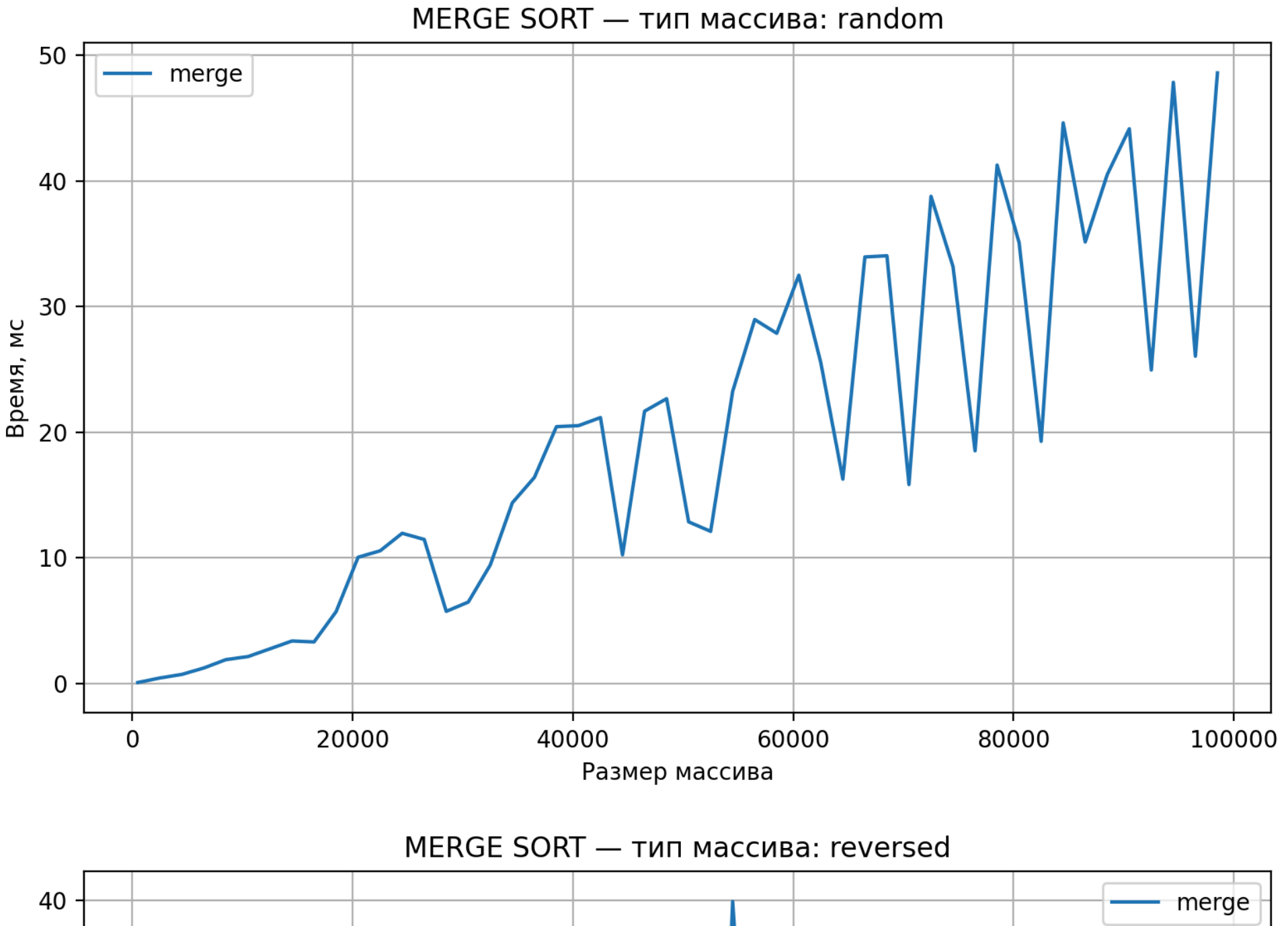
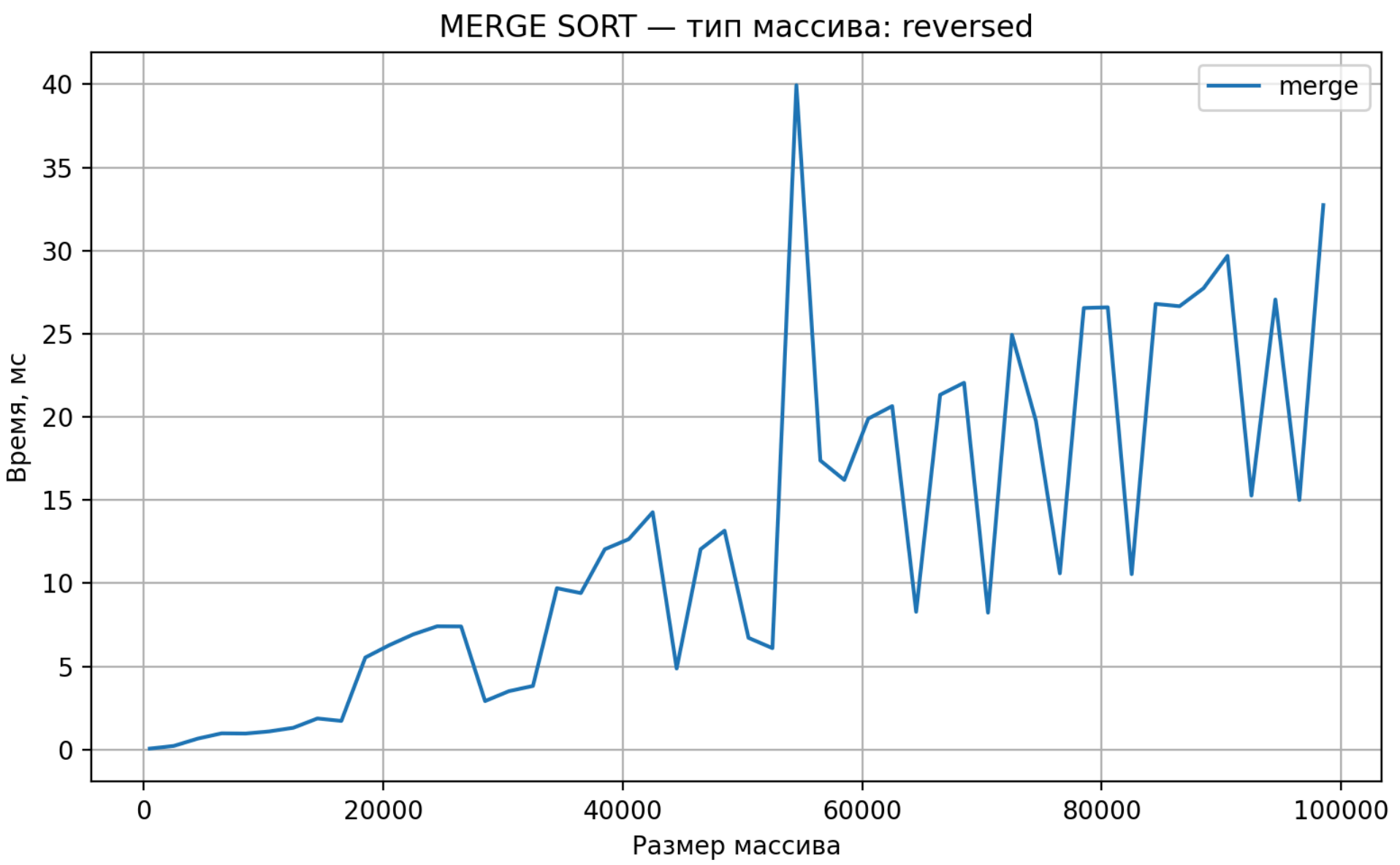
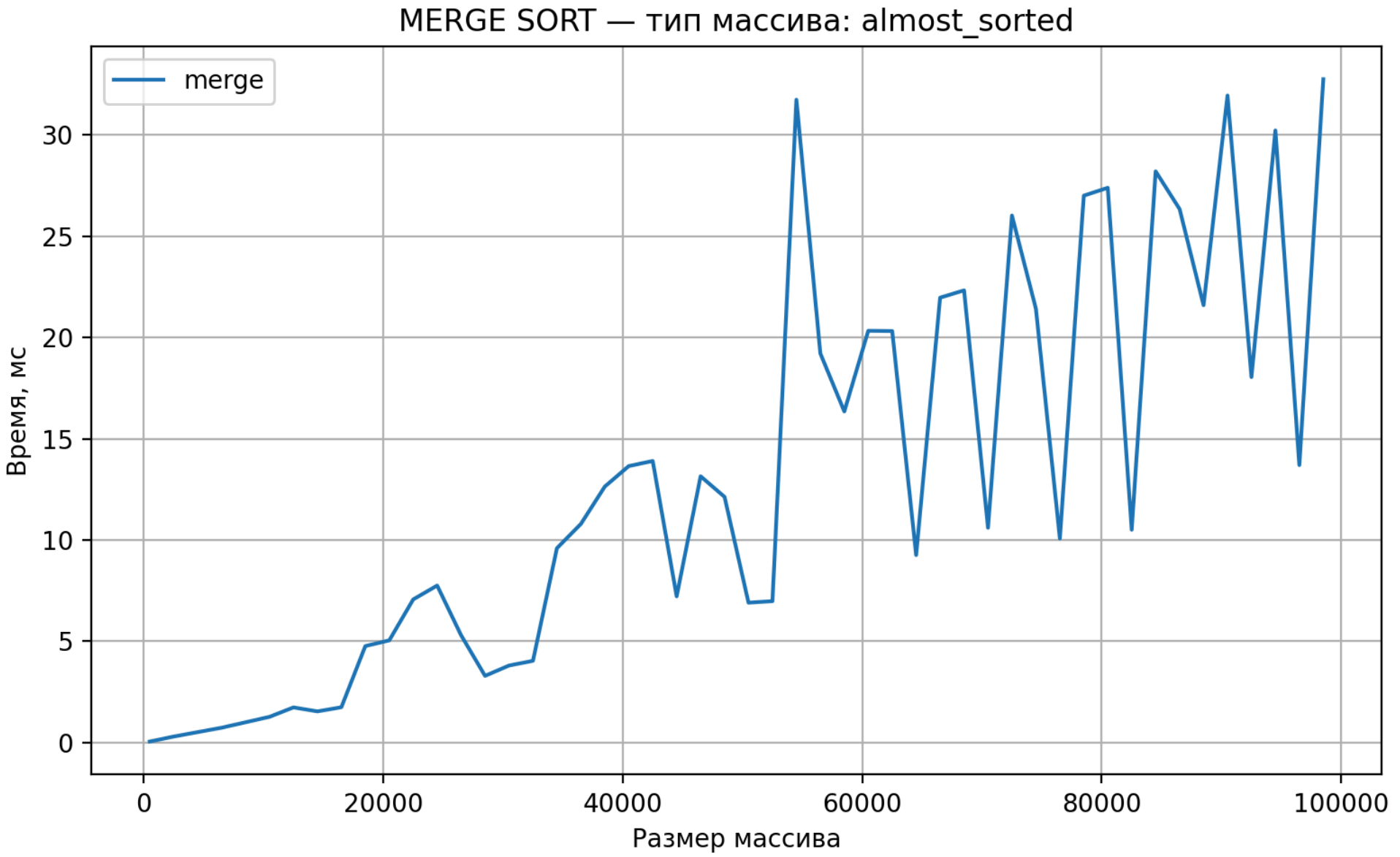
Номер посылки: 349337084
Данные: GitHub

Замечание: при построение графиков я брал точки с шагом `step = 20`. Это позволило мне избавиться от избыточного количества точек, не позволявших провести нормальный анализ. Шаг можно поменять в `graphics_A2.py` на GH.
Для каждой точки размерности для каждой сортировки алгоритм запускался несколько раз (5 раз), и на графики наносилось усреднённое время работы в миллисекундах.

Чистый Merge Sort

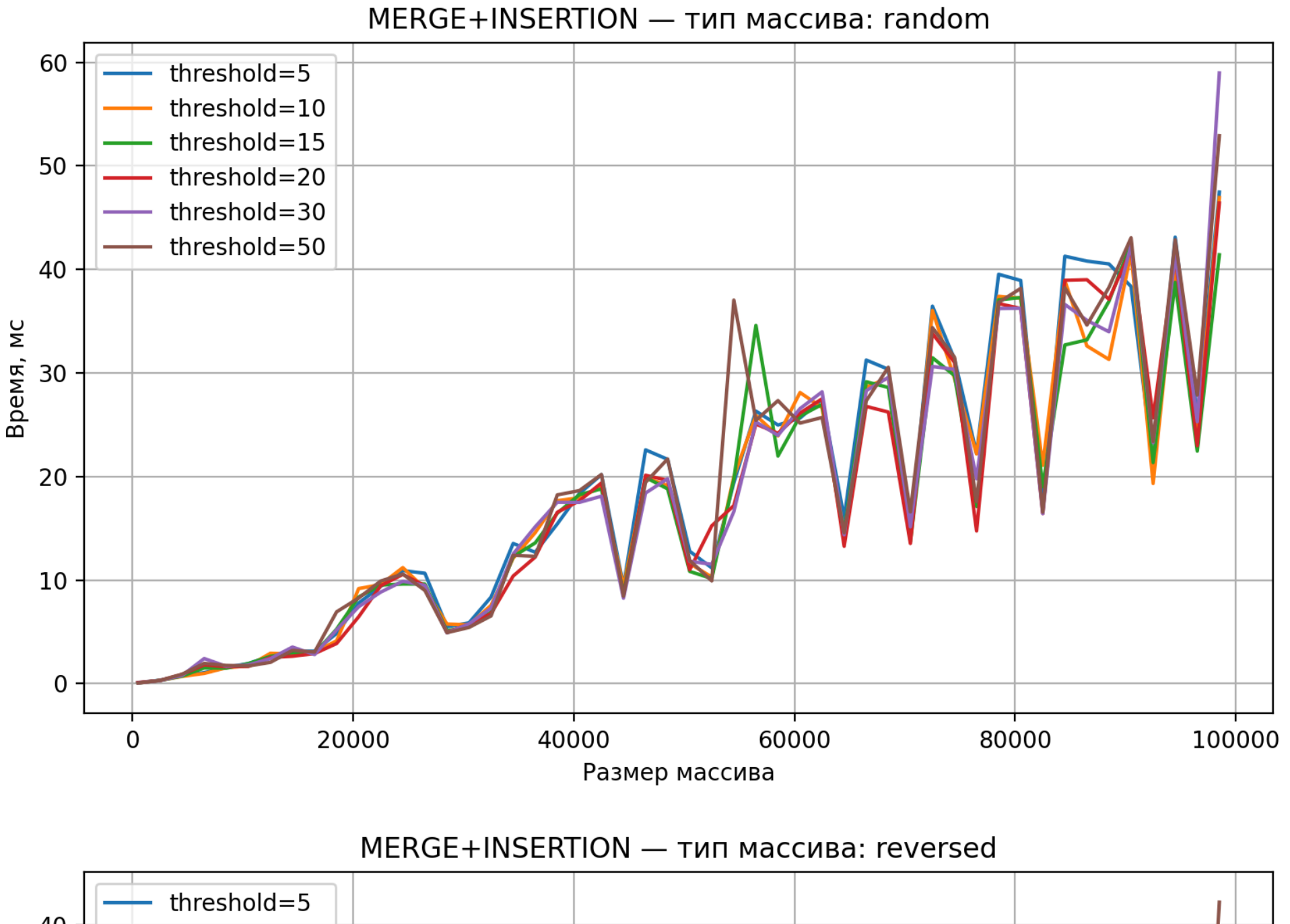

MERGE SORT — тип массива: random
merge
Время, мс
0
10
20
30
40
50
0
20000
40000
60000
80000
100000
Размер массива

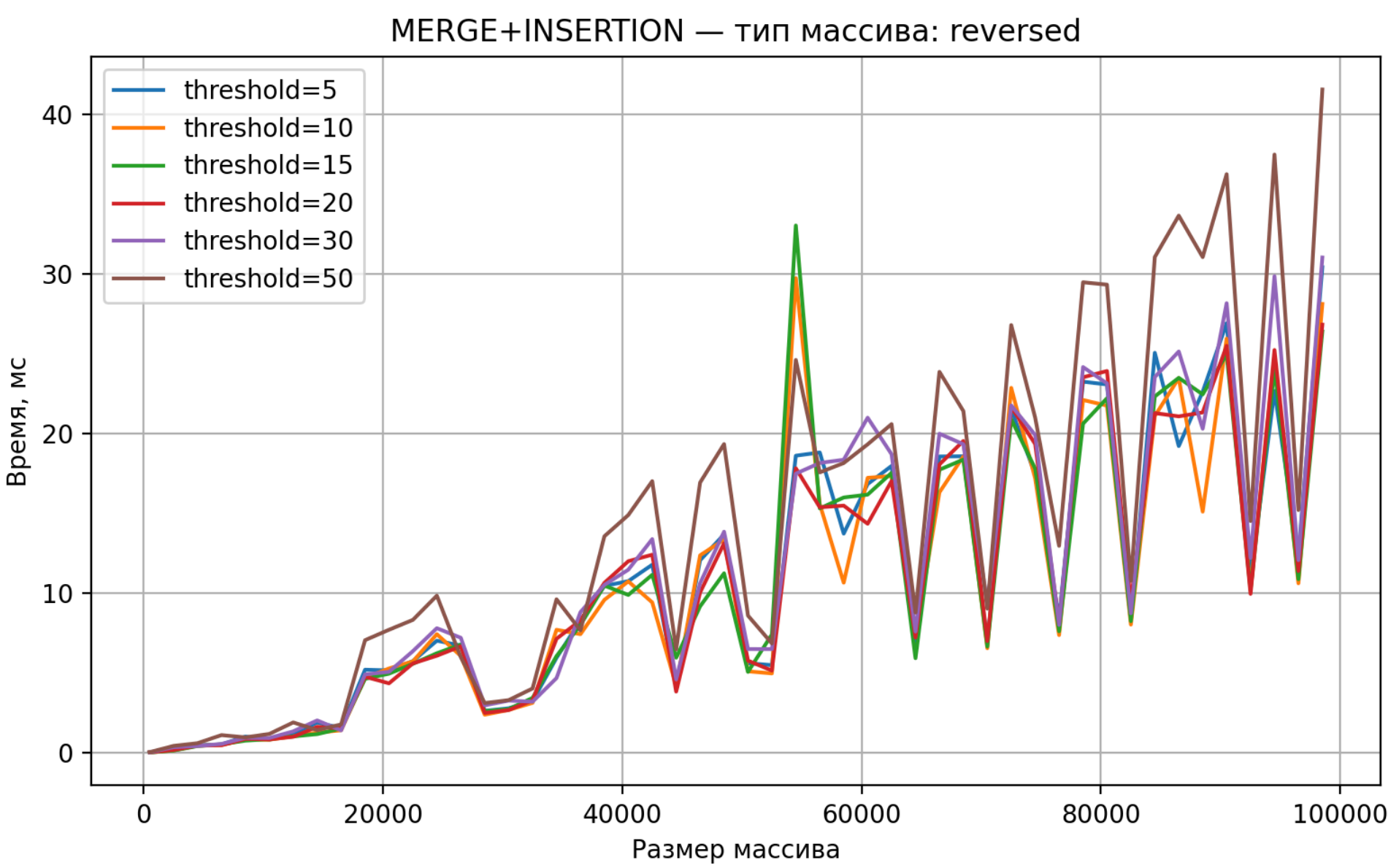

MERGE SORT — тип массива: reversed
merge
Время, мс
0
5
10
15
20
25
30
35
40
0
20000
40000
60000
80000
100000
Размер массива

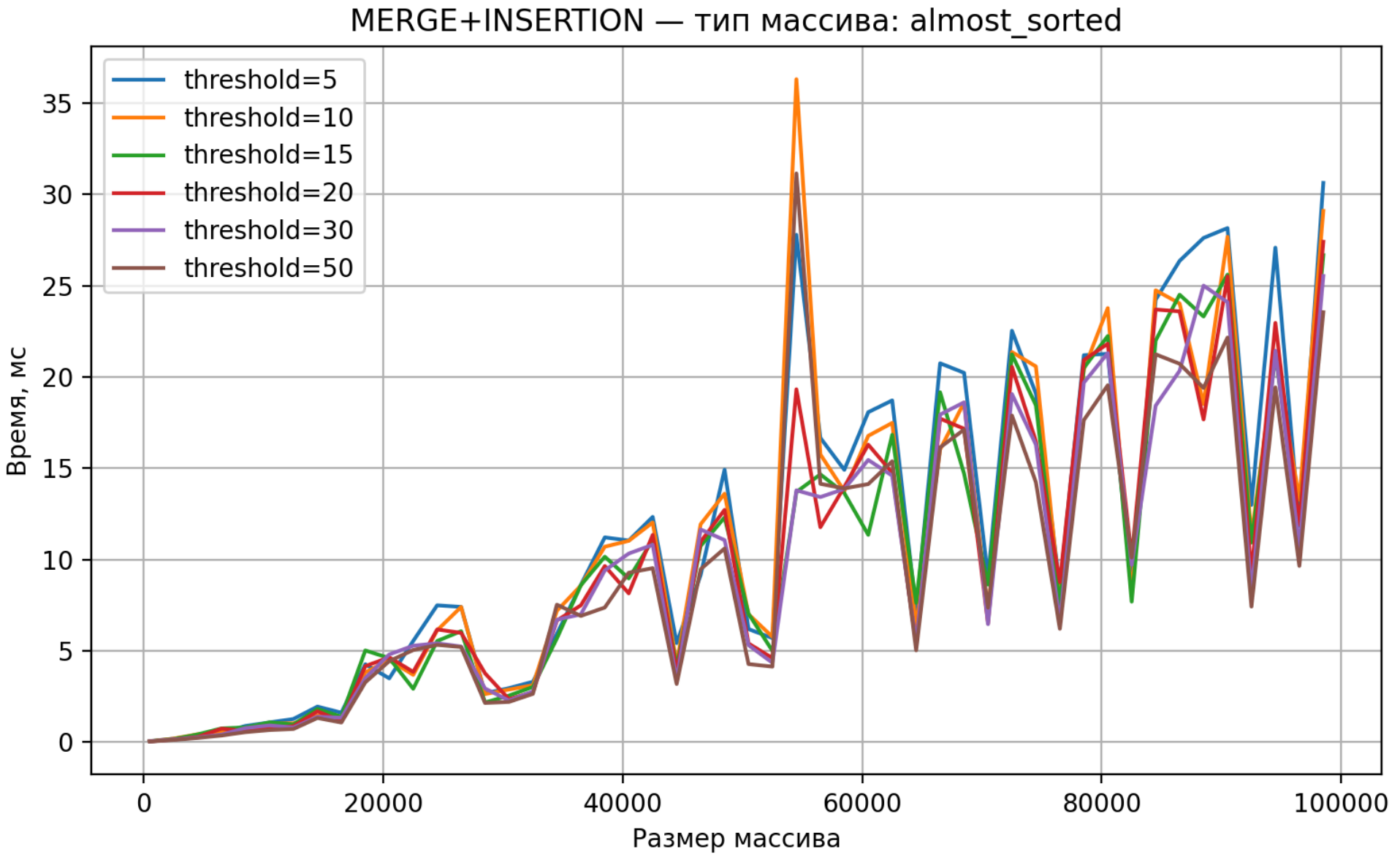


1. В reversed и almost_sorted примерно на 55000 элементов массива произошел резкий скачок во времени (мс). Однако такие отдельные резкие пики на конкретных размерах не образуют закономерностей, связанных с типом массива или размером входа.
2. На всех трёх графиках видно, что время работы монотонно растёт с увеличением размера входа. Это отражает ключевое свойство Merge Sort: время работы практически не зависит от степени упорядоченности массива. Алгоритм в любом случае полностью рекурсивно делит массив пополам и выполняет одинаковое количество операций слияния независимо от расположения элементов.

Merge + Insertion

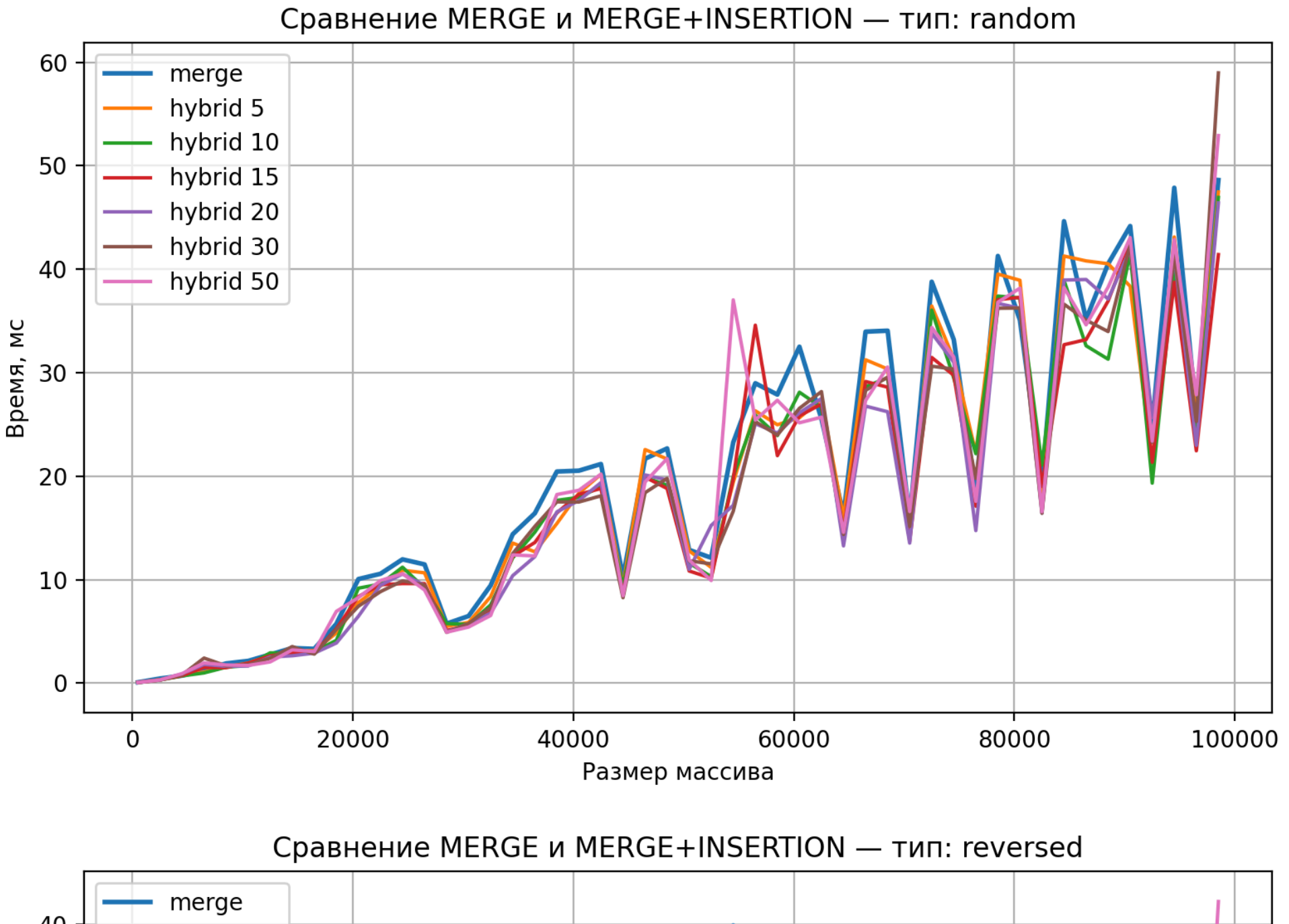

MERGE+INSERTION — тип массива: random
threshold=5
threshold=10
threshold=15
threshold=20
threshold=30
threshold=50
Время, мс
0
10
20
30
40
50
60
0
20000
40000
60000
80000
100000
Размер массива

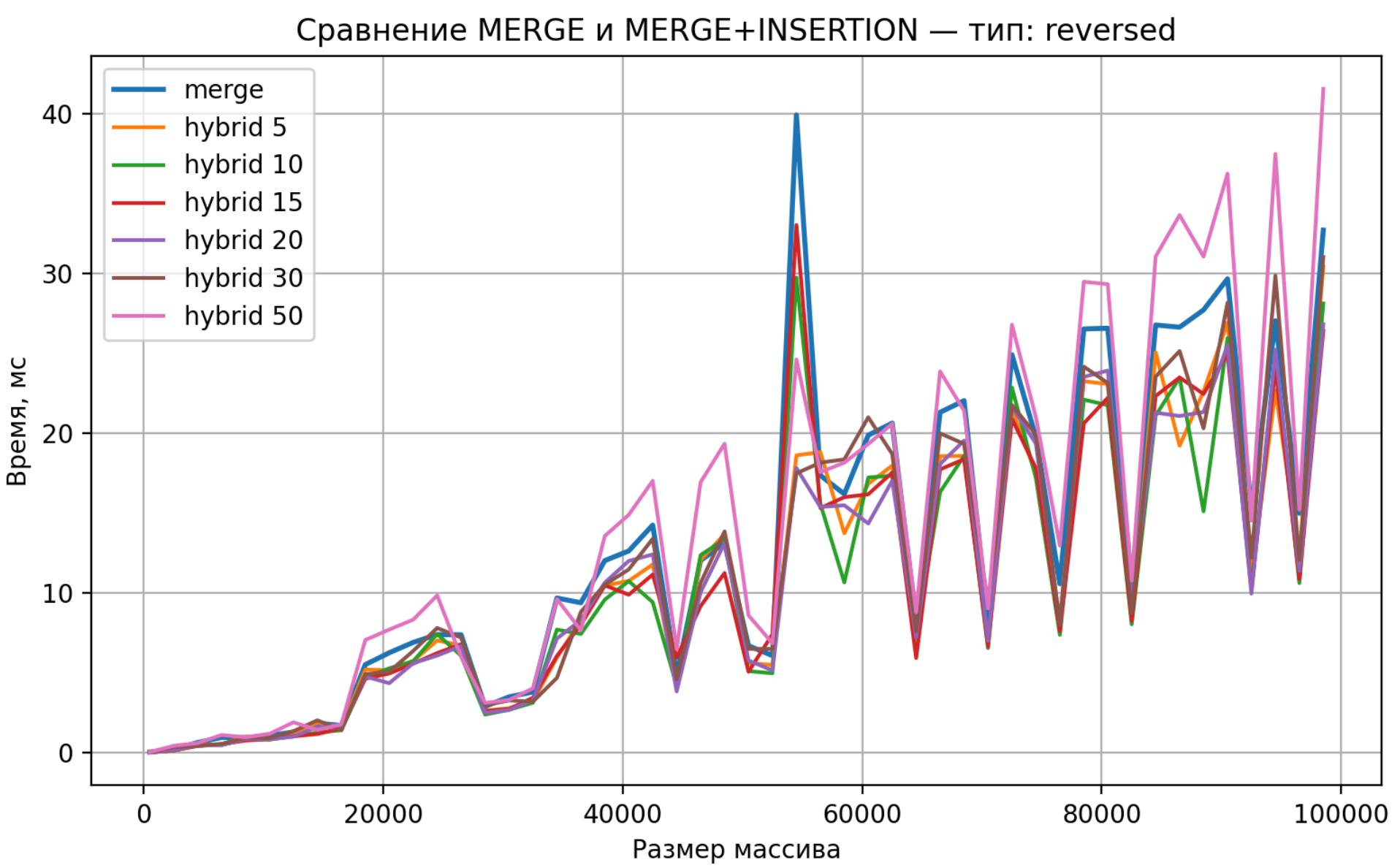

MERGE+INSERTION — тип массива: reversed
threshold=5
threshold=10
threshold=15
threshold=20
threshold=30
threshold=50
Время, мс
0
10
20
30
40
0
20000
40000
60000
80000
100000
Размер массива

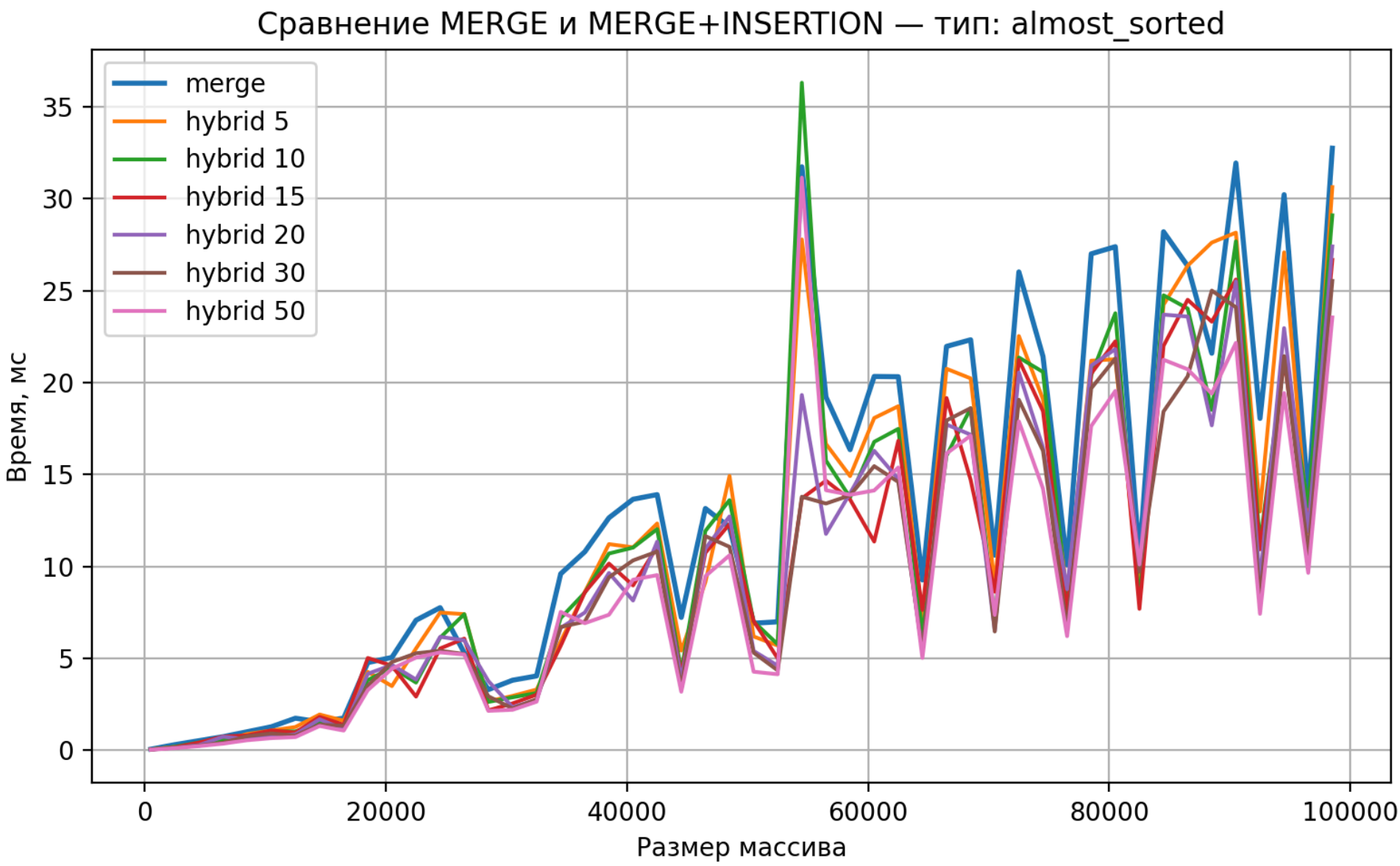


1. На reversed явно видно, что большой threshold сказывается негативно на времени работы - кривая `threshold = 50` почти на всем отрезке находится над остальными. За ней следует `threshold = 30`. Это логично: чем больше `threshold`, тем крупнее подмассивы сортируются квадратичным Insertion Sort, и тем больше его вклад в общее время.
2. В остальном поведение кривых на разных размерах массива при разных значениях параметра threshold практически идентичное (за исключением резких скачкой при $N \approx 10^5$, что я отношу к шуму измерений).
3. Отдельные резкие пики на некоторых размерах (например, локальные всплески у отдельных threshold) не образуют систематической картины и, по-видимому, связаны с шумом измерений.
4. Хочется отметить, что самую стабильную работу (независимость от шумов) показала кривая с `threshold = 15`.

Сводная


Сравнение MERGE и MERGE+INSERTION — тип: random
merge
hybrid 5
hybrid 10
hybrid 15
hybrid 20
hybrid 30
hybrid 50
Время, мс
0
10
20
30
40
50
60
0
20000
40000
60000
80000
100000
Размер массива


Сравнение MERGE и MERGE+INSERTION — тип: reversed
merge
hybrid 5
hybrid 10
hybrid 15
hybrid 20
hybrid 30
hybrid 50
Время, мс
0
10
20
30
40
0
20000
40000
60000
80000
100000
Размер массива



1. random
   - На малых размерах (до ~20 тысяч элементов) все кривые практически совпадают: и чистый Merge Sort, и гибрид при любых значениях `threshold` дают близкое время.
   - На остальном отрезке оси (кроме $N \approx 10^5$) при любых значениях параметра threshold `Merge + Insertion` справляется чуть-чуть быстрее, чем обычный `Merge` за исключением короткого отрезка при $N \approx 50000$.
2. reversed
   - Картина похожа на random: на малых размерах различий между алгоритмами практически нет.
   - Для `threshold` 10–20 гибрид по большей части лучше стандартного MERGE.
   - Для `threshold = 30` выигрыш уже неочевиден: кривая гибрида колеблется вокруг кривой MERGE.
   - Для `threshold = 50` гибрид заметно деградирует: линия hybrid 50 почти везде выше линии MERGE и , то есть гибрид начинает работать стабильнее медленнее стандартной реализации.
3. almost_sorted
   - На почти отсортированных массивах эффект от гибридизации выражен сильнее всего. Это объясняется тем, что Insertion Sort особенно эффективен на почти отсортированных данных.
   - При любых значениях threshold из множества `Merge + Insertion` исполняется быстрее `Merge`, за исключением несистематических скачков.

Пороговое значение threshold

1. При дальнейшем увеличении порога, примерно начиная с значений порядка 30 и выше, выигрыш исчезает:
   - при `threshold = 30` гибрид работает на одном уровне с MERGE или немного хуже;
   - при `threshold = 50` гибрид на больших размерах массивов систематически уступает стандартной реализации, особенно на обратно отсортированных данных.
2. Таким образом, можно считать, что «пороговым» значением параметра является интервал около 30 элементов: при больших значениях гибридный алгоритм в среднем начинает работать медленнее, чем чистый Merge Sort.