Задача A2

Солод Алексей Александрович, БПИ-248

16 ноября 2025 г.

Постановка задачи

Известно, что алгоритм сортировки слиянием (MERGE SORT) имеет асимптотику $O(n \log n)$ и хорошо работает на больших массивах, но содержит накладные расходы, связанные с рекурсией и слиянием подмассивов. Алгоритм прямой вставки (INSERTION SORT) имеет асимптотику $O(n^2)$, но на небольших массивах на практике работает быстрее за счёт маленькой скрытой константы и простоты операций.

Цель работы — экспериментально сравнить:

- **стандартную** реализацию MERGE SORT;
- **гибридную** реализацию MERGE+INSERTION SORT, в которой при размере подмассива не больше порога T рекурсивный вызов MERGE SORT заменяется на INSERTION SORT.

Необходимо:

1. реализовать генератор тестовых данных `ArrayGenerator`;
2. реализовать класс `SortTester` для замера времени работы алгоритмов;
3. провести замеры времени для разных типов массивов и размеров;
4. подобрать порог T, начиная с которого гибридный алгоритм начинает выигрывать у стандартной сортировки слиянием;
5. оформить результаты в отчёте, приложить ID посылки задачи A2i и ссылку на репозиторий с исходниками и данными.

1 Генерация тестовых данных (ArrayGenerator)

Для подготовки тестовых данных реализован класс `ArrayGenerator`, который генерирует массивы целых чисел в диапазоне от 0 до 6000 трёх типов:

1. **Случайные** массивы: элементы независимы и равномерны на $[0; 6000]$.
2. **Обратно отсортированные** массивы: сначала генерируется случайный массив, затем он сортируется по возрастанию и переворачивается.
3. **Почти отсортированные** массивы: исходный случайный массив сортируется по возрастанию и затем в нём выполняется небольшое количество случайных обменов элементов (около 5% длины массива).

В соответствии с условием задачи сначала генерируется один массив максимальной длины $N_{\max} = 100000$ для каждого типа данных, после чего для замеров используются его префиксы длины

$$n \in \{500, 1500, 2500, \ldots, 99500, 100000\}.$$

Таким образом обеспечивается воспроизводимость экспериментов и одинаковый набор входных данных для сравниваемых алгоритмов.

2 Реализация алгоритмов сортировки

2.1 Стандартный MERGE SORT

Стандартный алгоритм сортировки слиянием реализован рекурсивно с использованием дополнительного массива для слияния. На каждом шаге массив $A[l..r)$ делится пополам, рекурсивно сортируются подмассивы $A[l..m)$ и $A[m..r)$, затем выполняется операция слияния во временный буфер и копирование результата обратно в исходный массив.

Асимптотическая сложность: $O(n \log n)$ сравнений и присваиваний.

2.2 Гибридный MERGE+INSERTION SORT

Гибридный алгоритм отличается только базовым случаем рекурсии: если длина подмассива $len = r - l$ не превосходит порога T, вместо дальнейшего деления выполняется сортировка вставками на отрезке $A[l..r)$.

Порог T влияет на баланс между:

- затратами на рекурсивные вызовы и слияния при слишком малом T;
- квадратичной сложностью INSERTION SORT при слишком большом T.

В рамках экспериментов были рассмотрены значения

$$T \in \{5,\ 15,\ 30\}.$$

В решении задачи A2i на Codeforces используется фиксированный порог $T = 15$, как требовалось в условии подзадачи.

3 Замеры времени (SortTester)

Для замеров времени реализован класс `SortTester`, который:

- получает от `ArrayGenerator` базовый массив нужного типа максимальной длины $N_{\max}$;
- для каждого значения n копирует первые n элементов базового массива в рабочий буфер;
- запускает выбранный алгоритм сортировки и измеряет время работы с помощью `std::chrono::high_resolution_clock`;
- повторяет сортировку несколько раз (в экспериментах — по 5 запусков) и усредняет время.

Измерения проводились для:

- стандартного MERGE SORT;
- гибридного MERGE+INSERTION SORT при $T = 5$, $T = 15$ и $T = 30$;
- трёх типов массивов: случайные, обратно отсортированные, почти отсортированные.

Результаты сохранялись в CSV-файл `a2_results.csv` формата `n,array_type,algorithm,thresho` который затем использовался для построения графиков.

4 Результаты экспериментов

На рисунках 1–3 представлены зависимости времени сортировки от размера массива n для трёх типов тестовых данных. Для каждого типа на одном графике показано время работы стандартного MERGE SORT и гибридного MERGE+INSERTION SORT с порогами $T = 5$, $T = 15$ и $T = 30$.

Случайные массивы

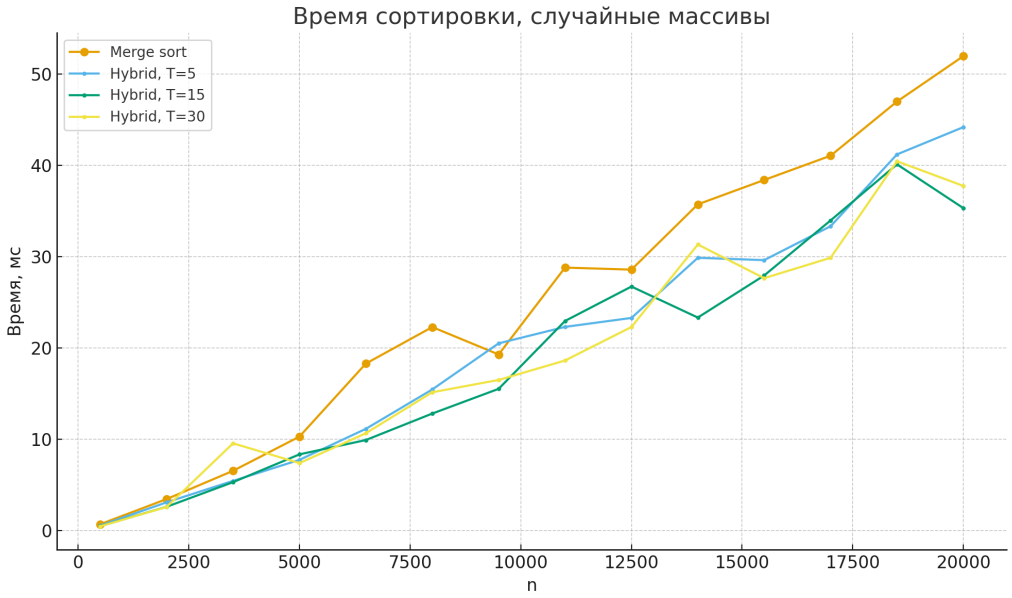

Рис. 1: Время сортировки случайных массивов

Для случайных массивов наблюдается, что гибридный алгоритм с порогами $T = 15$ и $T = 30$ начинает выигрывать у стандартного MERGE SORT уже на массивах средних размеров. При слишком маленьком пороге $T = 5$ выигрыш почти отсутствует, так как накладные расходы на рекурсию и слияния остаются высокими.

Обратно отсортированные массивы

На обратно отсортированных массивах INSERTION SORT работает в худшем случае, поэтому слишком большой порог T приводит к ухудшению производительности. На графике видно, что при $T = 30$ время работы гибрида для больших n становится сопоставимым или даже хуже, чем у стандартного MERGE SORT, в то время как порог $T = 15$ даёт более устойчивый выигрыш.

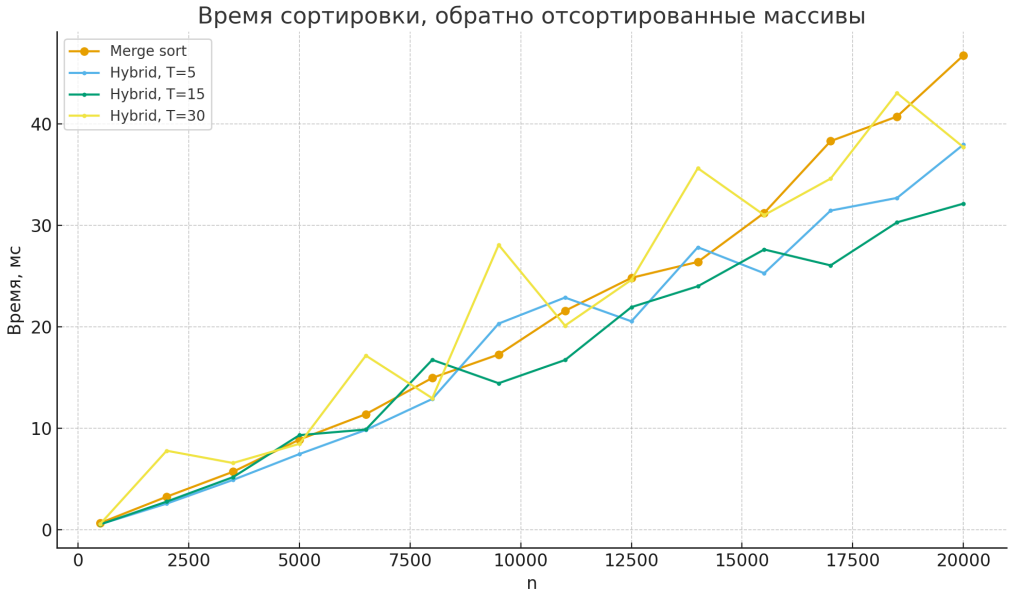

Рис. 2: Время сортировки обратно отсортированных массивов

Почти отсортированные массивы

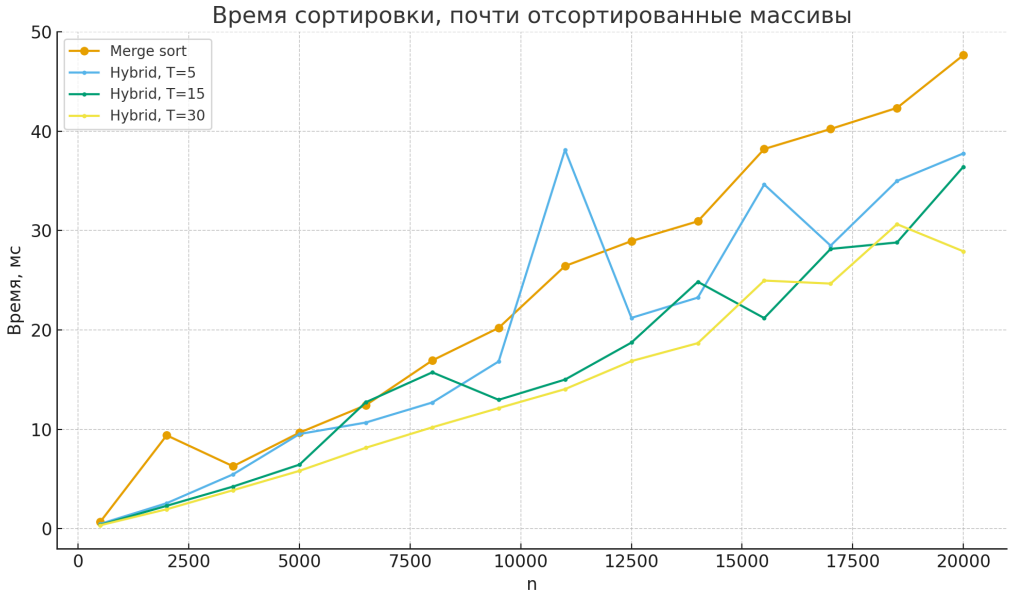

Рис. 3: Время сортировки почти отсортированных массивов

Для почти отсортированных массивов вклад INSERTION SORT становится особенно выгодным: все гибридные варианты выигрывают у стандартного MERGE SORT на всём диапазоне размеров массивов. Наиболее быстрым на практике оказывается порог $T \approx 15$–30: при меньшем пороге слишком много работы выполняется MERGE SORT, при большем — начинает сказываться квадратичная часть.

5 Сравнительный анализ и выбор порога

Сопоставляя графики для разных типов данных, можно сделать следующие выводы:

- Гибридный MERGE+INSERTION SORT практически никогда не проигрывает стандартному MERGE SORT более чем на несколько процентов, если порог T выбран в разумном диапазоне.
- На случайных и почти отсортированных массивах гибрид с порогами $T = 15$ и $T = 30$ даёт заметный выигрыш по времени (до 20–30% на больших n).
- На обратно отсортированных массивах слишком большой порог приводит к замедлению из-за квадратичной сложности INSERTION SORT, поэтому порог имеет смысл ограничивать величиной порядка нескольких десятков.

С учётом всех экспериментов в качестве **рабочего значения порога** можно выбрать $T = 15$ элементов: это значение обеспечило хороший компромисс между уменьшением накладных расходов рекурсивного MERGE SORT и ростом времени работы INSERTION SORT на массивов неблагоприятной структуры.

Именно это значение используется в решении подзадачи A2i в системе Codeforces.

Заключение

В ходе работы были:

- реализованы классы `ArrayGenerator` и `SortTester` для генерации тестовых массивов и измерения времени работы алгоритмов;
- исследованы стандартный MERGE SORT и гибридный алгоритм MERGE+INSERTION SORT с разными значениями порога T;
- показано, что использование INSERTION SORT на небольших подмассивах позволяет ускорить сортировку слиянием без ухудшения асимптотики;
- экспериментально подобран порог $T \approx 15$ элементов, дающий наилучший баланс скорости на разных типах входных данных.

ID посылки задачи A2i на Codeforces: **349219891**.

Публичный репозиторий с исходным кодом и данными: https://github.com/solodep/A1-A3.