 4 Зависимость времени выполнения программы от количества элементов массива.

# Заключение

Итого реализован алгоритм сортировки Хоара с четно-нечетным слиянием Бэтчера в трёх версиях: последовательной, параллельной OpenMP и параллельной TBB.

Проведены вычислительные эксперименты отражающие преимущества параллельных версий над последовательной.