Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Сортировки»**

**Выполнил**:

студент группы 3821Б1ПМ2

Деревянкин К.Е.

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2021

**Содержание**

[Постановка задачи 3](#_Toc26962562)

[Метод решения 4](#_Toc26962563)

[Руководство пользователя 7](#_Toc26962564)

[Описание программной реализации 8](#_Toc26962565)

[Подтверждение корректности 9](#_Toc26962566)

[Результаты экспериментов 10](#_Toc26962567)

[Заключение 19](#_Toc26962568)

[Приложение 20](#_Toc26962569)

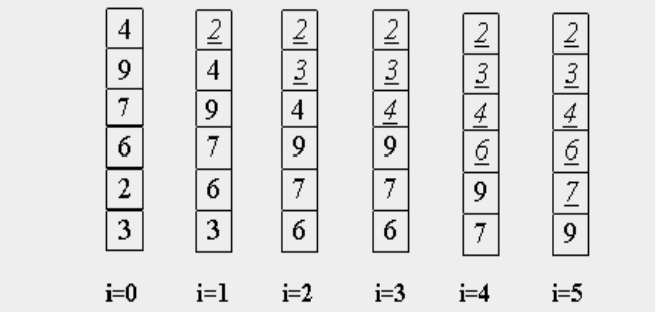
# Постановка задачи

Целью лабораторной работы является реализация массива по возрастанию с типом данных double. Для выполнения этого задания используется четыре программы: сортировка пузырьком, сортировка расческой, сортировка слиянием, поразрядная сортировка. Для каждого алгоритма нужно сосчитать количество сравнений и перестановок, а также для подтверждения сложности необходимо провести эксперименты, описать способ проведения экспериментов и сделать вывод по полученным данным

# Метод решения

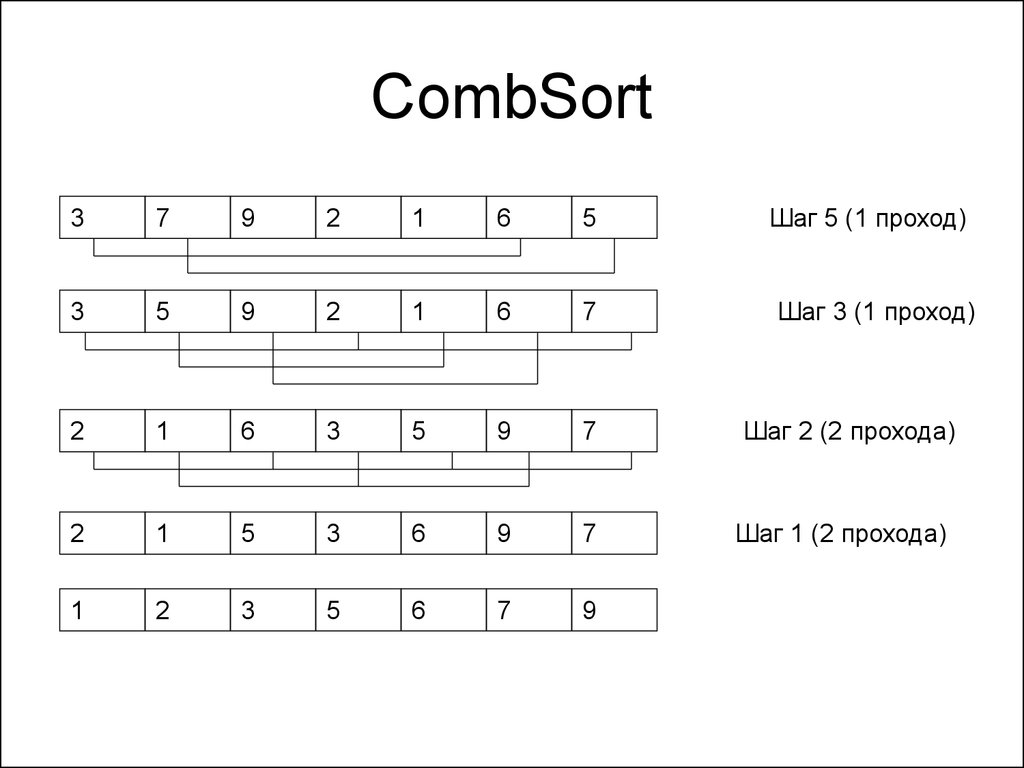
## Сортировка пузырьком.

Бессмертная классика жанра. Принцип действий прост: обходим массив от начала до конца, попутно меняя местами неотсортированные соседние элементы. В результате первого прохода на последнее место «всплывёт» максимальный элемент. Теперь снова обходим неотсортированную часть массива (от первого элемента до предпоследнего) и меняем по пути неотсортированных соседей. Второй по величине элемент окажется на предпоследнем месте. Продолжая в том же духе, будем обходить всё уменьшающуюся неотсортированную часть массива, запихивая найденные максимумы в конец. Таким образом, теоретическая сложность данной сортировки в лучшем случае - O(n), в худшем - O(), и в среднем - O().



## Сортировка расческой.

Основная идея «расчёски» в том, чтобы первоначально брать достаточно большое расстояние между сравниваемыми элементами и по мере упорядочивания массива сужать это расстояние вплоть до минимального. Таким образом мы как бы причёсываем массив, постепенно разглаживая на всё более аккуратные пряди. Первоначальный разрыв между сравниваемыми элементами лучше брать с учётом специальной величины называемой**фактором уменьшения**, оптимальное значение которой равно примерно 1,2473309. Сначала расстояние между элементами равно размеру массива разделённого на **фактор уменьшения** (результат, естественно, округляется до ближайшего целого). Затем, пройдя массив с этим шагом, мы снова делим шаг на **фактор уменьшения** и проходим по списку вновь. Так продолжается до тех пор, пока разность индексов не достигнет единицы. В этом случае массив досортировывается обычным пузырьком. Таким образом, теоретическая сложность данной сортировки в лучшем случае - O(n \* ), в худшем - O(), и в среднем - O(), где p – количество приращений.

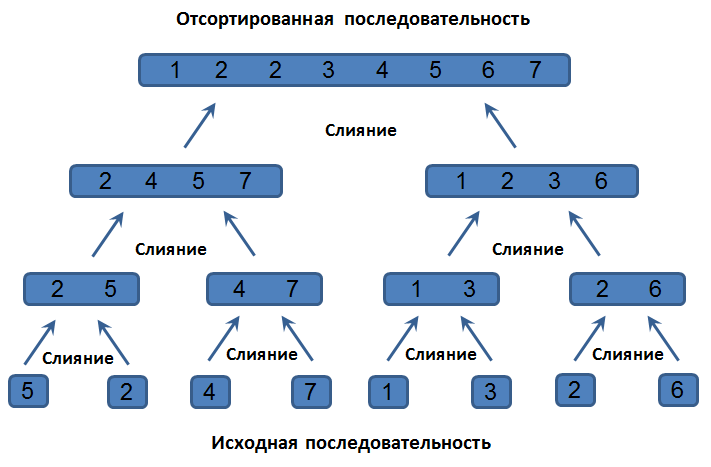


## Сортировка слиянием.

Эта сортировка — хороший пример использования принципа «разделяй и властвуй». Сначала задача разбивается на несколько подзадач меньшего размера. Затем эти задачи решаются с помощью рекурсивного вызова или непосредственно, если их размер достаточно мал. Наконец, их решения комбинируются, и получается решение исходной задачи.

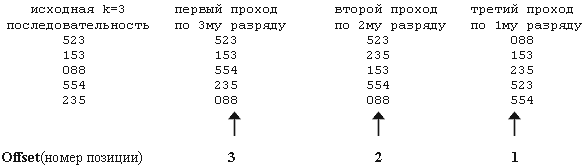
Массив рекурсивно разбивается пополам, и каждая из половин делиться до тех пор, пока размер очередного подмассива не станет равным единице; далее выполняется операция алгоритма, называемая слиянием. Два единичных массива сливаются в общий результирующий массив, при этом из каждого выбирается меньший элемент (сортировка по возрастанию) и записывается в свободную левую ячейку результирующего массива. После чего из двух результирующих массивов собирается третий общий отсортированный массив, и так далее. В случае если один из массивов закончиться, элементы другого дописываются в собираемый массив; в конце операции слияния, элементы перезаписываются из результирующего массива в исходный.

Таким образом, теоретическая сложность данной сортировки в любом случае - O(n \* ).



## Поразрядная сортировка.

Поразрядная сортировка является улучшенной сортировкой подсчётом, которая позволяет сортировать числа большего диапазона, используя другую устойчивую сортировку. Идея сортировки состоит в следующем: отсортировать числа по младшему разряду; потом устойчивой сортировкой отсортировать по второму, третьему, и так далее до старшего разряда: в качестве устойчивой сортировки можно выбрать сортировку подсчетом, в виду малого работы времени. Таким образом, теоретическая сложность данной сортировки в любом случае - O(n\*k), где k – это максимальное количество разрядов.



# Руководство пользователя

Все сортировки выводят исходный массив со случайными числами. По умолчанию количество чисел составляет 5, но его можно изменить. Дальше выводится отсортированный массив (числа расположены по возрастанию), количество перестановок и сравнений.

# Описание программной реализации

В самом начале программы будут представлены две переменные: srav и per, которые считают количество сравнений и перестановок.

void rands (double array[], int b) – функция, которая создает массив из случайных чисел длинной n.

void sont (double array[], int i) – функция, которая копирует массив из случайных чисел.

## Сортировка пузырьком.

void bubblesort (double mas[]) – на вход функция принимает массив для сортировки. Данная функция сортирует массив, выталкивая легкие пузырьки на поверхность.

## Сортировка расческой.

void combsort (double mas[]) – на вход функция принимает массив для сортировки. Данная функция сортирует массив, первоначально берет достаточно большое расстояние между сравниваемыми элементами и по мере упорядочивания массива сужает это расстояние вплоть до минимального.

## Сортировка слиянием.

int merge (double a[], int first, int middle, int second) – на вход функция принимает массив для сортировки. Данная функция при помощи mergeSort делит массив на подмассивы а потом собирает отсортированные под-массивы в отсортированный массив.

void mergeSort (double a[], int l, int r) – на вход функция принимает массив для сортировки, левый крайний индекс элемента и правый крайний индекс элемента.

## Поразрядная сортировка.

int\* createCounters(double\* data, int\* counters, long N) – на вход принимает указатель на первый элемент массива, создается еще один массив, когда функция подсчитывает сколько раз какое число значение каждого байта числа встретилось, то записывается все в этот массив, также на вход принимается длина этого массива.

void radixPass(short Offset, double\* source, double\* dest, int\* count, long N) - функция принимает позицию ,начиная с которой нужно вставлять число в выходной массив, длину сортируемого массива, указатель на первый элемент исходного массива, указатель на первый элемент выходного массива , указатель на начало позиций сортируемого разряд из массива, созданного функцией createCounters(). Функция выполняет поразрядную сортировку.

void signedRadixLastPass(double\* source, double\* dest, long N) - функция принимает на вход указатель на первый элемент входного массива , указатель на первый элемент выходного массива, необходимого для функции указанной выше, их длину и указатель на массив в который запишем результат. Данная функция сортирует отрицательные числа.

void radixSort(double\* in, double\* out, int\* counters, long N) - функция принимает позицию ,начиная с которой нужно вставлять число в выходной массив, длину сортируемого массива, указатель на первый элемент исходного массива, указатель на первый элемент выходного массива , указатель на начало позиций сортируемого разряд из массива, созданного функцией createCounters() . Функция выполняет поразрядную сортировку.

# Подтверждение корректности

Для подтверждения корректности в программе я использовал сортировку пузырьком, потому что она понятная и проста в реализации.

# Результаты экспериментов

Во всех сортировках использовал массив из случайных чисел.

## Сортировка пузырьком.

Сложность данной сортировки является О(.

В сортировке пузырьком длина массива менялась с 1 до 4000 с шагом 1.

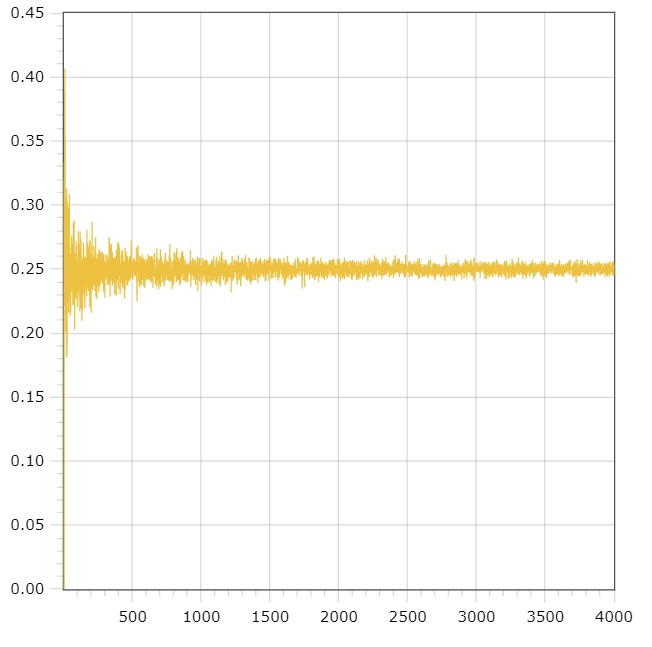
На первом графике можно видеть соотношение длины массива (по x) к количеству

перестановок деленное на сложность (по y). Видно, что значения сходятся к 0,25. Так как

диапазон значений с ростом количества элементов уменьшается и значение становится все

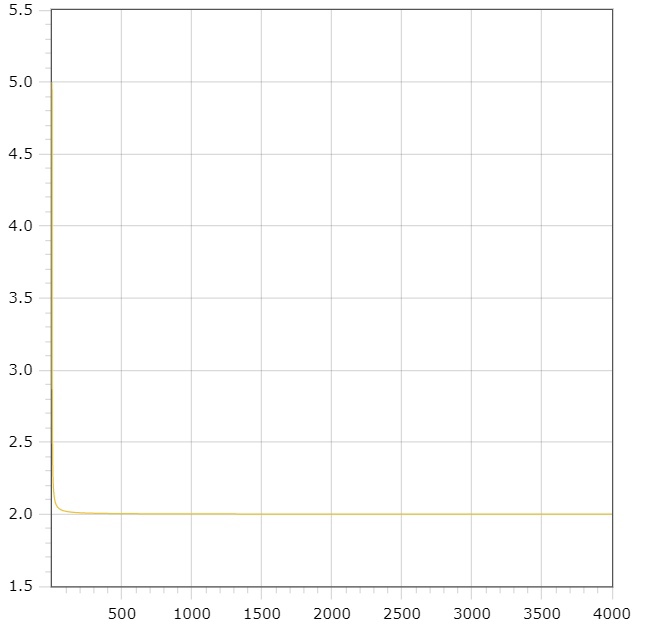
ближе к некоторому числу, то это является экспериментальным подтверждением указанной

сложности по сравнению для данной сортировки.

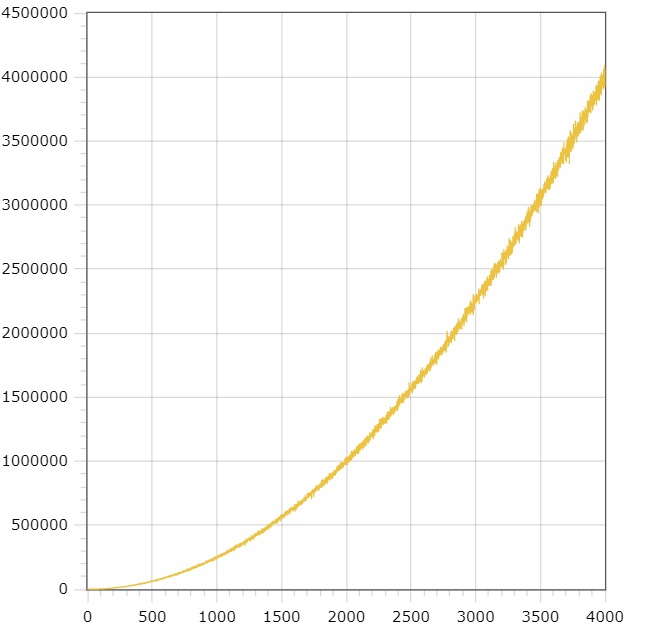


На втором графике можно видеть соотношение длины массива (по x) к количеству

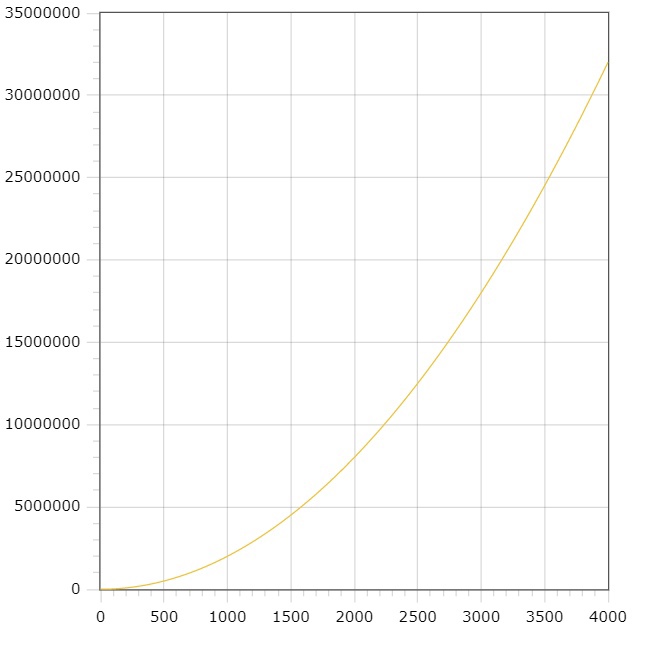
сравнений деленное на сложность (по y). Видно, что значения сходятся к 2. Так как график сходится к некоторому значению, то это является экспериментальным подтверждением указанной сложности по сравнениям для данной сортировки.



Третий график показывает отношение количества перестановок (по y) к длине массива (по x).



Четвёртый график показывает отношение количества сравнений (по y) к длине массива (по x).



## Сортировка расчёской.

Сложность данной сортировки является в худшем - O(), и в среднем - O(), где p – количество приращений.

В сортировке пузырьком длина массива менялась с 1 до 4000 с шагом 1.

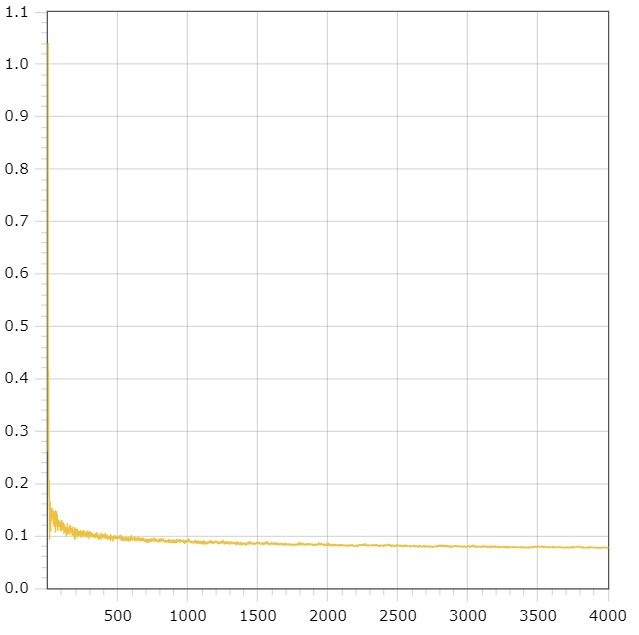
На первом графике можно видеть соотношение длины массива (по x) к количеству

перестановок деленное на сложность (по y). Видно, что значения сходятся к 0,8. Так как

диапазон значений с ростом количества элементов уменьшается и значение становится все

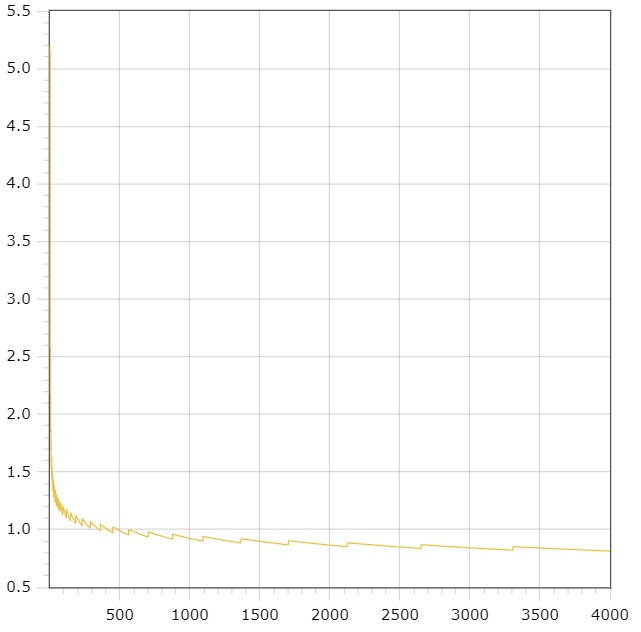
ближе к некоторому числу, то это является экспериментальным подтверждением указанной

сложности по сравнению для данной сортировки.

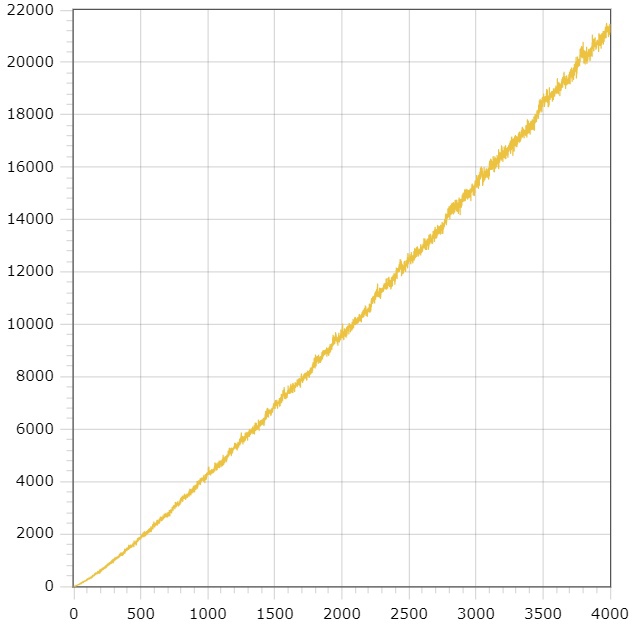


На втором графике можно видеть соотношение длины массива (по x) к количеству

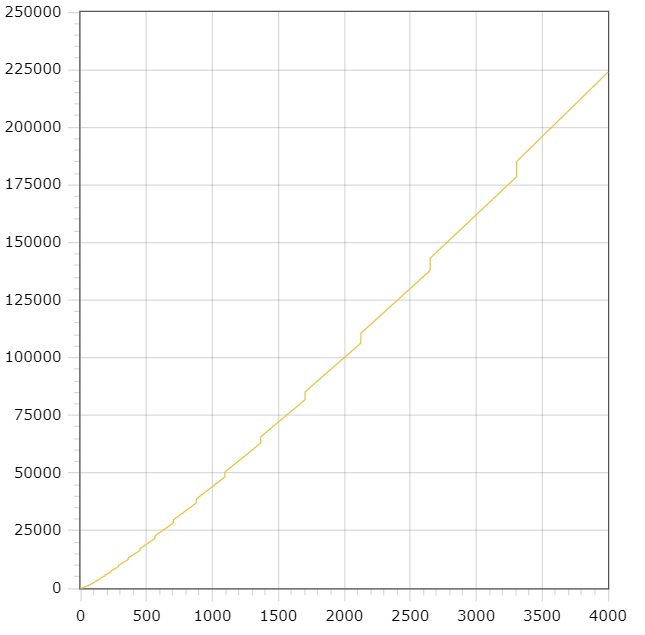
сравнений деленное на сложность (по y). Видно, что значения сходятся к 0,8. Так как график сходится к некоторому значению, то это является экспериментальным подтверждением указанной сложности по сравнениям для данной сортировки.



Третий график показывает отношение количества перестановок (по y) к длине массива (по x).



Четвёртый график показывает отношение количества сравнений (по y) к длине массива (по x).



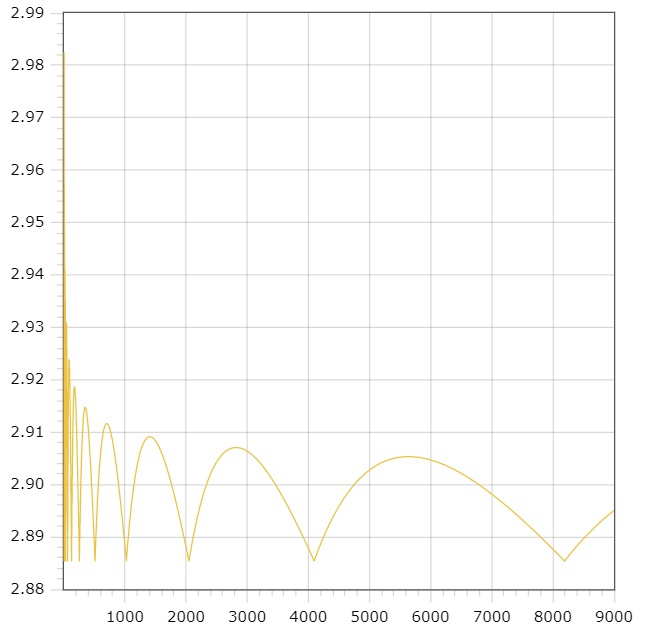
## Сортировка слиянием.

Сложность данной сортировки является O(n\*.

В сортировке пузырьком длина массива менялась с 1 до 9000 с шагом 1.

На первом графике можно видеть соотношение длины массива (по x) к количеству

перестановок деленное на сложность (по y). По графику видно, что с некоторого момента теоретическая сложность верна с точностью до константы.

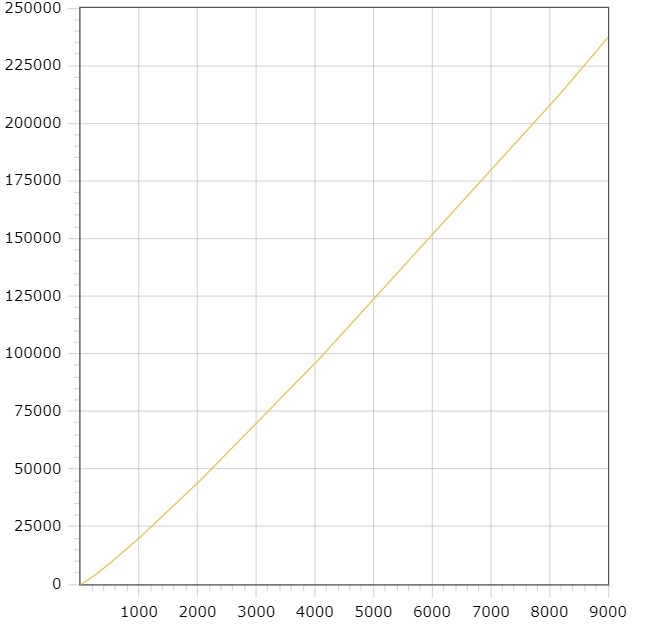


На втором графике можно видеть соотношение длины массива (по x) к количеству

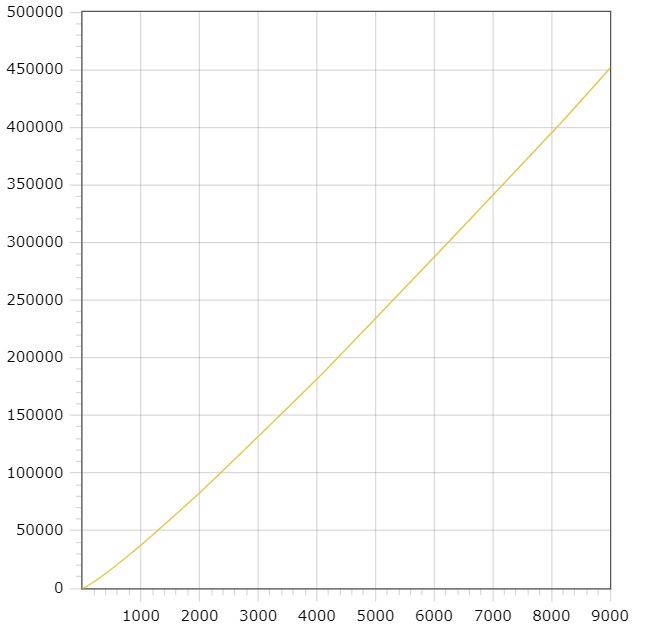
сравнений деленное на сложность (по y). По графику видео, что с некоторого момента теоретическая сложность верна с точностью до константы.

## https://sun9-59.userapi.com/impg/ePONYVSiBZJhq2uR8CsvjQ6KkQFicsXYVVo0jw/HRQUmwG4664.jpg?size=645x631&quality=96&sign=cf1b77007ed345e07c93e7d977626b91&type=album

Третий график показывает отношение количества перестановок (по y) к длине массива (по x).



Четвёртый график показывает отношение количества сравнений (по y) к длине массива (по x).



## Поразрядная сортировка.

Сложность данной сортировки является O(4 \* (n+256) + n)

В сортировке пузырьком длина массива менялась с 1 до 11000 с шагом 1.

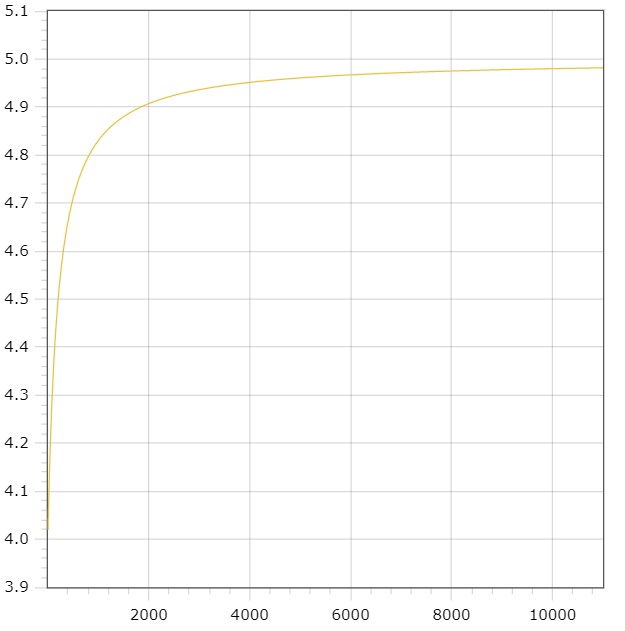
На первом графике можно видеть соотношение длины массива (по x) к количеству

перестановок деленное на сложность (по y). Видно, что значения сходятся к 4,98. Так как

диапазон значений с ростом количества элементов уменьшается и значение становится все

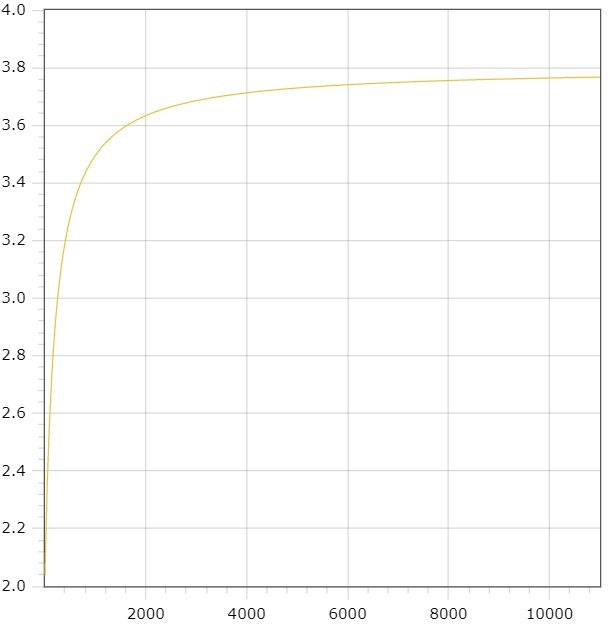
ближе к некоторому числу, то это является экспериментальным подтверждением указанной

сложности по сравнению для данной сортировки.

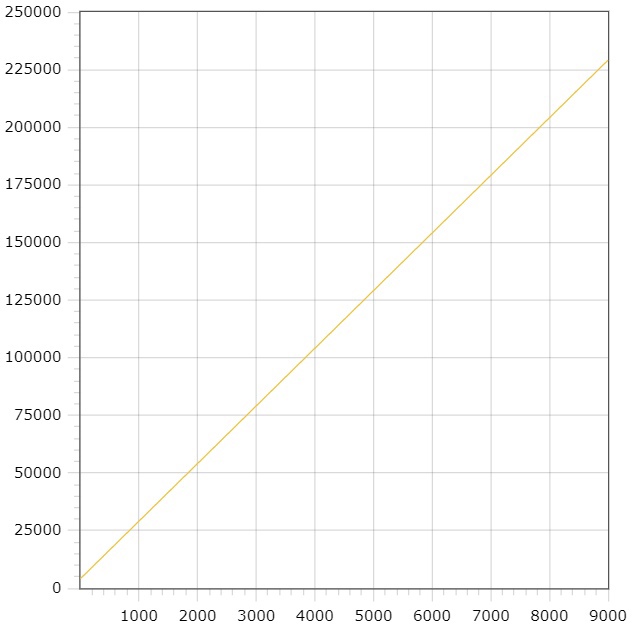


На втором графике можно видеть соотношение длины массива (по x) к количеству

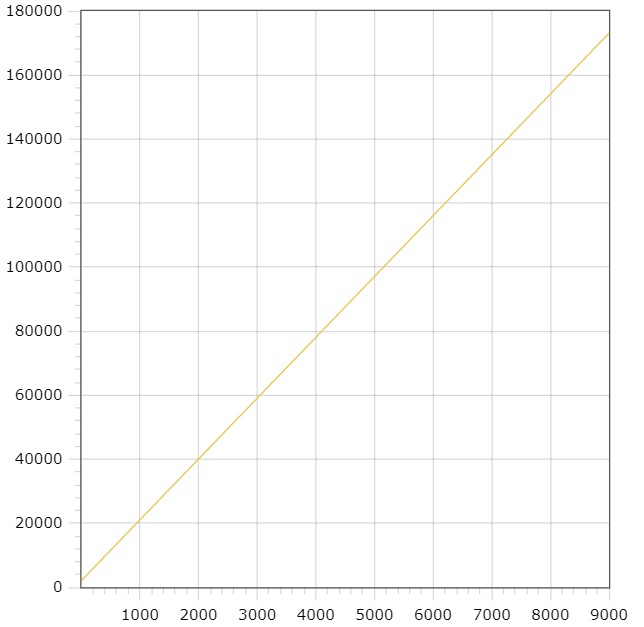
сравнений деленное на сложность (по y). Видно, что значения сходятся к 3,76. Так как график сходится к некоторому значению, то это является экспериментальным подтверждением указанной сложности по сравнениям для данной сортировки.



Третий график показывает отношение количества перестановок (по y) к длине массива (по x).



Четвёртый график показывает отношение количества сравнений (по y) к длине массива (по x).

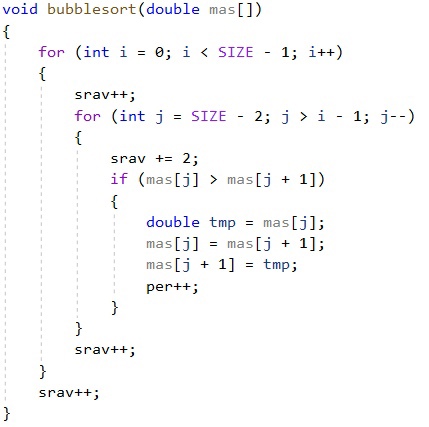


# Заключение

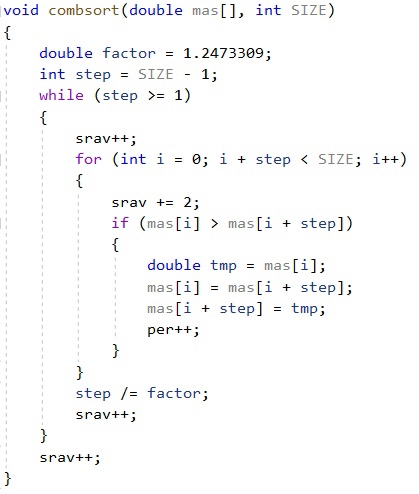
В ходе работы были реализованы: сортировка пузырьком, сортировка расчёской, сортировка слиянием, поразрядная сортировка для типа данных double. Я описал алгоритмы работы сортировок, подтвердил корректность сортировок. Также подтвердил теоретическую сложность для каждой из сортировок.

# Приложение

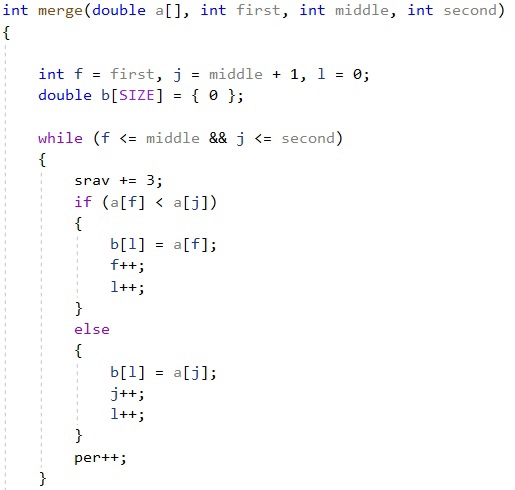
## Сортировка пузырьком.

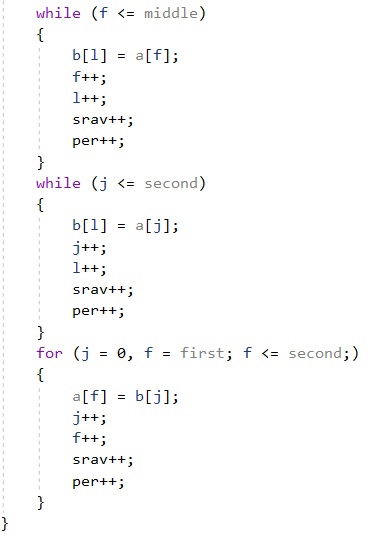


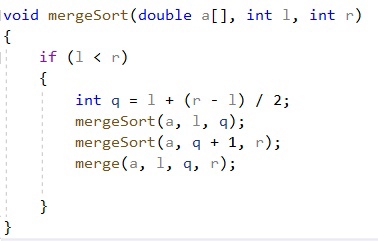
## Сортировка расчёской.



## Сортировка слиянием.







## Поразрядная сортировка.

