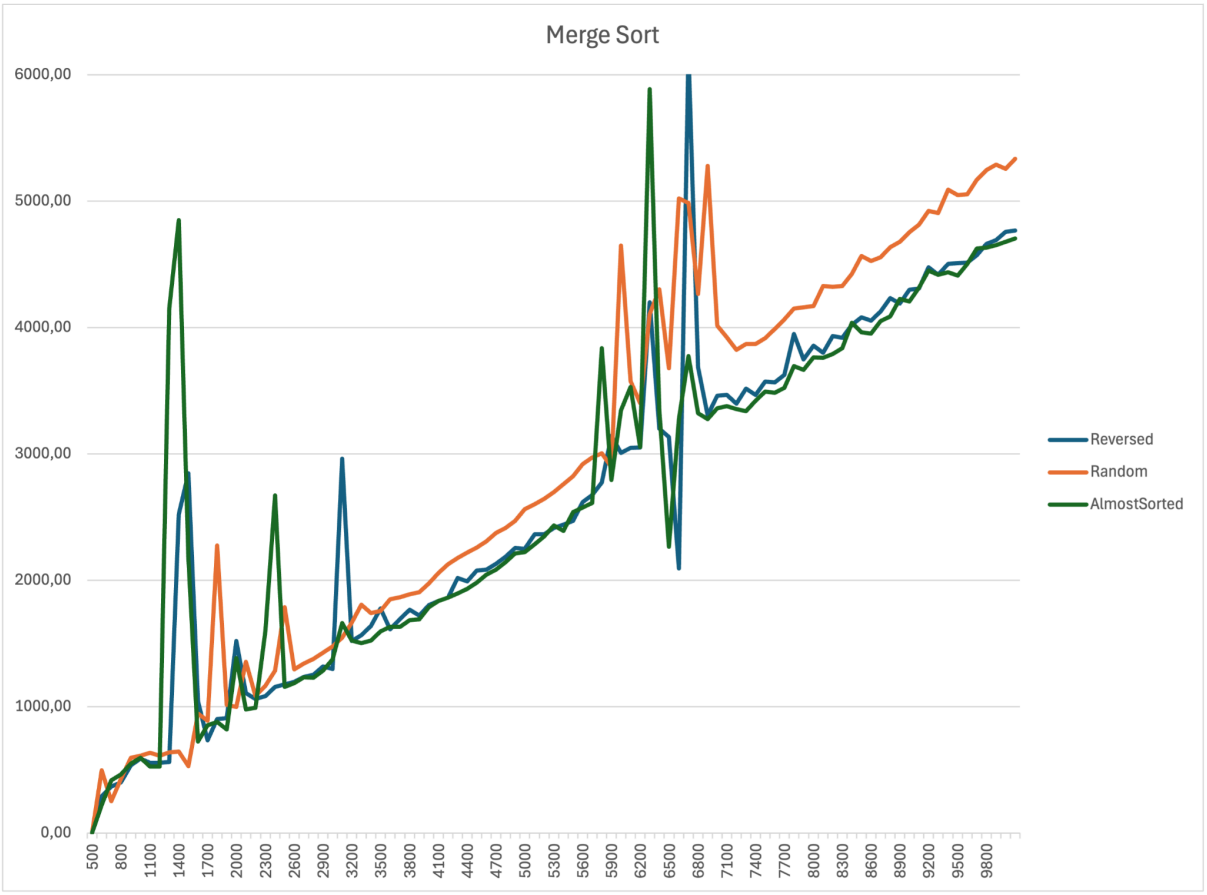
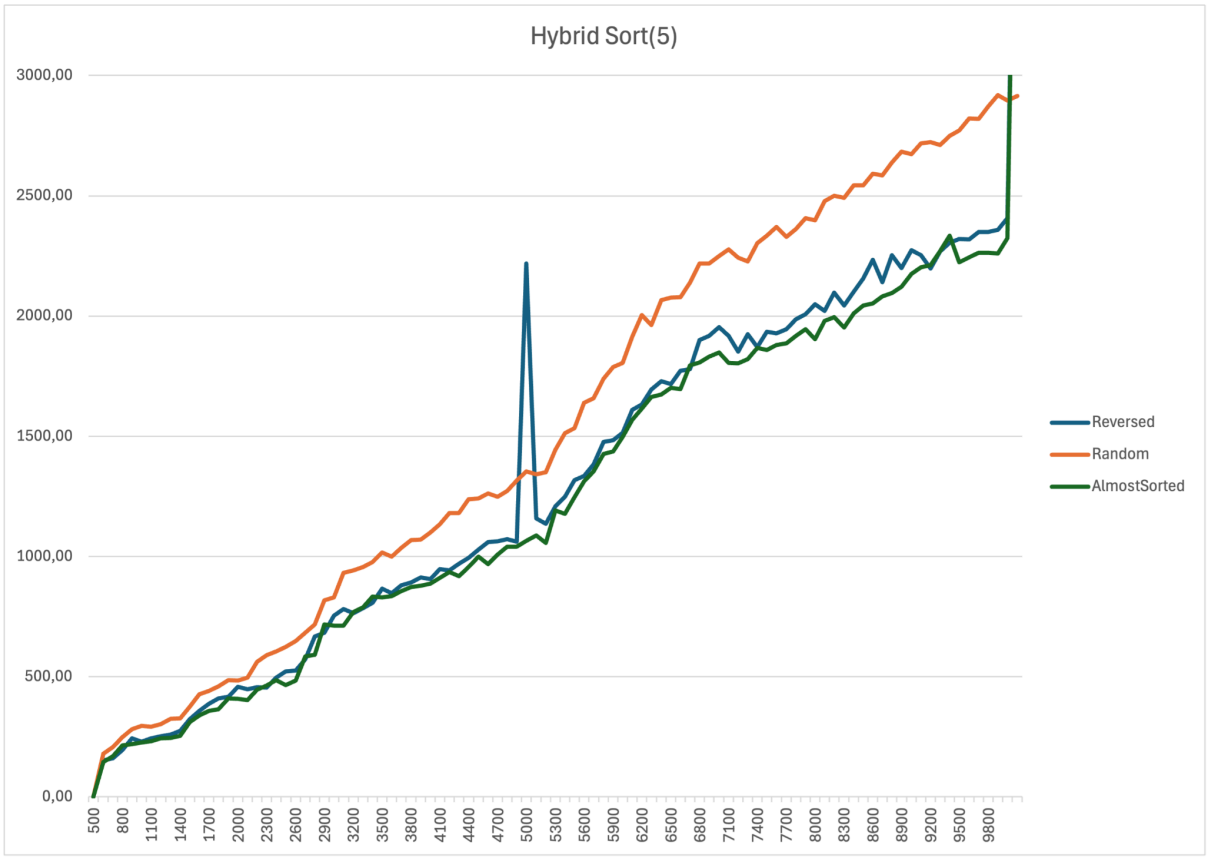
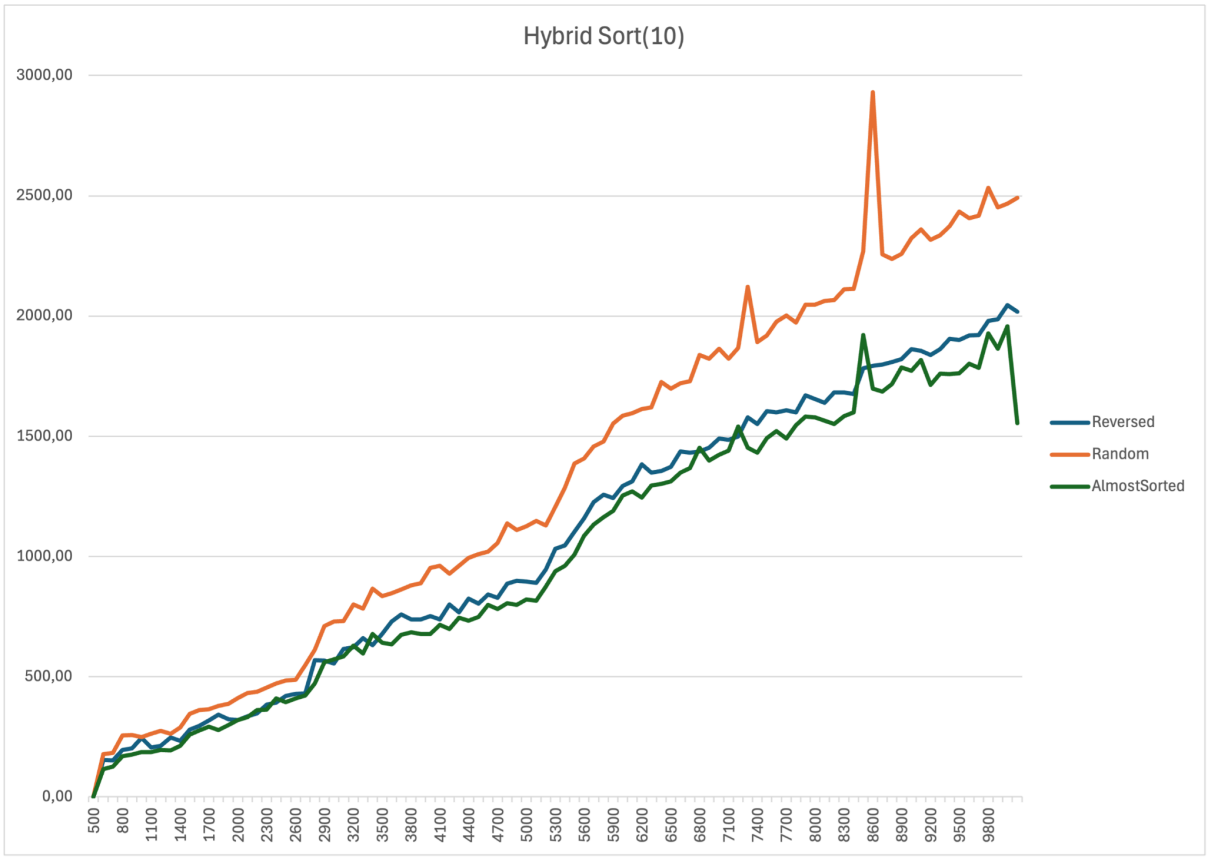
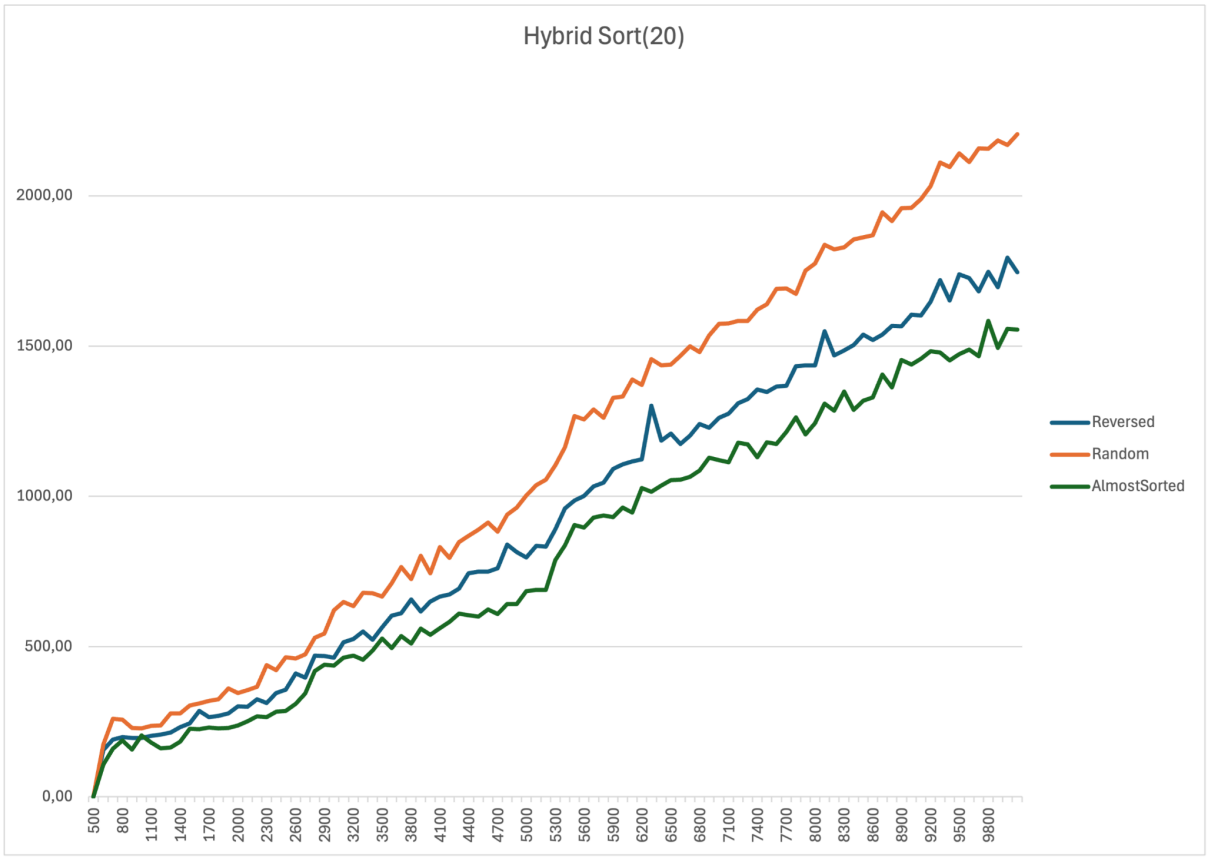
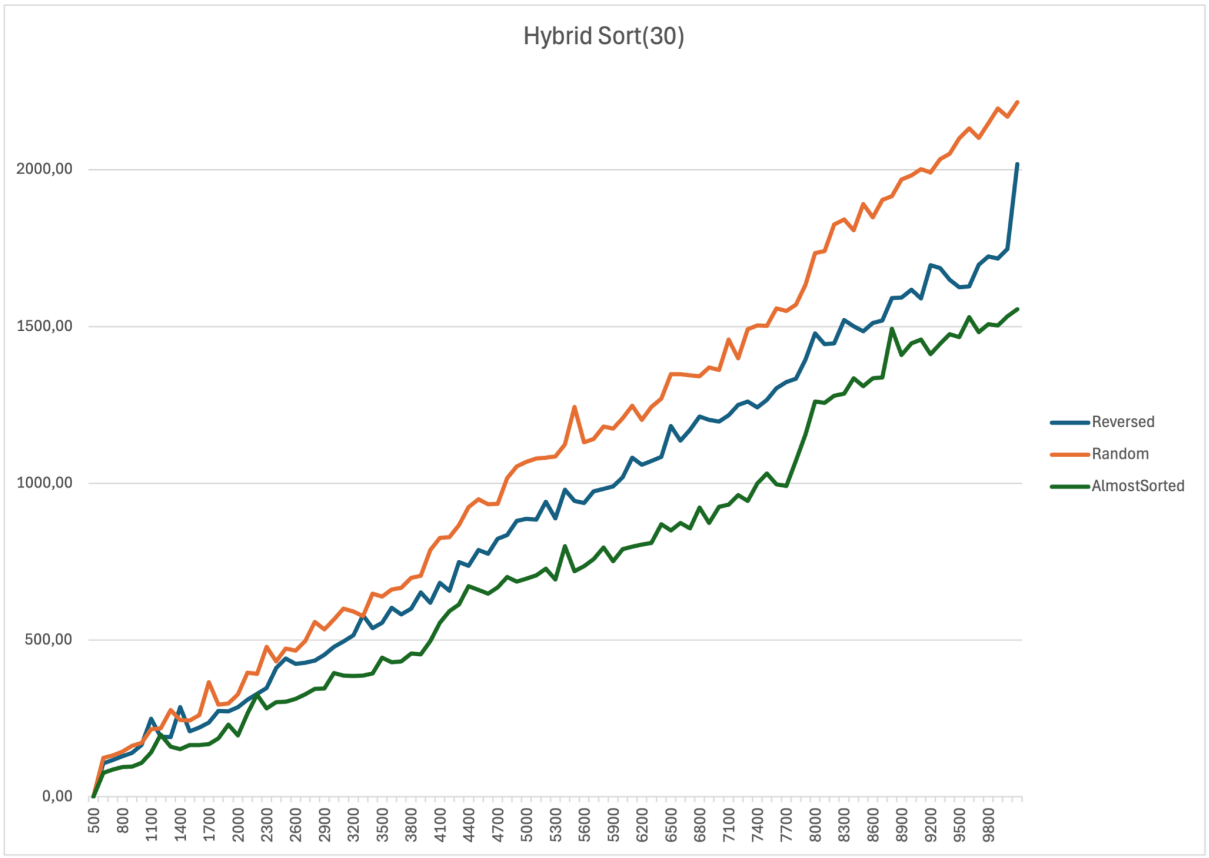
A2.
ID посылки: 293148516
Ссылка на гитхаб с кодом: https://github.com/vilina4kaa/Algorithms.git
Графики (вертикальная ось - время в мс, горизонтальная - size массива):
Отдельные для каждой функции:

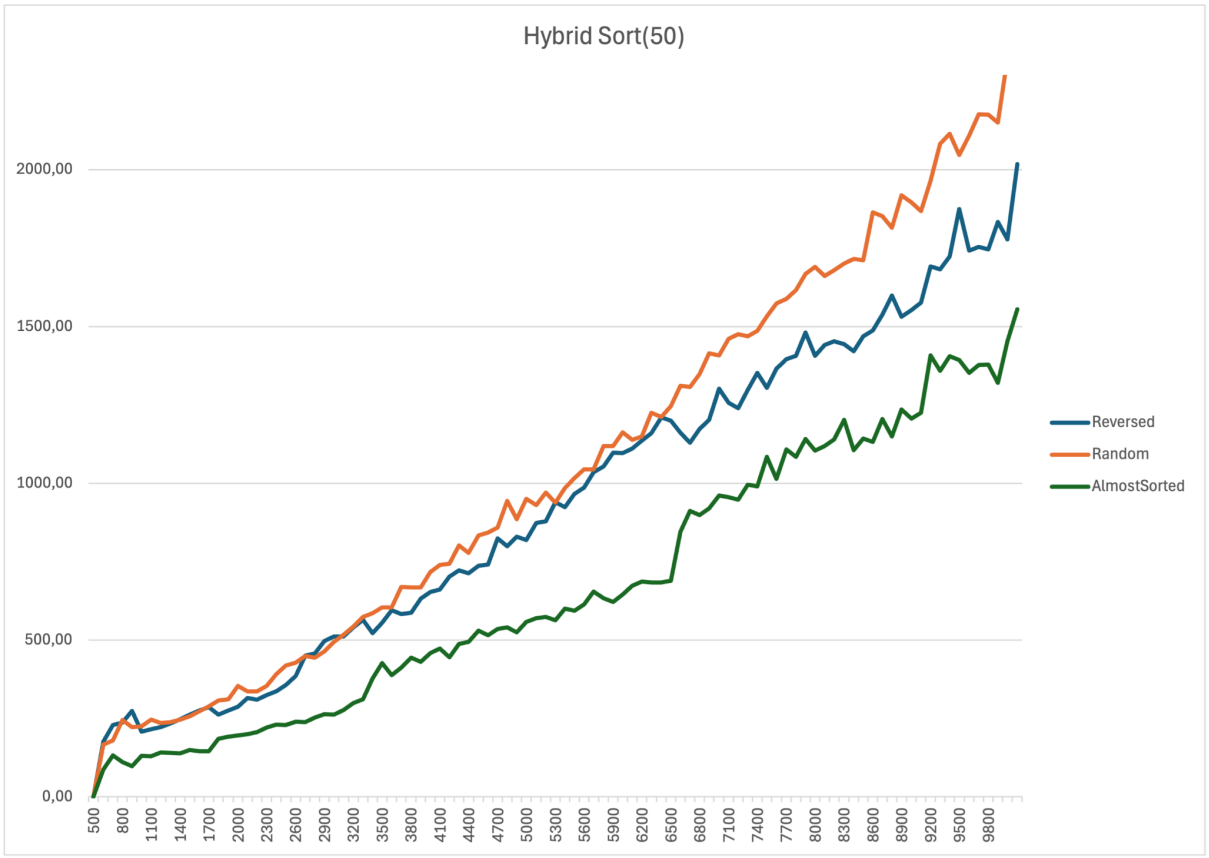



Hybrid Sort(5)
3000,00
2500,00
2000,00
1500,00
1000,00
500,00
0,00
500 800 1100 1400 1700 2000 2300 2600 2900 3200 3500 3800 4100 4400 4700 5000 5300 5600 5900 6200 6500 6800 7100 7400 7700 8000 8300 8600 8900 9200 9500 9800
Reversed
Random
AlmostSorted

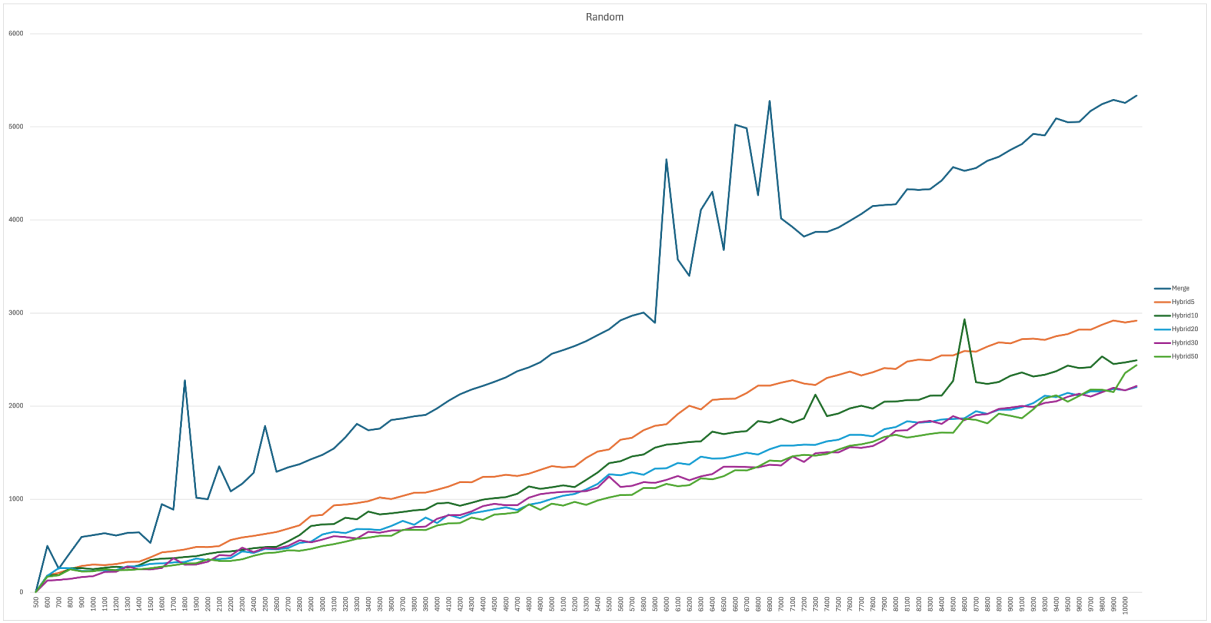

Hybrid Sort(10)
3000,00
2500,00
2000,00
1500,00
1000,00
500,00
0,00
500 800 1100 1400 1700 2000 2300 2600 2900 3200 3500 3800 4100 4400 4700 5000 5300 5600 5900 6200 6500 6800 7100 7400 7700 8000 8300 8600 8900 9200 9500 9800
Reversed
Random
AlmostSorted

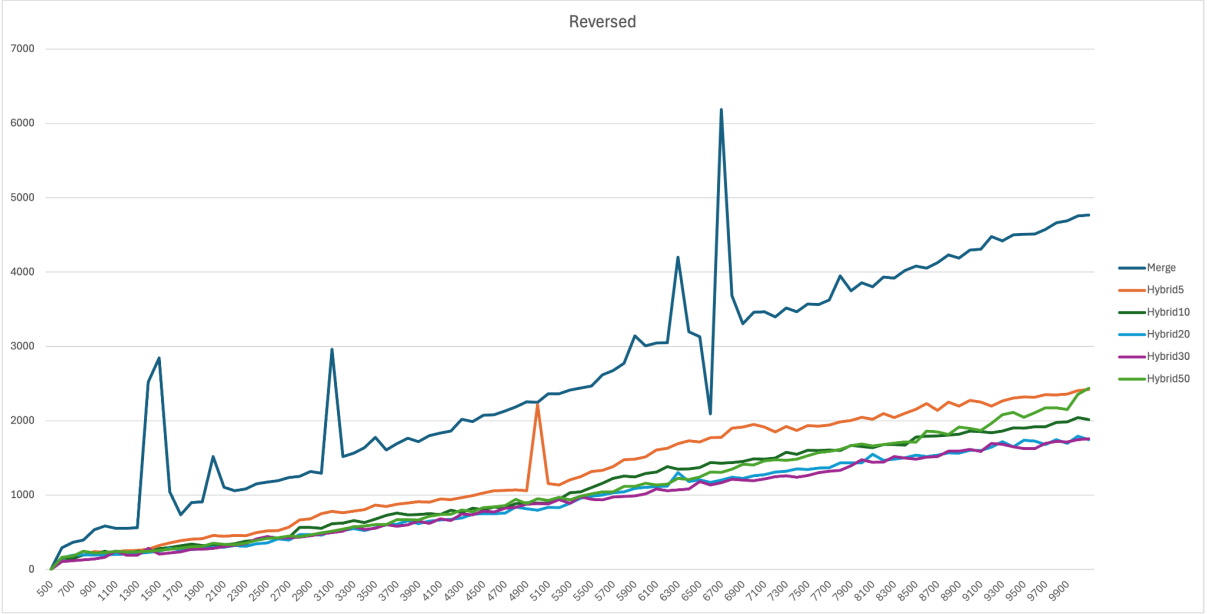

Hybrid Sort(20)
2000,00
1500,00
1000,00
500,00
0,00
500
800
1100
1400
1700
2000
2300
2600
2900
3200
3500
3800
4100
4400
4700
5000
5300
5600
5900
6200
6500
6800
7100
7400
7700
8000
8300
8600
8900
9200
9500
9800
Reversed
Random
AlmostSorted

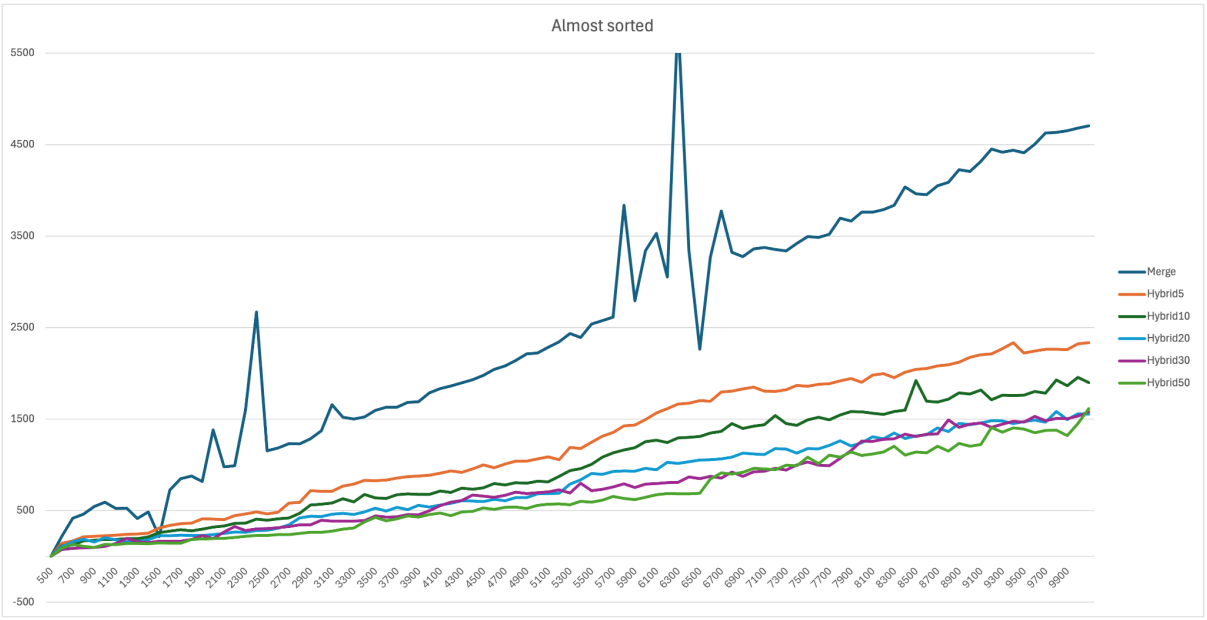

Hybrid Sort(30)
2000,00
1500,00
1000,00
500,00
0,00
500
800
1100
1400
1700
2000
2300
2600
2900
3200
3500
3800
4100
4400
4700
5000
5300
5600
5900
6200
6500
6800
7100
7400
7700
8000
8300
8600
8900
9200
9500
9800
Reversed
Random
AlmostSorted


Hybrid Sort(50)
2000,00
1500,00
1000,00
500,00
0,00
500 800 1100 1400 1700 2000 2300 2600 2900 3200 3500 3800 4100 4400 4700 5000 5300 5600 5900 6200 6500 6800 7100 7400 7700 8000 8300 8600 8900 9200 9500 9800
Reversed
Random
AlmostSorted

Сводные:


Random
6000
5000
4000
3000
2000
1000
0
Merge
Hybrid5
Hybrid10
Hybrid20
Hybrid30
Hybrid50





Выводы:

Для анализа скорости работы merge sort и гибридной merge-insertion sort я написала программу и построила соответствующие графики.

В моем коде я написала функции для сортировки. Функции merge() и mergeSort() реализуют merge sort, также есть функция insertion sort. А для добавления insertion sort к hybrid я добавила дополнительный параметр в аргументы функции - threshold. Он показывает, на какой длине массивы нужно перестать мерджить и сделать вместо этого сортировку вставками.

Для генерации рандомного массива длиной 10000 я использовала функцию generateRandomArray(), в которой использую рандомный алгоритм mt19937. Для генерации перевернутого и почти отсортированных массивов я использую функции generateReversedArray() и generateAlmostSortedArray(). Для отделения куска некоторой длины от массива, я использую функцию getPartOfArray().

По результатам замеров и построенным графикам можно сделать следующие выводы:

1. На полностью рандомных массивах лучше всего себя показывает hybridSort(50);
2. На перевернутых массивах лучше всего себя ведет hybridSort(30), очень близко к ней hybridSort(20). Она все еще дает значительный выигрыш перед обычным mergeSort, но при этом работает лучше hybridSort(50), так как перевернутый массив - худший случай для insertionSort, и она работает за O(n^2).
3. На почти отсортированных массивах лучший результат снова дала hybridSort(50), при этом она значительно отрывается от конкурентов (заметно для merge sort и hybridSort(5-10)). Так происходит потому что чем больше отсортированный массив - тем более это хороший случай для insertionSort, она работает почти за линию.
4. Стоит отметить, что обычная mergeSort ведет себя примерно одинаково при любых данных. Действительно, вне зависимости от начального состояния массива, она выполнит одинаковое количество операций (разделения и слияния).
5. Также можно заметить, что на отдельных графиках почти всегда хуже всего работают random массивы, на merge sort и hybridSort(5-10) почти отсортированные и перевернутые работают примерно одинаково, а вот на hybridSort(20-50) почти отсортированные вырываются вперед, выполняя сортировку за меньшее время.