Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

Выполнил студент группы КС-36 Алёшин Михаил Алексеевич

Ссылка на репозиторий: https://github.com/MUCTR-IKT-CPP/MAAleshin\_36\_algo

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Дата сдачи: 14.02.2022

Оглавление

[Описание задачи. 2](#_Toc63548272)

[Описание метода/модели. 2](#_Toc63548273)

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

# Необходимо реализовать сортировку слиянием для разного количества элементов. Подсчитать количество дополнительной потребляемой памяти, подсчитать количество вызовов рекурсивной функции, и высоту рекурсивного стека. Построить график потребления памяти и сравнить его с функцией c \* n, построить график худшего, лучшего, и среднего случая для каждой серии тестов для количества вызовов рекурсивных функций. Подобрать константы с1 и с2 для ограничения графика сверху и снизу. Проанализировать полученные графики.

# Описание метода/модели.

# Сортировка слиянием - один из самых быстрых известных универсальных алгоритмов сортировки массивов: в среднем 𝑂(𝑛 log 𝑛 ) обменов при упорядочении 𝑛 элементов; из-за наличия ряда недостатков на практике обычно используется с некоторыми доработками.

# Алгоритм:

1. Сортируемый массив разбивается на две части примерно одинакового размера
2. Каждая из получившихся частей сортируется отдельно, например — тем же самым алгоритмом;
3. Два упорядоченных массива половинного размера соединяются в один.

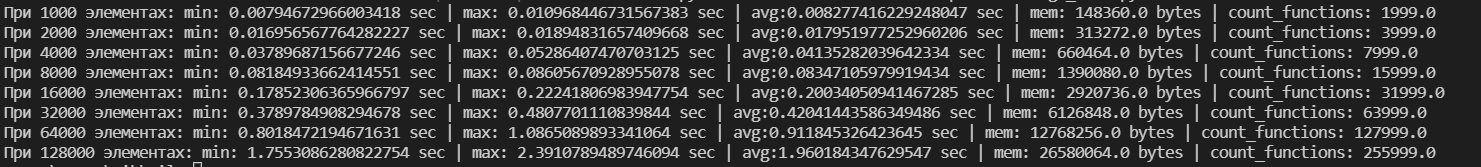
У нас есть два массива a и b (фактически это будут две части одного массива, но для удобства будем писать, что у нас просто два массива). Нам надо получить массив c размером |a|+|b|. Для этого можно применить процедуру слияния. Эта процедура заключается в том, что мы сравниваем элементы массивов (начиная с начала) и меньший из них записываем в финальный. И затем, в массиве у которого оказался меньший элемент, переходим к следующему элементу и сравниваем теперь его. В конце, если один из массивов закончился, мы просто дописываем в финальный другой массив. После мы наш финальный массив записываем вместо двух исходных и получаем отсортированный участок.

Достоинства метода:

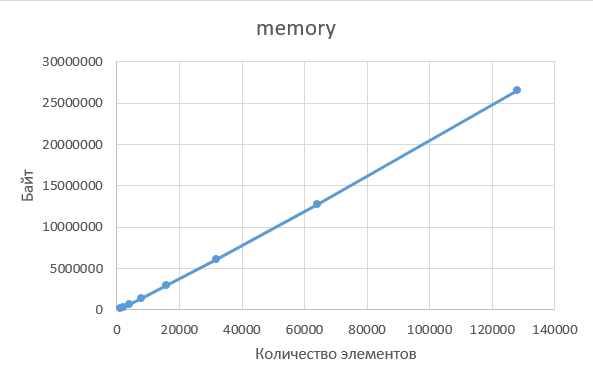
* Устойчива
* Многопоточна
* Распределена

# Выполнение задачи.

Данная сортировка была реализована на языке python. Реализованы функции mergeSortRecursive и merge соответственно. При вызове функции генерируется новый массив от -1 до 1 (числа с плавающей точкой) и вызывается метод анализа сортировки, в котором 20 раз прогоняется каждое количество элементов начиная с 1000 и умножая на 2, заканчивая 128000 элементами. В конце получаем данный результат:

  
Рис. 1. Результаты анализа.

Далее посмотрим результаты потребления памяти на графике:

  
Рис. 2. График зависимости памяти от количества элементов массива.

Если сравнить данный график с графиком c\*n, где константой c является отношение y/x последней точки графика. Получим следующий результат:

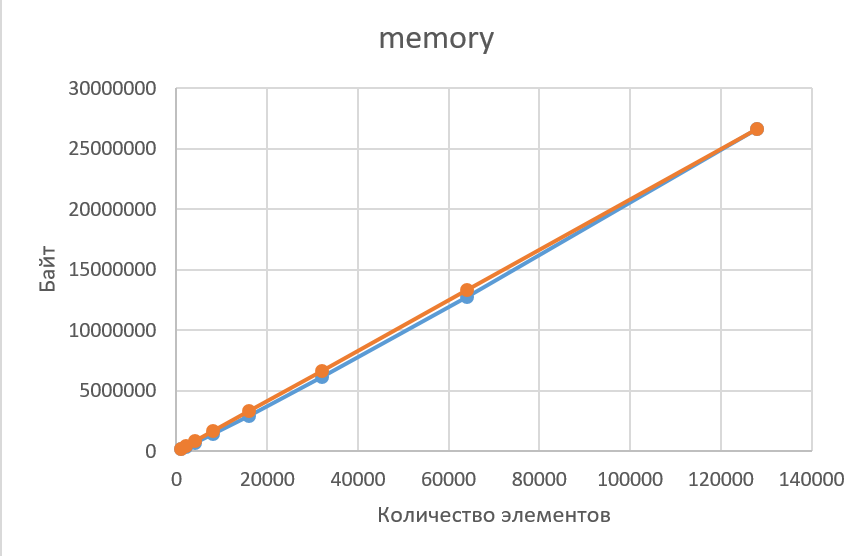


Рис. 3. Функция c\*n.

Из графика видно, что наш график имеет более параболический вид, чем график c\*n, следовательно, при увеличении количества элементов потребуется выделение большего количества памяти.

Количество вызовов функции будет всегда константным и равняться N\*2-1. Далее на графике можно увидеть график, который будет очень схож с графиком 2\*n.

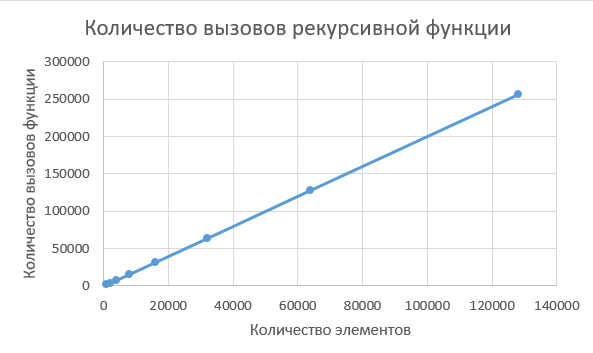


Рис. 4. Количество вызовов рекурсивной функции.

Далее посмотрим на результат времени работы сортировки относительно количества элементов:

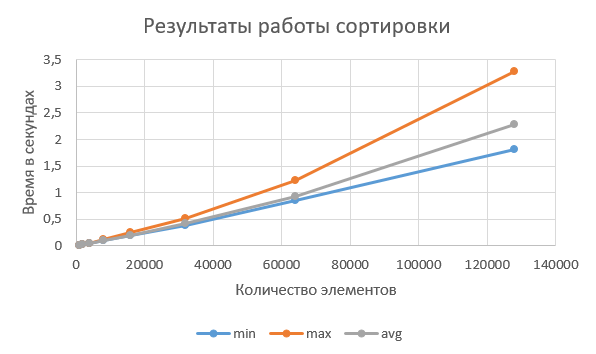


Рис. 5. Время работы сортировки.

Далее подберем константы с1 и с2 для графика с1\*N\*log(N) и c2\*N\*log(N) соответственно, чтобы первый график ограничивал наш график сверху, а график с константой с2 ограничивал наш график снизу. Подобрав значения, получим с1 = 5,04801E-06 и с2 = 2,33333E-06. Далее посмотрим результаты на графике:

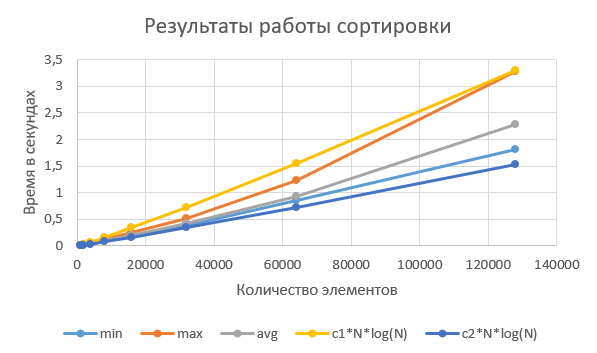


Рис. 6. Константы с1 и с2.

# Заключение.

Из полученных нами результатов видно, что данный метод сортировки является довольно эффективным, так как имеет скорость O(n\*log(n)). На рис.1 наглядно видно, что при увеличении элементов в 2 раза скорость работы сортировки увеличивается чуть более, чем в 2 раза, что доказывает сложность O(n\*log(n)). Недостатком данного метода является то, что он может занимать большое количество памяти для хранения промежуточных результатов. Однако метод является стабильным, несмотря на данный недостаток, связанный с памятью. Данный метод сортировки является рекурсивным, однако количество рекурсий является константным (в моём случае) и равняется 2\*n-1. Деградация метода отсутствует.