Задача A2. Анализ MERGE+INSERTION SORT

Известно, что алгоритм MERGE SORT является одним из оптимальных алгоритмов сортировки на основе сравнений элементов и имеет сложность O(n·log n), а алгоритм INSERTION SORT имеет сложность O(n^2). Однако на массивах сравнительно небольшого размера INSERTION SORT сортирует быстрее MERGE SORT ввиду лучшей оценки скрытой константы, а также снижения накладных расходов, которые связаны с рекурсией. В рамках этой задачи требуется провести экспериментальное исследование двух реализаций алгоритма MERGE SORT:

• стандартной рекурсивной (с выделением дополнительной памяти)

• гибридной, которая на массивах малого размера переключается на INSERTION SORT.

1. Реализация гибридного алгоритма сортировки MERGE+INSERTION SORT.

ID посылки: 292495717

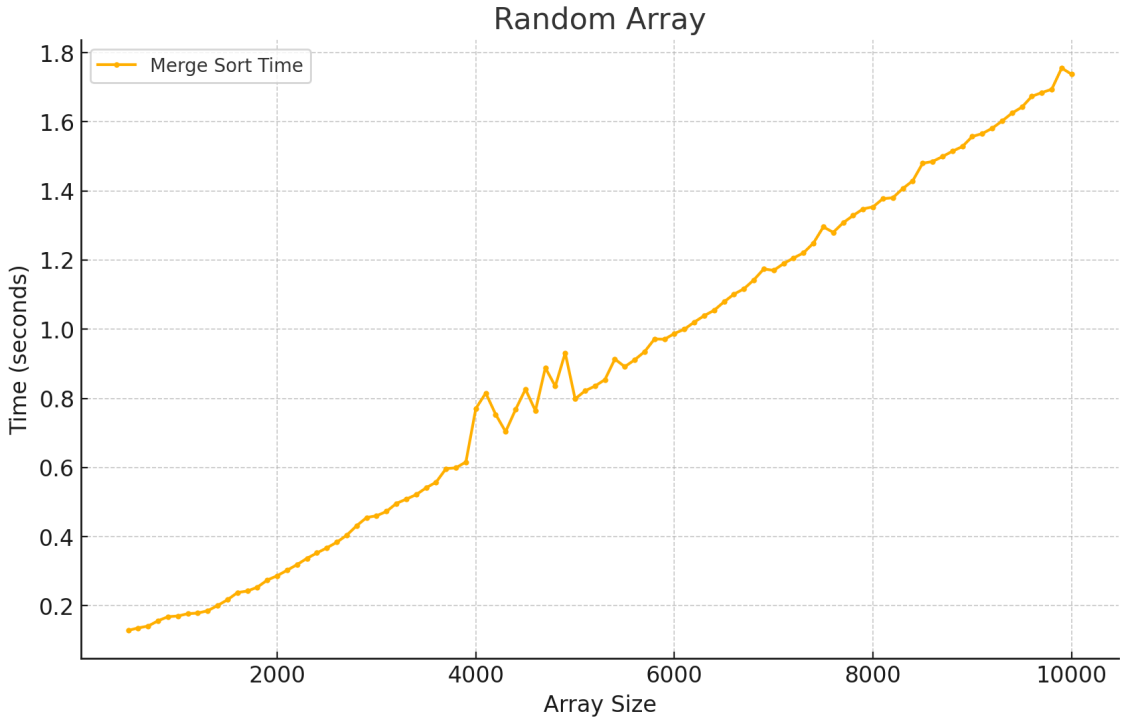
2. Реализация внутренней инфраструктуры для экспериментального анализа — классы ArrayGenerator и SortTester.

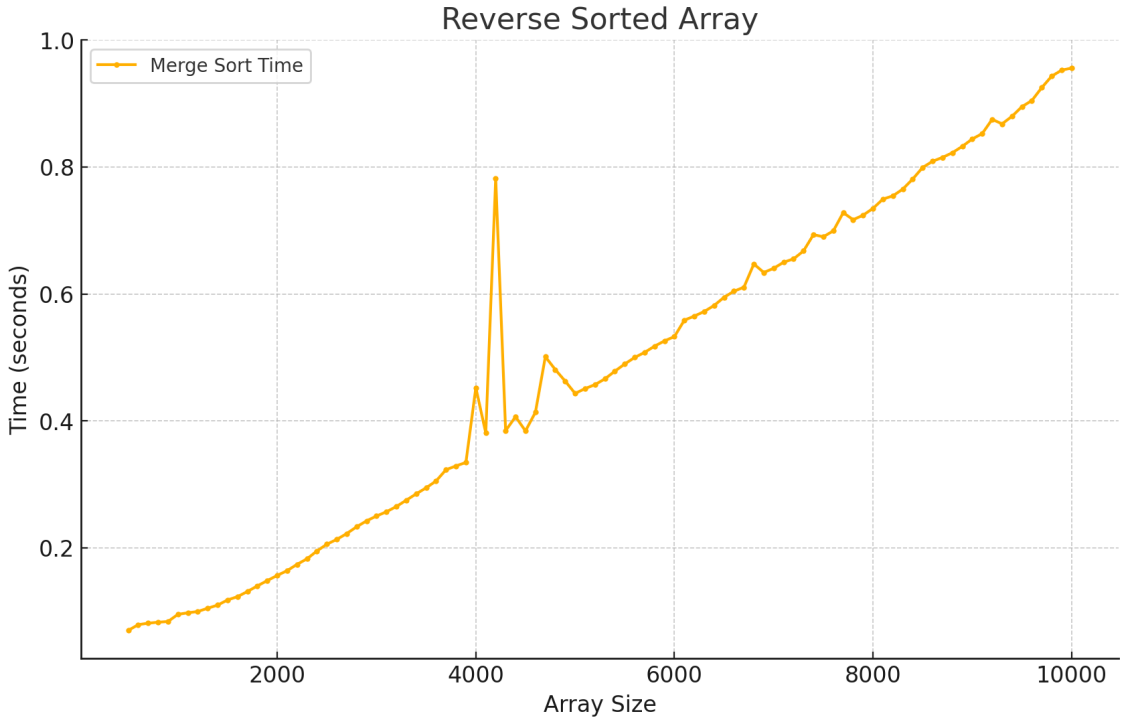
Реализацию и исходные данные можно найти тут:

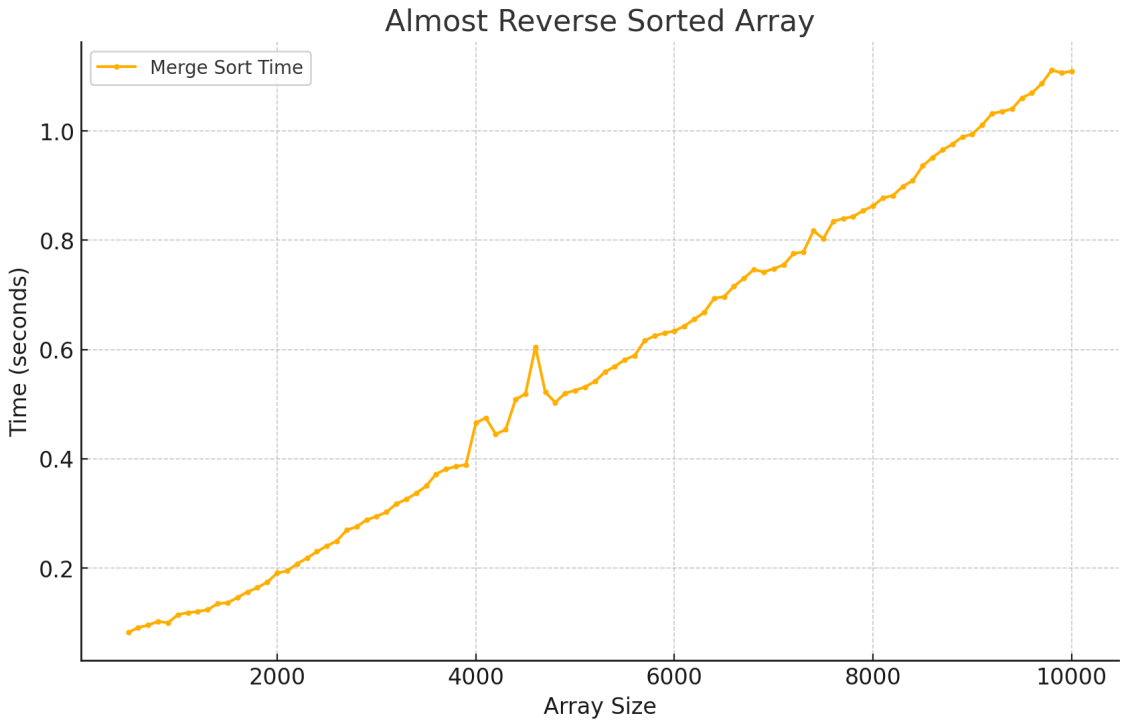
`https://github.com/artishokq/A2.git`

3. Представление эмпирических замеров времени работы рассматриваемых алгоритмов.

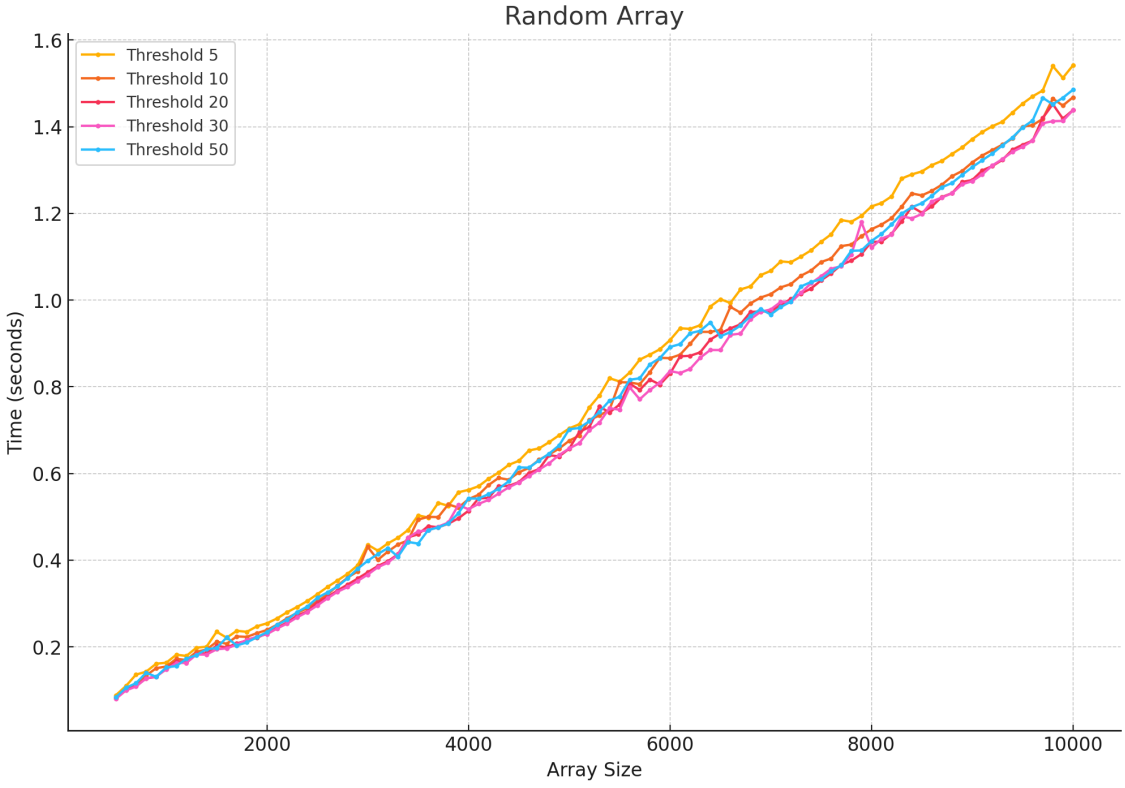
Эмпирический анализ стандартного алгоритма MERGE SORT:

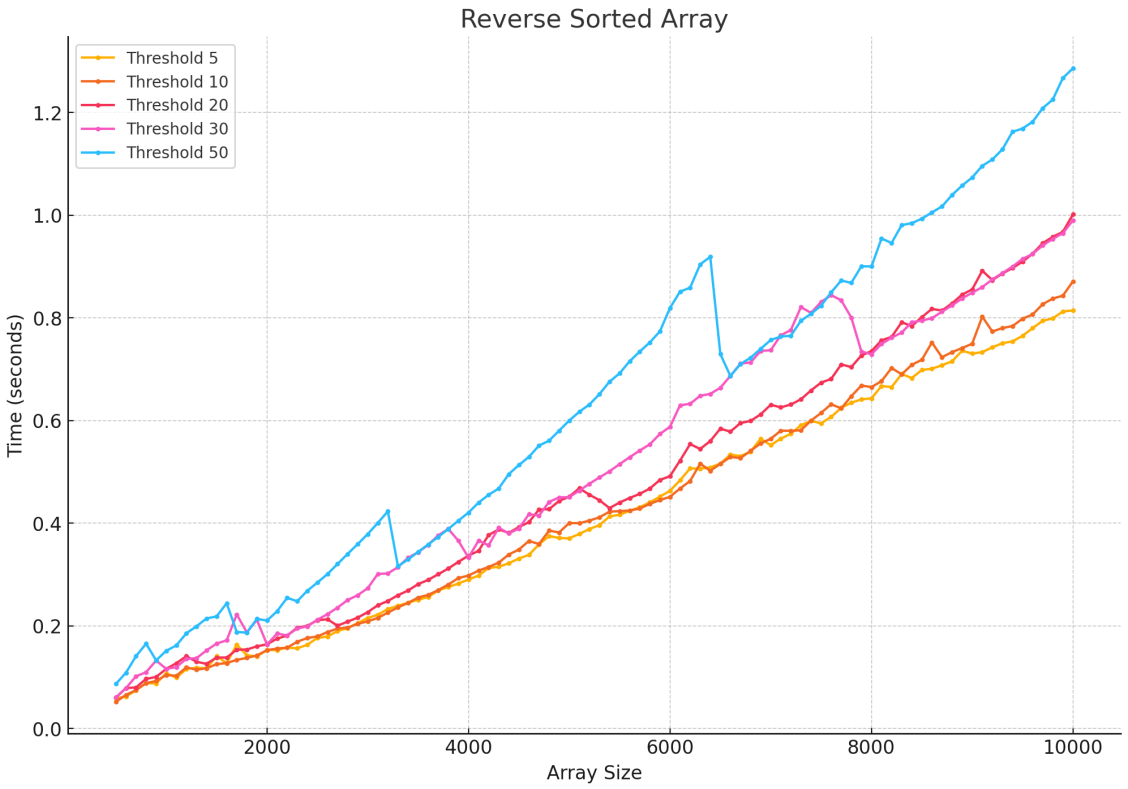


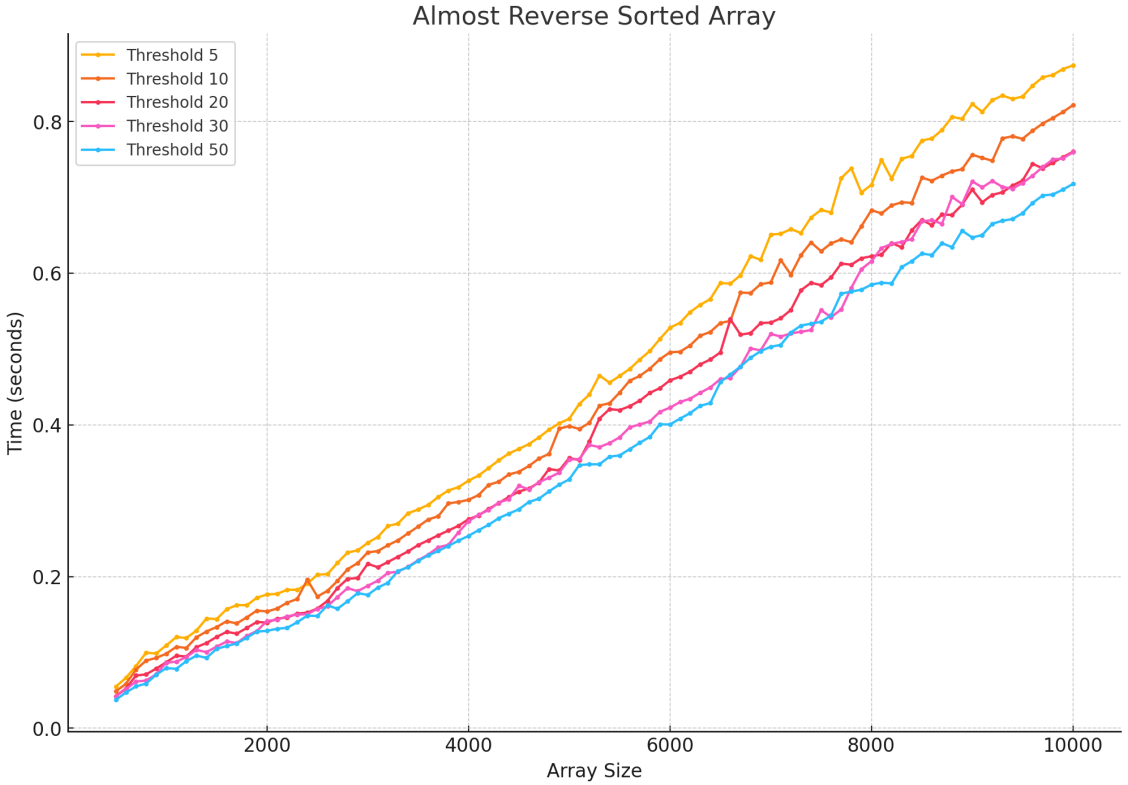




Эмпирический анализ гибридного алгоритма MERGE+INSERTION SORT:




Reverse Sorted Array
Threshold 5
Threshold 10
Threshold 20
Threshold 30
Threshold 50
Time (seconds)
Array Size
0.0
0.2
0.4
0.6
0.8
1.0
1.2
2000
4000
6000
8000
10000


Almost Reverse Sorted Array
Threshold 5
Threshold 10
Threshold 20
Threshold 30
Threshold 50
Time (seconds)
Array Size
0.0
0.2
0.4
0.6
0.8
2000
4000
6000
8000
10000

4. Сравнительный анализ полученных эмпирических данных.

Использование дополнительной памяти и операций копирования в стандартном Merge Sort существенно влияет на его производительность при больших размерах массивов. Гибридный Merge+Insertion Sort, за счёт уменьшения числа копирований, способен превзойти стандартную реализацию по скорости на больших данных.

Влияние порога на время выполнения:
Низкие значения порога (T=5):

Гибридный алгоритм работает хуже, если значение порога < 10 на массиве из рандомных значений.

Аналогично для почти отсортированного массива, меньшие значения порога приводят к ухудшению работы алгоритма, однако всё же лучше обычного MergeSort.

Однако для обратно отсортированного массива, меньшие значения порога улучшают работу алгоритма.

Средние значения порога (T=10 – T=30):

Гибридный алгоритм показывает оптимальную производительность в массиве из рандомных значений. Особенно для порога от 20 до 30.

При обратно отсортированном массиве, работает похуже чем порог = 5.

Для почти отсортированного массива, значения порога 20-30 работают лучше чем при пороге < 10. Увелечение порога улучшает работу алгоритма.

Высокие значения порога (T=40 – T=50):

При значениях порога > 30, алгоритм начинает работать хуже на массиве из рандомных чисел, но лучше чем при значениях < 5.

Также, при значениях > 30, алгоритм работает всё хуже и хуже для обратно отсортированного массива, наблюдается сильный прирост времени исполнения. Работает хуже обычного MergeSort.

Однако, для почти обратно отсортированного массива, увеличение порога улучшает работу алгоритма!

Оптимальное значение порога около THRESHOLD $= 20$. Значение порога, после которого гибридный алгоритм становится медленнее THRESHOLD > 30 или THRESHOLD < 10.

Использование дополнительной памяти и операций копирования в стандартном Merge Sort существенно влияет на его производительность при больших размерах массивов. Гибридный Merge+Insertion Sort, за счёт уменьшения числа копирований, способен превзойти стандартную реализацию по скорости на больших данных.