A2

Пётр Лопаткин

23 ноября 2024 г.

1 Введение

В рамках данной задачи был разарботан класс **ArrayGenerator**, для автоматической генерации тестов. Его можно найти в соответствующем файле внутри репозитория. Параметры генерации использованы согласно заданию, но было проигнорировано условие про выделение подмассивов меньшего размера из большего, т.к в этом случае данные получались недостаточно случайными(на мой взгляд). Для самого тестирования был разработан класс **SortTester**, который также можно найти внутри репозитория. Также важно уточнить что я проводил анализ для разных threshold, и на графиках будут указаны mergeSort с разными threshold. Это сделано с целью чтобы проверить как на одних и тех же данных ведут себя разные сортировки. То есть сортировки с одинаковыми threshold сортировали абсолютно эдентичные данные. В результате тестирования гибридного алгоритма сортировки и классического алгоритма сортировки merge sort было выявлено следующее.

2 Эмпирический анализ

В ходе прогона тестов были получены следующие результаты:

2.1 Рандомно сгенерированные массивы

На рисунке 1 приведены результаты тестрования гибридного алгоритма сортировки и классического mergeSort. Тренд графиков выглядит линейно, т.к координаты расположены не равномерно, но на результаты анализа не влияет. На графике легко увидеть, что все вариации гибридной сортировки работают быстрее чем любая из вариаций mergeSort. Также можно увидеть зависимость, что гибридная сортировка при threshold $>= 20$ начала проседать по времени. Что показывает, что при большом threshold наша сортировка переходит в большую асимптотику, что логично. Однако это все ещё заметно быстрее классического mergeSort.

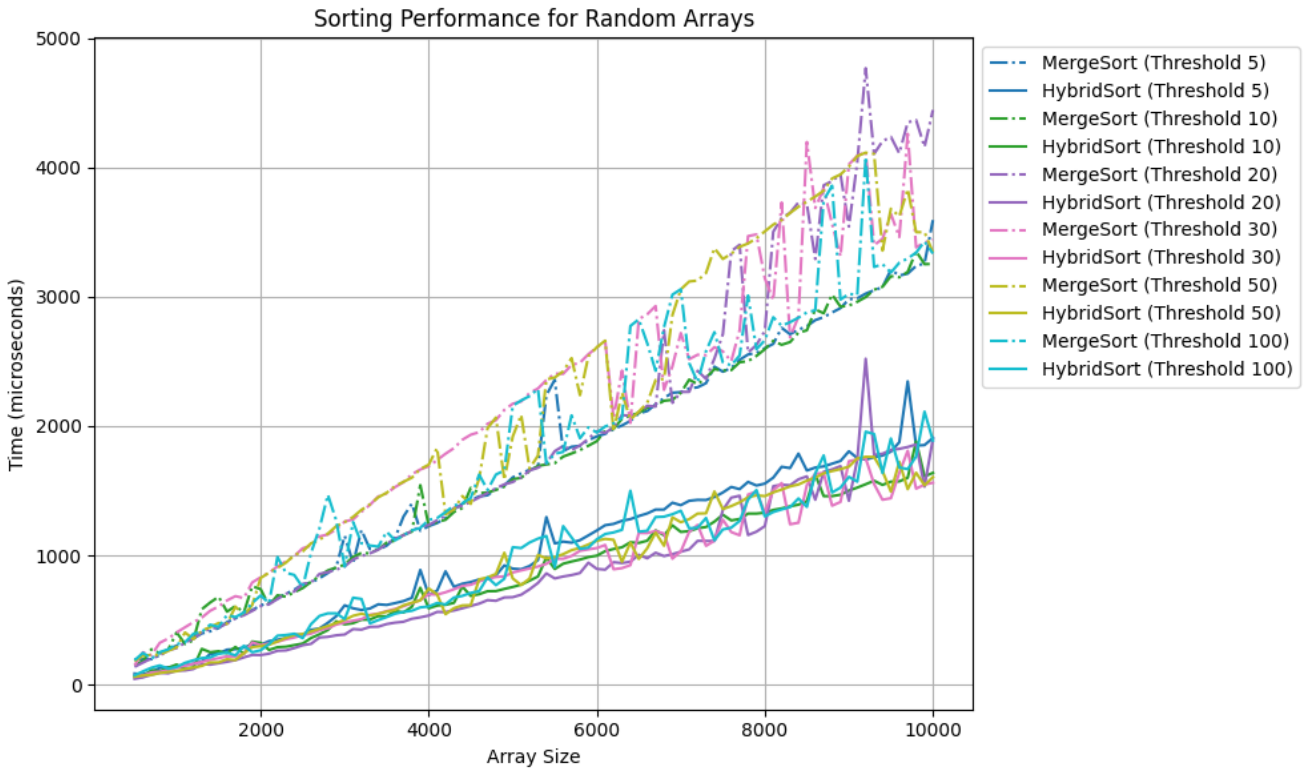

Рис. 1. Скорость сортировки рандомно сгенерированных массивов

2.2 обратно отсортированные массивы

На рисунке 2 приведены результаты тестрования гибридного алгоритма сортировки и классического mergeSort. Тренд графиков выглядит линейно, т.к координаты расположены не равномерно, но на результаты анализа не влияет. На графике можно увидеть, что местами гибридная сортировка при threshold равном 100 начинает сильно уступать скорости mergesort, что довольно очевидно, т.к сортировка вставками в данном случае работает всегда за $O(N^2)$, следовательно асимптотика должна ухудшаться на фоне mergeSort, который всегда будет работать за $O(Nlog(N))$. При маленьком threshold сортировка работает значительно быстрее, т.к мы усложняем вычисления незначительно, на $N = 10, 50...$ шага рекурсии. Также на этом графике отрыв гибридной сортировки не так значителен, как во всех других тестах, что логично, т.к вычисления мы только усложняем.

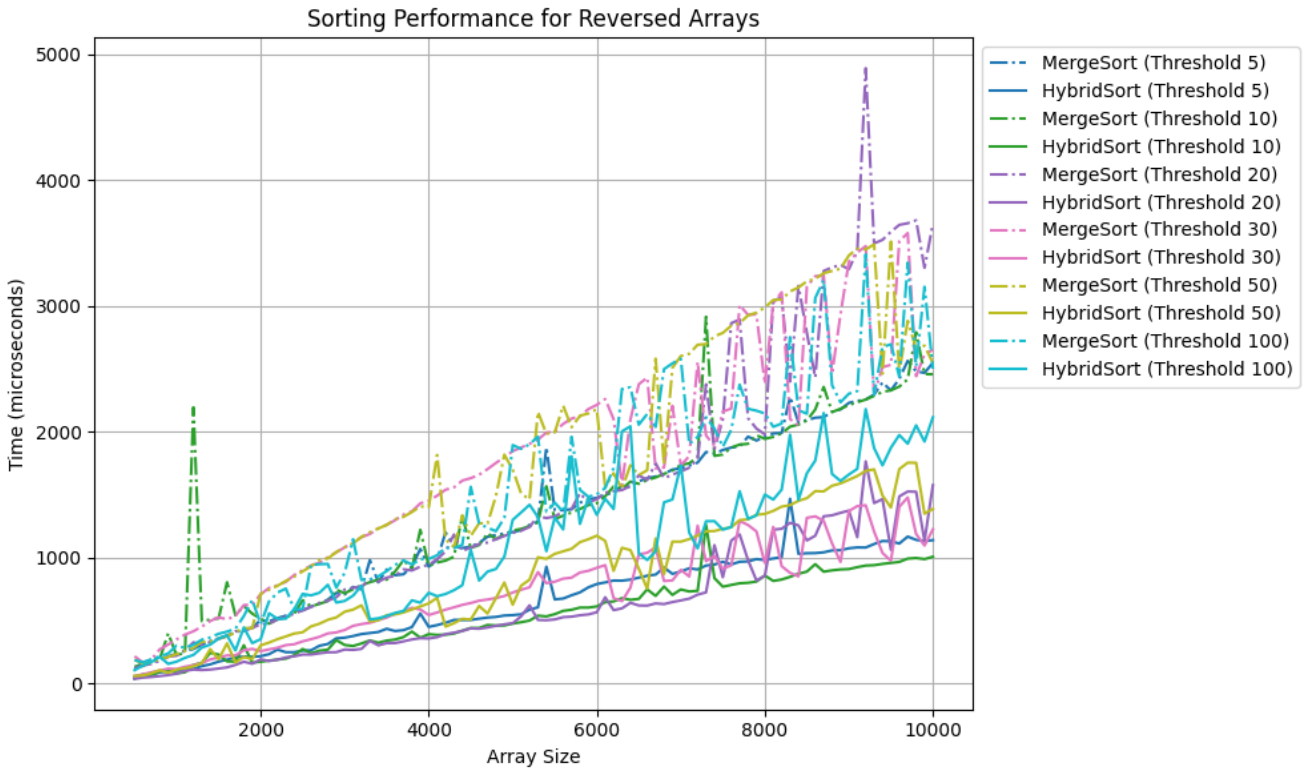

Рис. 2. Скорость сортировки обратно отсортированных массивов.

2.3 Почти отсортированные массивы

На рисунке 3 риведены результаты тестрования гибридного алгоритма сортировки и классического mergeSort. Тренд графиков выглядит линейно, т.к координаты расположены не равномерно, но на результаты анализа не влияет. Тут также гибридная сортировка обходит классическую, причем заметно сильнее, чем в предыдущих тестах. Это логично, т.к мы сокращаем несколько шагов рекурсии и проделываем их за $O(threshold)$. Однако тут уже видно, что при большем threshold сортировка работает быстрее чем при меньшем, что сильно отличает эти тесты от всех предыдущих. Это опять же довольно логично, исходя из того что мы по сути сортируем большую часть почти за линию.

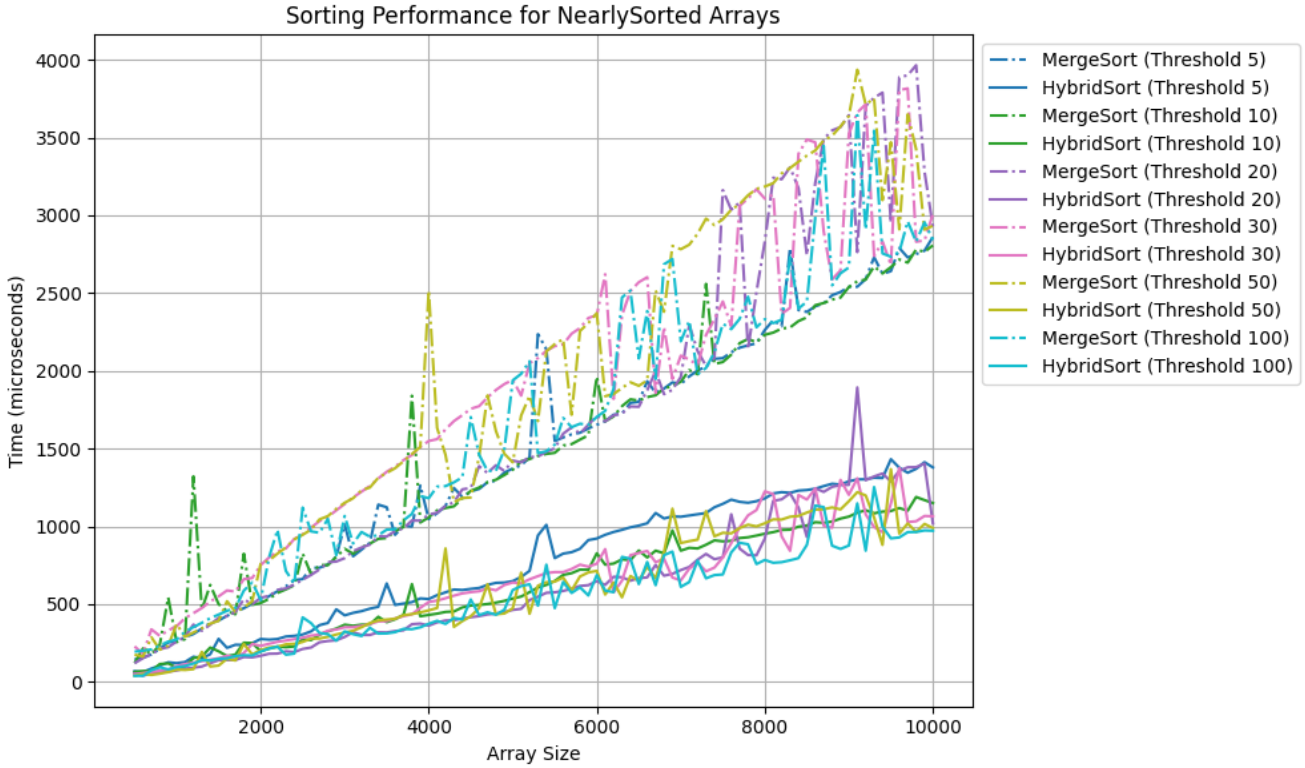

Рис. 3. Скорость сортировки почти отсортированных массивов

3 Заключение

По итогу гибридная сортировка работает значительно быстрее по времени, однако вопрос её применимости остается открытым, т.к мы не провели тестов памяти. Есть гипотеза, что возможно по памяти это решение будет сильно проигрывать обычному mergeSort. Также стоит отметить, что threshold не имеет смысла больше 50, т.к в таком случае мы начинаем уже терять в скорости, на фоне меньших значений threshold значительно.

4 Данные:

ID посылки: 292394315
Реализации классов: реализации
Ссылка на репозиторий со всеми данными репозиторий