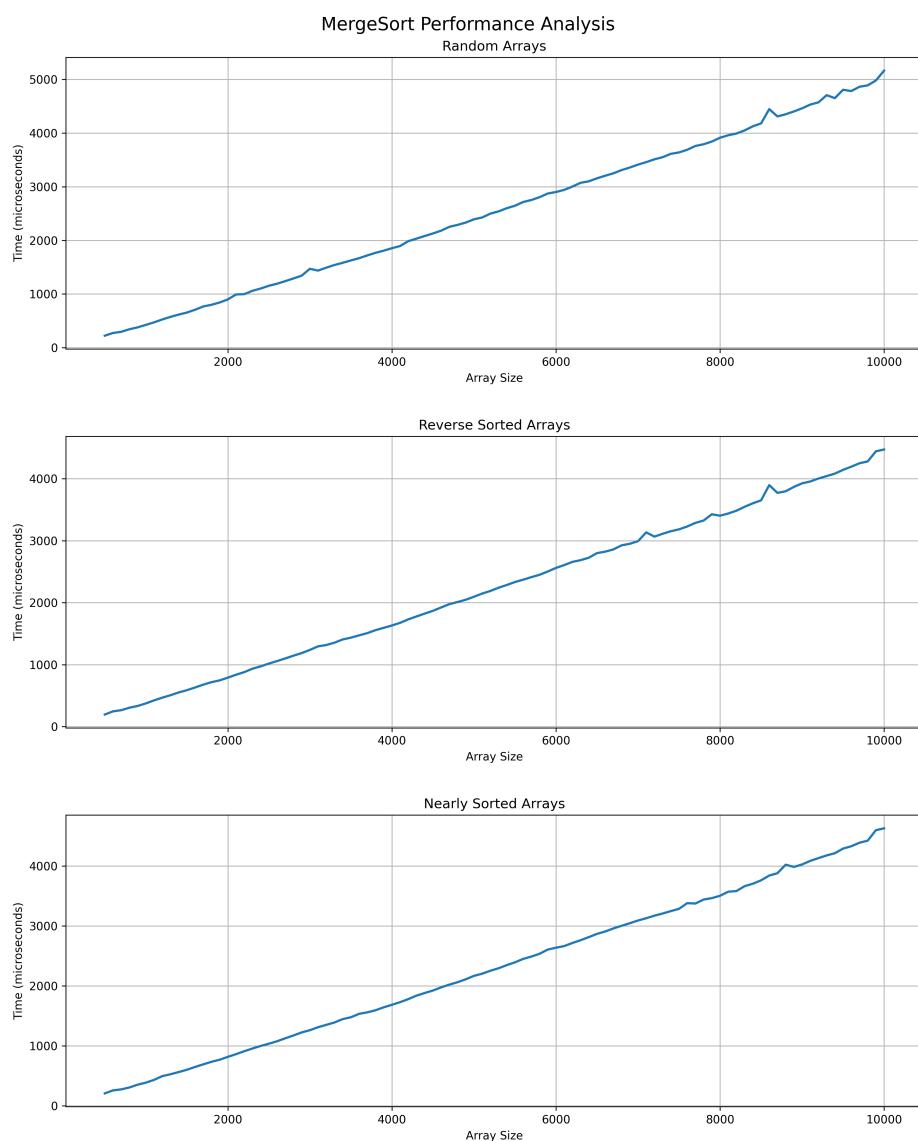
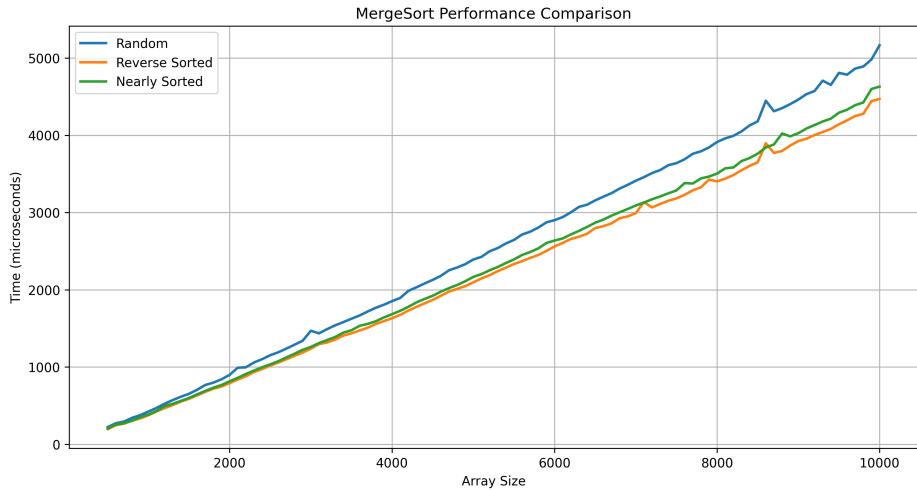


# A2

Трофим Петянов

0. ID посылки: **293153762**. [Ссылка на репозиторий](#). Реализации там же.
1. **Этап 1.** Подготовка тестовых данных:  
ArrayGenerator реализован (см. репозиторий/Program).  
Константы для генерации массивов задаются в arrayGenerator.h  
файле, реализация написана в arrayGenerator.cpp.
2. **Этап 2.** Эмпирический анализ стандартного алгоритма MergeSort:



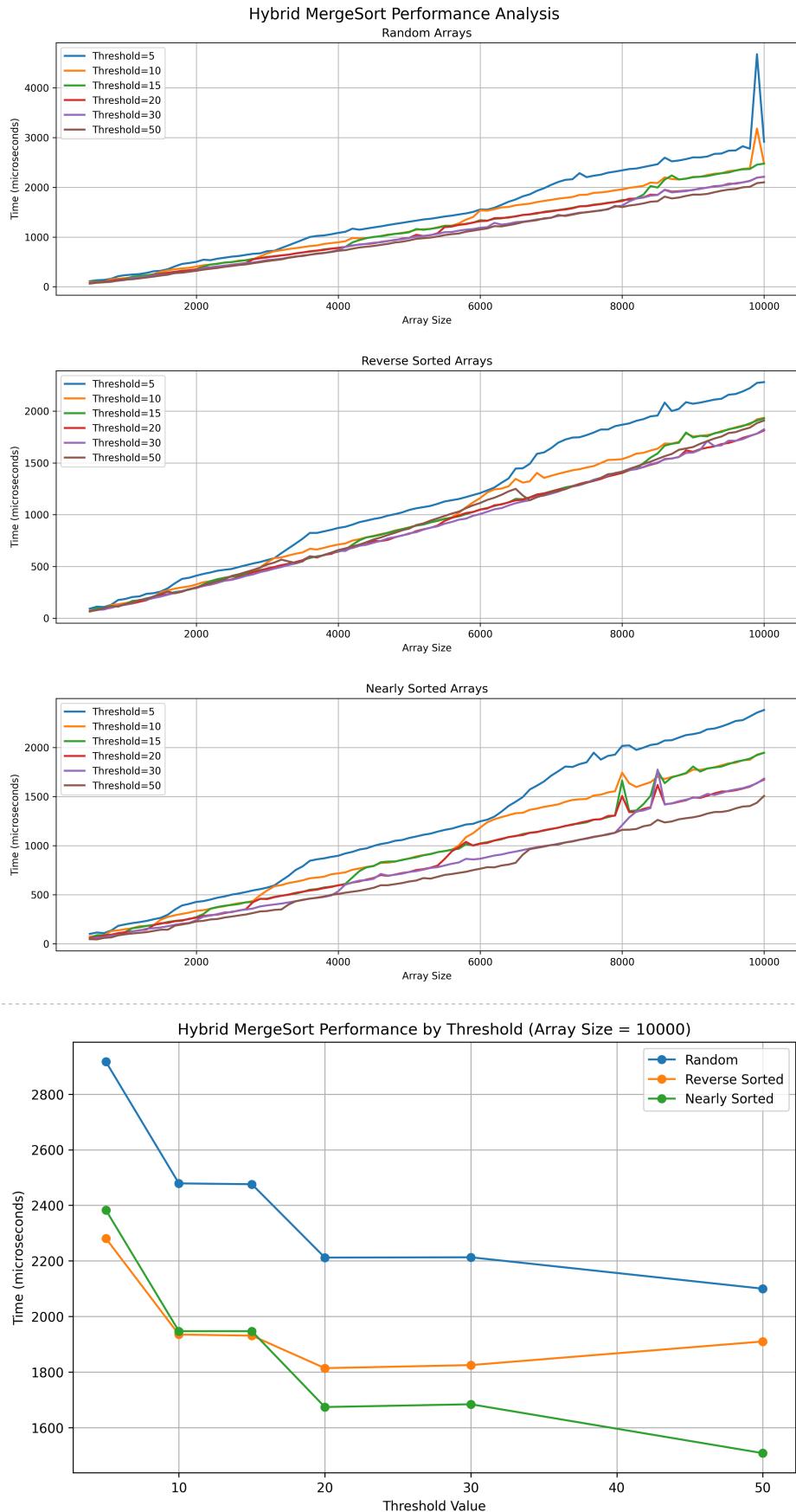


Замеры проведены. Влияние других программ минимизировано. В качестве единиц измерения были выбраны микросекунды. Для более точного ответа каждый тест изначально выполнялся по 5 раз, однако возникал необъяснимо большой пик во времени выполнения для значений около 9.000, поэтому была выбрана константа 10. Выбиралось среднее время выполнения каждого теста.

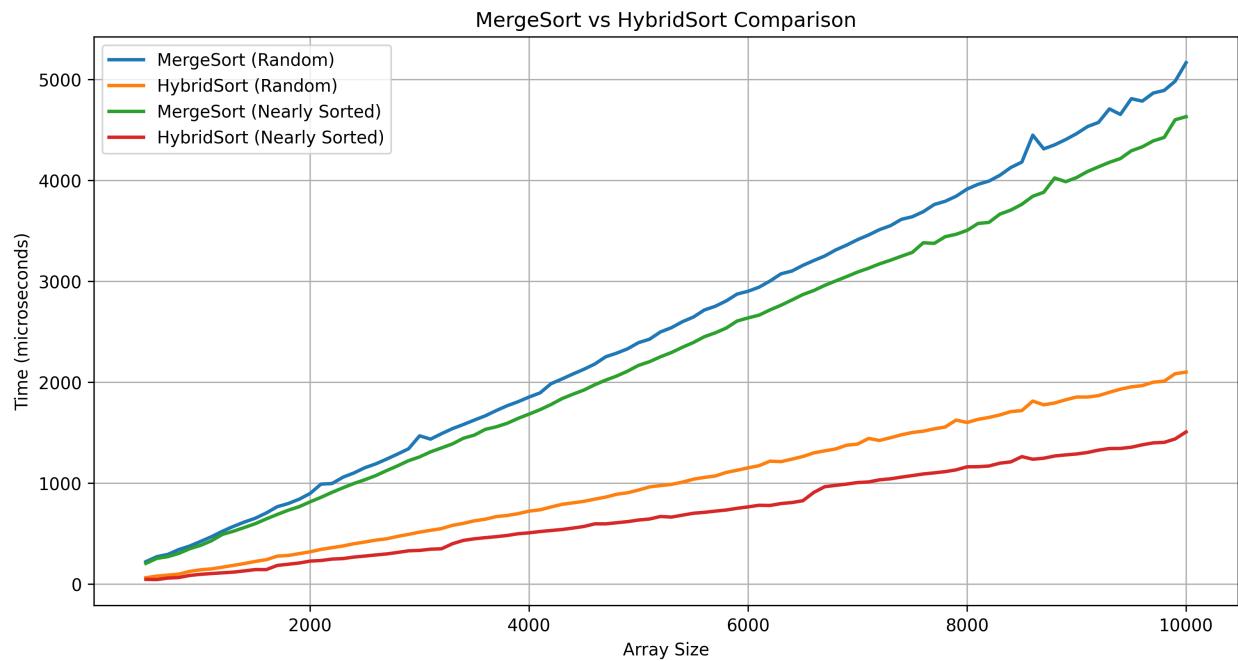
Анализ показал, что MergeSort демонстрирует ожидаемое поведение с ростом времени выполнения  $O(n \log n)$ , при этом наиболее эффективно обрабатываются обратно отсортированные массивы, а случайные массивы требуют заметно больше времени на сортировку, что заметно при увеличении размера входных данных.

3. **Этап 3.** Эмпирический анализ алгоритма Merge+InsertionSort:  
Замеры были проведены на Этапе 2. Были рассмотрены следующие трэшхолды: 5, 10, 20, 30 и 50. Функции эмпирического замера времени работы рассматриваемых алгоритмов сортировки реализованы в `sortTester.h` и `sortTester.cpp`.

Анализ показал, что большие пороговые значения (30, 50) дают лучшую производительность для всех типов массивов. При этом порог 5 стабильно показывает худшие результаты. Наиболее эффективной оказалась работа алгоритма на почти отсортированных массивах с порогом 50, что логично, учитывая преимущества сортировки вставками на почти упорядоченных данных.



#### 4. Этап 4. Сравнительный анализ.



4.1. Гибридная версия алгоритма показывает значительно лучшую производительность для всех типов массивов. Для массива размером 10000 элементов гибридная версия (с порогом 50) работает быстрее классического MergeSort:

- На случайных массивах: в 2.46 раза (5165 vs 2100 мкс)
- На почти отсортированных массивах: в 3.07 раза (4630 vs 1508 мкс)

4.2. Пороговые значения:

Гибридный алгоритм не показал ухудшения производительности по сравнению с классическим MergeSort ни при одном из тестируемых порогов (5-50).

Оптимальное пороговое значение оказалось равным 50, показывая лучшие результаты для всех типов массивов.

Порогового значения, когда MergeSort оказался бы лучше гибридного в рамках данного эксперимента не нашлось.

P.S. в конце я попытался найти это пороговое значение, из-за чего удалились предыдущие данные, поэтому то, что лежит в репозитории может немного отличаться.