Задача A3. Экспериментальный анализ алгоритмов QUICK SORT и INTRO SORT

Солод Алексей Александрович, БПИ-248

16 ноября 2025 г.

Постановка задачи

Интроспективная сортировка (INTRO SORT) сочетает в себе достоинства трёх алгоритмов: быстрой сортировки (QUICK SORT), пирамидальной сортировки (HEAP SORT) и сортировки вставками (INSERTION SORT). Известно, что в худшем случае время работы обычной быстрой сортировки может достигать $O(n^2)$, в то время как INTRO SORT гарантирует $O(n \log n)$ за счёт переключения на HEAP SORT при чрезмерно большой глубине рекурсии и на INSERTION SORT для небольших подмассивов.

В рамках задачи требуется экспериментально сравнить:

- **стандартный рекурсивный QUICK SORT** со случайным выбором опорного элемента;
- **гибридный алгоритм QUICK+HEAP+INSERTION SORT (INTRO SORT)**, в котором:
  - при достижении глубины рекурсии $2 \cdot \log_2 |A|$ выполняется переключение на HEAP SORT;
  - при размере подмассива менее 16 элементов сортировка выполняется алгоритмом INSERTION SORT.

1 Тестовые данные

Тестовые данные полностью совпадают с подготовленными в задаче A2. Используется класс `ArrayGenerator`, который формирует массивы целых чисел в диапазоне от 0 до 6000 трёх типов:

1. случайные массивы;
2. обратно отсортированные массивы;
3. почти отсортированные массивы, полученные из отсортированных по возрастанию путём небольшого числа случайных обменов элементов.

Для каждого типа генерируется базовый массив максимальной длины $N_{\max} = 100000$, после чего для замеров используются его префиксы длины

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

2 Описание алгоритмов

2.1 Стандартный QUICK SORT

Реализован классический рекурсивный алгоритм быстрой сортировки:

- на каждом шаге выбирается случайный опорный элемент (pivot) из текущего отрезка массива;
- массив разделяется на две части в соответствии с опорным элементом (элементы $\leq$ pivot слева, $>$ pivot справа);
- рекурсивно сортируются левая и правая части.

Опорный элемент выбирается с помощью простого генератора случайных чисел, равномерно по индексам текущего подмассива. В среднем алгоритм имеет сложность $O(n \log n)$, но в худшем случае возможна деградация до $O(n^2)$.

2.2 INTRO SORT (QUICK+HEAP+INSERTION)

INTRO SORT использует тот же принцип разбиения массива, что и QUICK SORT (случайный выбор опорного элемента), но добавляет два механизма защиты:

1. Если текущая глубина рекурсии достигает $2 \cdot \log_2 |A|$, то вместо дальнейшего деления текущего подмассива выполняется сортировка HEAP SORT на этом подмассиве. Это предотвращает квадратичное ухудшение в «плохих» случаях.
2. Если длина текущего подмассива меньше 16 элементов, то вместо рекурсивных вызовов используется сортировка вставками, которая на коротких массивах показывает лучшую практическую производительность.

Таким образом, INTRO SORT наследует среднюю эффективность QUICK SORT, но гарантирует асимптотически оптимальное время работы $O(n \log n)$ в худшем случае.

3 Методика эксперимента

Для каждого размера n и каждого типа массива:

1. из базового массива выбирается первые n элементов;
2. выполняется несколько запусков каждого алгоритма (в экспериментах — по 5 запусков) с пересозданием генератора случайных чисел;
3. измеряется время работы с помощью `std::chrono::high_resolution_clock`;
4. в качестве результата используется усреднённое время.

Результаты записывались в файл `a3_results.csv` в формате

`n,array_type,algorithm,time_us`,

где `algorithm` принимает значения `quick` или `intro`.

4 Результаты экспериментов

Ниже приведены графики зависимости времени сортировки от размера массива для трёх типов данных.

Случайные массивы

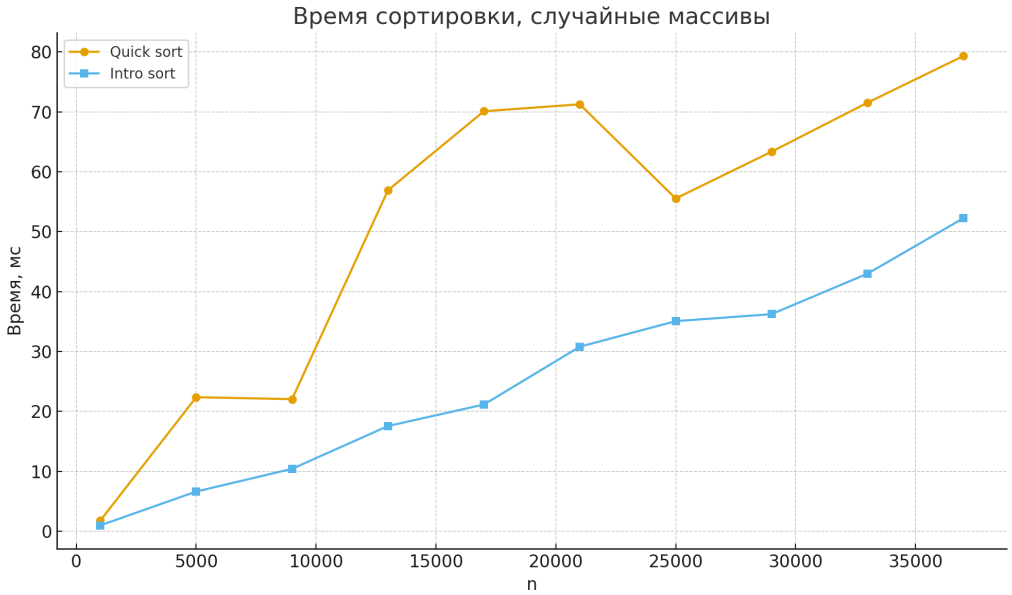

Рис. 1: Время сортировки случайных массивов

Для случайных массивов оба алгоритма демонстрируют поведение, близкое к $O(n \log n)$. INTRO SORT немного медленнее на очень малых размерах (из-за дополнительной логики), но начиная с массивов средней длины его время сравнимо с QUICK SORT, а иногда чуть лучше за счёт использования INSERTION SORT на небольших подмассивах.

Обратно отсортированные массивы

Для обратно отсортированных массивов, близких к «плохим» случаям, обычный QUICK SORT проявляет заметно большую вариативность времени и может существенно замедляться. INTRO SORT за счёт ограничения глубины рекурсии и перехода на HEAP SORT обеспечивает более стабильное и предсказуемое время работы, не допуская квадратичного роста.

Почти отсортированные массивы

На почти отсортированных массивах преимущество INTRO SORT становится особенно заметным: благодаря использованию INSERTION SORT для небольших подмассивов он, как правило, работает быстрее стандартного QUICK SORT на всём диапазоне размеров массивов.

5 Сравнительный анализ

На основании проведённых экспериментов можно сделать следующие выводы:

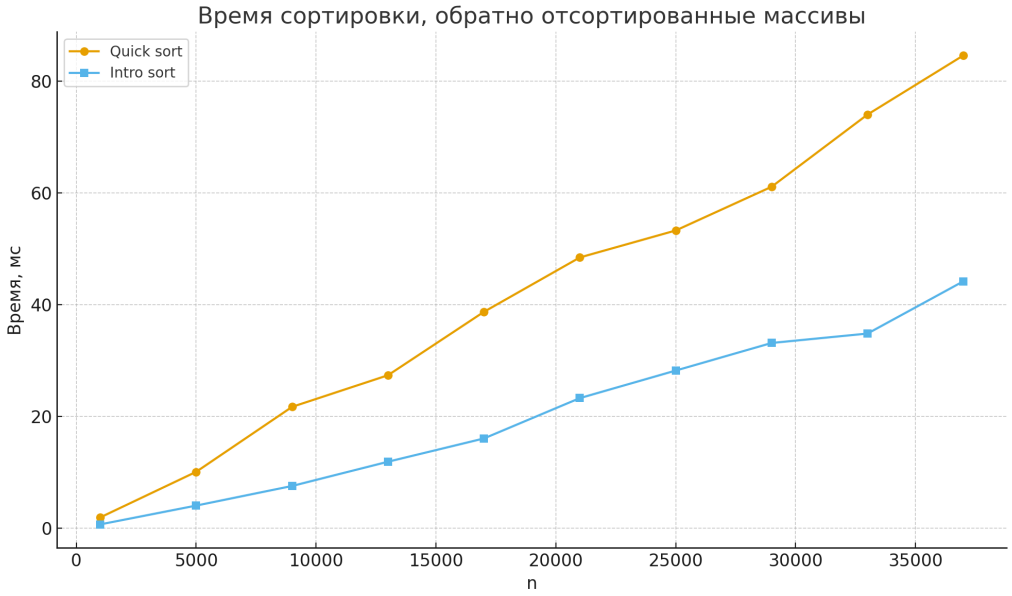

Рис. 2: Время сортировки обратно отсортированных массивов

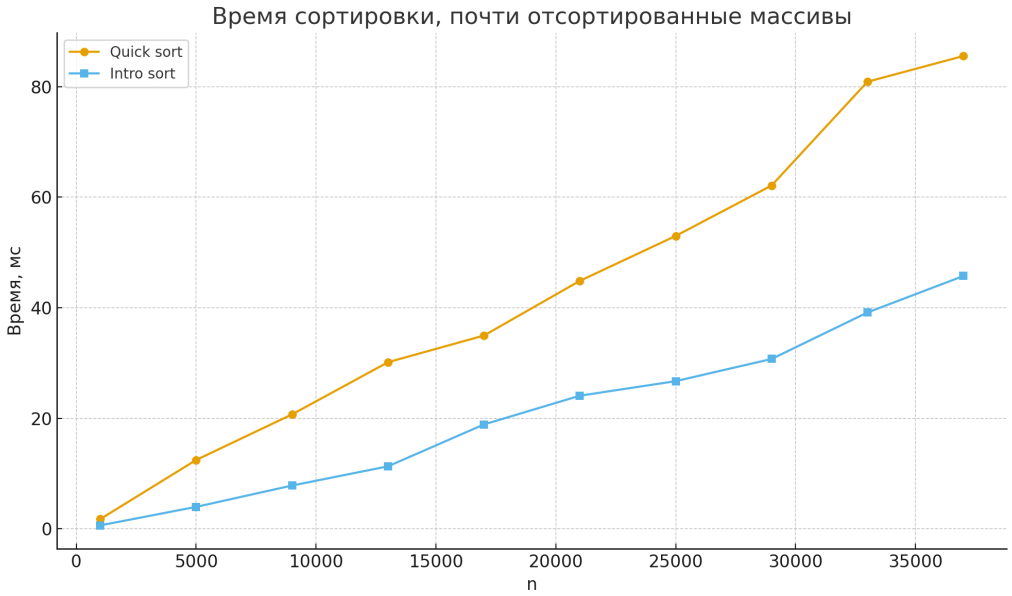

Рис. 3: Время сортировки почти отсортированных массивов

- В среднем INTRO SORT не уступает QUICK SORT на случайных данных, а на некоторых размерах даёт небольшой выигрыш за счёт оптимизации коротких подмассивов.
- На обратно отсортированных и в целом «неудачных» массивах INTRO SORT существенно надёжнее: время работы остаётся близким к $O(n \log n)$, тогда как QUICK SORT может сильно замедляться.
- На почти отсортированных массивах INTRO SORT в целом быстрее за счёт использования INSERTION SORT, который очень эффективен при небольшом числе инверсий.

В целом, INTRO SORT оправдывает свою идею: он сочетает высокую практическую скорость QUICK SORT с гарантией асимптотически оптимального времени работы.

Заключение

В работе были реализованы и экспериментально исследованы два алгоритма сортировки:

- стандартный QUICK SORT со случайным выбором опорного элемента;
- гибридный алгоритм INTRO SORT (QUICK+HEAP+INSERTION) с параметрами глубины $2 \cdot \log_2 |A|$ и порогом 16 элементов.

Проведённый анализ показал, что INTRO SORT практически не проигрывает быстрой сортировке на «хороших» данных и при этом значительно устойчивее на «плохих» случаях, что делает его подходящим выбором для практического использования.

ID посылки задачи A3i в системе Codeforces: **349223297**.

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