ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 5

«Сортировки»

Выполнил работу

Конькова Нина

Академическая группа J3113

Принято

Старший преподаватель, Ходненко И. В.

Санкт-Петербург

2024

В этом отчете мною будут рассмотрены 3 сортировки:

1. **Comb sort** (Сортировка «расческой»)
2. **Timsort** (Гибридный алгоритм сортировки, сочетающий сортировку вставками и сортировку слиянием)
3. **Introsort** (Алгоритм, использующий «быструю» сортировку и переключающийся на пирамидальную сортировку, когда глубина рекурсии превысит некоторый заранее установленный уровень)

**Comb sort**

Реализация алгоритма:

**Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание**

**Introsort**

Реализация алгоритма:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

**Timsort**

Реализация алгоритма:

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описаниеИзображение выглядит как снимок экрана, черный, темнота

Автоматически созданное описание**

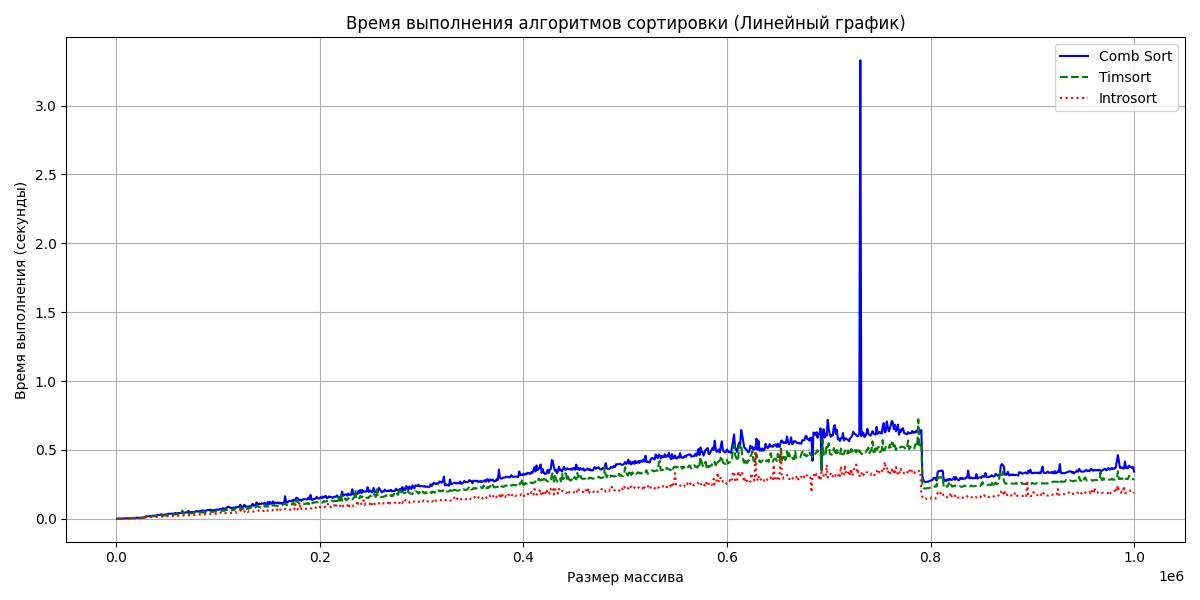
**Подсчет асимптотики**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Comb sort** | **Introsort** | **Timsort** |
| **Лучший случай** | O(n\*logn) | O(n\*logn) | O(n) |
| **Средний случай** | O(n\*logn) | O(n\*logn) | O(n\*logn) |
| **Худший случай** | O(n^2) | O(n\*logn) | O(n\*logn) |

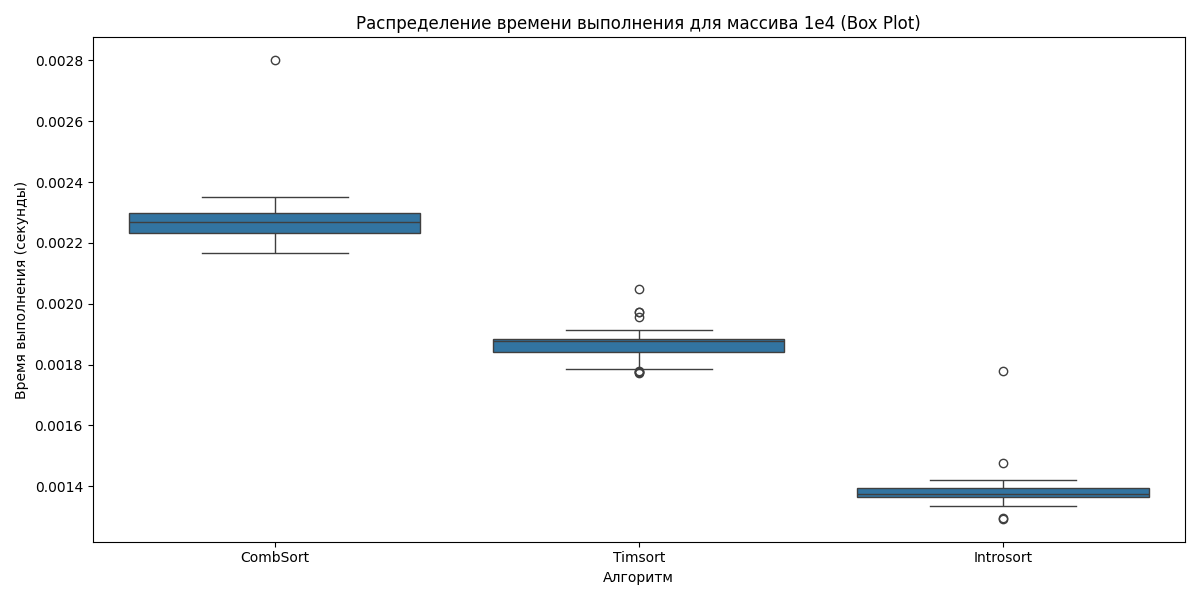
**Подсчет по памяти**

* **Comb Sort**: Это модификация пузырьковой сортировки, использующая фиксированный массив. Требует **O(1)** дополнительной памяти.
* **Introsort**: Это комбинация быстрой сортировки, пирамидальной сортировки и сортировки вставкой. Требует **O(logn)** дополнительной памяти для рекурсивных вызовов.
* **Timsort**: Основана на слиянии и вставке, использует вспомогательные массивы. Требует **O(n)** памяти.

Линейный график работы алгоритмов, для массивов от 1000 до 1e6 с шагом в 1000:



box plot график для времени работы алгоритмов с числом элементов 1e4:



box plot график для времени работы алгоритмов с числом элементов 1e5:

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

**Вывод**

**Соответствие асимптотики реальности**

* **Comb Sort:** Теоретическая асимптотика в среднем O(n\*logn), но из-за больших констант и сравнительно простого механизма алгоритм работает медленнее на практике по сравнению с более сложными алгоритмами (например, Timsort и Introsort). Это подтверждается ростом времени на линейном графике.
* **Timsort:** Асимптотика O(n\*logn) хорошо подтверждается экспериментами. Он превосходит другие алгоритмы для частично отсортированных массивов, так как оптимизирован для реальных сценариев (использует слияние и вставки).
* **Introsort:** это алгоритм с гарантированной асимптотикой O(n\*logn) за счет переключения между QuickSort, HeapSort и InsertionSort. На практике он стабильно показывает хорошее время, особенно на больших массивах, где QuickSort работает эффективно.

**Причины выбросов**

* **Comb Sort:** выбросы могут быть связаны с особенностями перестановок массива, когда шаг плохо согласуется с расположением элементов, приводя к большему числу сравнений.
* **Timsort:** небольшие выбросы могут наблюдаться на случайных массивах, где размер блоков и частота слияния не оптимальны.
* **Introsort:** выбросы могут возникнуть при редких случаях переключения на HeapSort (например, из-за глубокой рекурсии при неудачном выборе опорного элемента в QuickSort).

**Когда стоит применять эти алгоритмы сортировки?**

1. **Comb Sort:**

* Подходит для небольших массивов, где важна простота реализации.
* Можно использовать как замену BubbleSort для образовательных целей.

1. **Timsort:**

* Отлично подходит для реальных сценариев, где массивы часто частично отсортированы.
* Идеален для обработки данных, поступающих в порядке, близком к отсортированному, например, журналы событий или списки в приложениях.

1. **Introsort:**

* Универсальный выбор для массивов любого размера.
* Используется в системах, требующих надежной гарантии производительности.
* Хорош для массивов с любым распределением данных, особенно при работе с большими объемами информации.