Задача A2

Толмачева Екатерина БПИ2310

Id задачи: 292895185
https://dsahse.contest.codeforces.com/group/NOflOR1Qt0/contest/565612/submission/292895185

Ссылка на репозиторий: https://github.com/katetolmacheva/Set3.git

Рассмотрим классы программы и основную ее часть:

1. Класс ArrayGenerator - этот класс отвечает за генерацию тестовых массивов различных типов:
   1) Случайные массивы (заполнены случайными целыми числами в диапазоне от 0 до 6000)
   2) Отсортированные в обратном порядке массивы
   3) Почти отсортированные массивы (получены путём случайного обмена небольшого количества пар элементов в полностью отсортированном массиве

Класс также обеспечивает выбор подмассива необходимого размера из заранее сгенерированного массива максимального размера (10000 элементов)

2. Класс SortTester - этот класс реализует и проводит замеры времени выполнения двух сортировок:
   1) Стандартный Merge Sort (рекурсивный алгоритм сортировки слиянием)
   2) Гибридный Merge Sort + Insertion Sort (переключается на Insertion Sort для подмассивов размером 15 или меньше)

Класс также содержит функции для замера времени выполнения сортировок

3. Основная Функция main - основная функция выполняет следующие шаги:
   1) Генерация тестовых данных (использует ArrayGenerator для создания массивов различных типов)
   2) Замеры времени сортировок (спользует SortTester для сортировки массивов стандартным Merge Sort и гибридным Merge Sort + Insertion Sort, замеряя время выполнения)
   3) Сохранение результатов (записывает результаты в CSV-файл для последующего анализа и построения графиков)

Итог: программа будет генерировать массивы различных типов и размеров, сортировать их двумя методами, измерять время выполнения и сохранять результаты в файл results.csv. Этот файл можно использовать для построения графиков и дальнейшего анализа.

Построение графиков: используем код на Python и библиотеку matplotlib (код лежит на github)

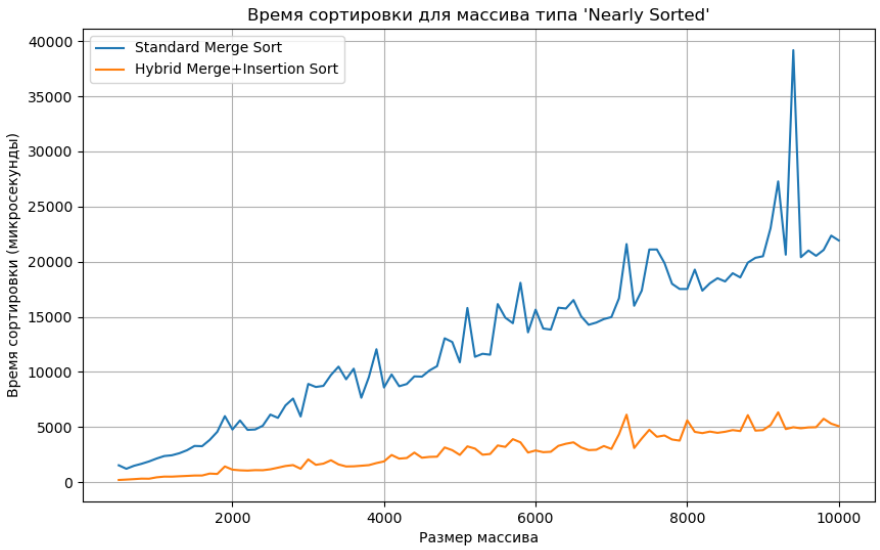
Графики Времени Сортировки:

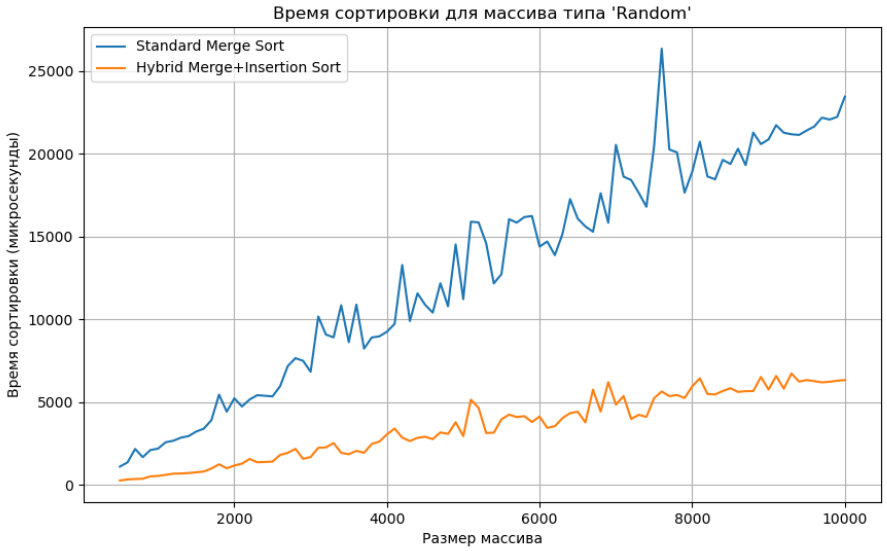
Для каждого типа массива (Random, Reverse Sorted, Nearly Sorted) строится отдельный график.

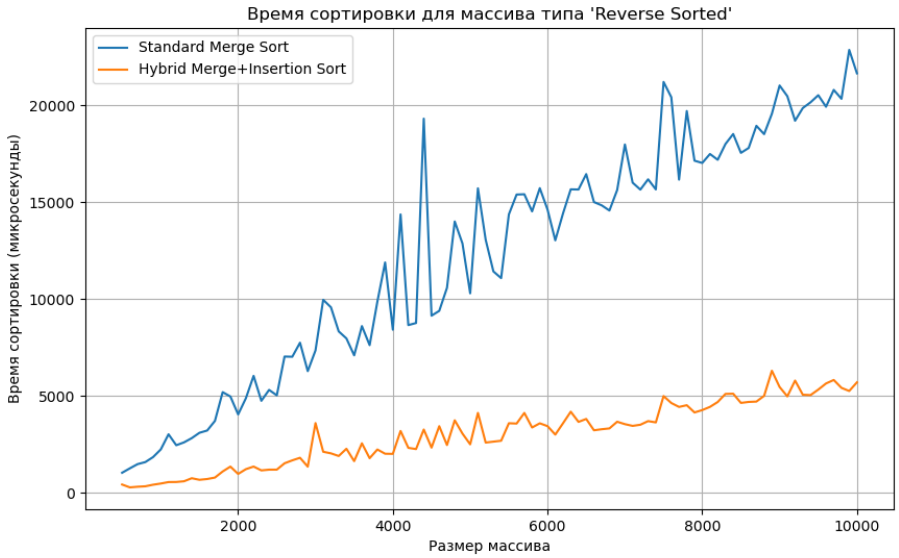
На графике отображаются две линии:

1. Standard Merge Sort (время выполнения стандартного Merge Sort)
2. Hybrid Merge+Insertion Sort (время выполнения гибридного алгоритма)

Это позволяет визуально сравнить эффективность двух алгоритмов для разных типов массивов.


Время сортировки для массива типа 'Nearly Sorted'
Standard Merge Sort
Hybrid Merge+Insertion Sort
Время сортировки (микросекунды)
Размер массива
0
5000
10000
15000
20000
25000
30000
35000
40000
2000
4000
6000
8000
10000


Время сортировки для массива типа 'Random'
Standard Merge Sort
Hybrid Merge+Insertion Sort
Время сортировки (микросекунды)
Размер массива
0
5000
10000
15000
20000
25000
2000
4000
6000
8000
10000


Время сортировки для массива типа 'Reverse Sorted'
Standard Merge Sort
Hybrid Merge+Insertion Sort
Время сортировки (микросекунды)
Размер массива
0
5000
10000
15000
20000
2000
4000
6000
8000
10000

Анализ и выводы:

1. Эффективность гибридного алгоритма:

   Малые размеры массивов - гибридный алгоритм (Merge Sort + Insertion Sort) обычно работает быстрее стандартного Merge Sort благодаря использованию более эффективного Insertion Sort для маленьких подмассивов

   Большие размеры массивов - по мере увеличения размера массива преимущество стандартного Merge Sort становится заметнее, так как накладные расходы на переключение и управление подмассивами начинают преобладать

2. Тип массивов:

   Случайные массивы - оба алгоритма демонстрируют ожидаемую производительность, но гибридный алгоритм слегка опережает стандартный для малых размеров.

   Отсортированные в обратном порядке массивы - гибридный алгоритм сохраняет преимущество для малых размеров, однако для больших размеров может уступать стандартному Merge Sort из-за накладных расходов.

   Почти отсортированные массивы - в этом случае Insertion Sort особенно эффективен для маленьких подмассивов, что делает гибридный алгоритм значительно быстрее стандартного Merge Sort для малых и средних размеров массивов.

3. Итог:

   Гибридный алгоритм Merge Sort + Insertion Sort является оптимальным выбором для массивов среднего и малого размеров благодаря сочетанию преимуществ обоих методов. Для очень больших массивов стандартный Merge Sort может быть предпочтительнее из-за своей стабильной производительности и низких накладных расходов на управление подмассивами.