Отчет A2

Обе реализации обладают теоретической асимптотикой O(n · log n), однако различаются по величине скрытых констант и накладным расходам, связанным с рекурсией и выделением временных буферов.

Результаты измерений

1. Массивы со случайными значениями

На графике random_comparison.png приведено сравнение времени сортировки случайных массивов.
Кривая алгоритма Merge Sort + Insertion Sort (красная) стабильно располагается ниже стандартного Merge Sort (синяя) во всём диапазоне размеров.

Среднее ускорение гибридного варианта составляет около 39 %**,** что объясняется:

- снижением числа рекурсивных вызовов для малых подмассивов;
- уменьшением накладных расходов на выделение временных векторов;
- лучшей локальностью памяти Insertion Sort для коротких блоков.

На случайных данных Merge Sort + Insertion Sort стабильно превосходит стандартный Merge Sort начиная уже с n ≈ 500 и до n = 100 000.

2. Обратно отсортированные массивы

График reverse_sorted_comparison.png показывает аналогичную зависимость. Гибридный алгоритм демонстрирует ещё более выраженное преимущество. Около 43 % ускорения по сравнению со стандартным.

Это объясняется тем, что Insertion Sort хоть и имеет квадратичную сложность, но на коротких полностью упорядоченных участках работает линейно, а также тем, что Merge Sort в таких случаях вынужден выполнять больше слияний, чем на случайных данных.

3. Почти отсортированные массивы

На графике (almost_sorted_comparison.png) алгоритм Merge Sort + Insertion Sort вновь показывает лучшие результаты (около 46 % ускорения), однако наблюдается один значительный выброс во времени стандартного Merge Sort (при n ≈ 50 000).
Этот выброс объясняется, вероятно, внешним системным фактором - временной приостановкой процесса или задержкой при записи файла результатов.

На графике time_complexity_analysis.png отображена зависимость времени сортировки от n в логарифмической шкале: обе реализации растут пропорционально n · log n, что подтверждает теоретическую сложность.
Различие между кривыми — следствие различия констант.

В данных измерениях в алгоритме Merge Sort + Insertion Sort использовался threshold = 15.

При проведении экспериментов, связанных с изменением порога, выявила, что его оптимальное значение находится между 10 и 20. При дальнейшем увеличении threshold время работы начинает расти, так как Insertion Sort выполняется на слишком больших подмассивах.

Выводы

- Гибридная реализация Merge+Insertion Sort значительно эффективнее стандартного рекурсивного Merge Sort во всех исследованных сценариях.

- Среднее ускорение составило от ≈ 39 % до ≈ 46 % в зависимости от характера данных.

- Оптимальный порог перехода на Insertion Sort — 15 элементов, что подтверждено экспериментально.

- Асимптотическая сложность обоих алгоритмов остаётся O(n log n), но Merge+Insertion Sort имеет меньшую константу и меньшие накладные расходы.

- Для практического применения лучше использовать вариант Merge+Insertion Sort с порогом в 10–20 элементов.

ID посылки на CodeForces: 348455316