Алгоритмы и структуры данных-1. SET-3. Task A2.

Илья Тямин БПИ226

1 Дисклеймер к заданиям

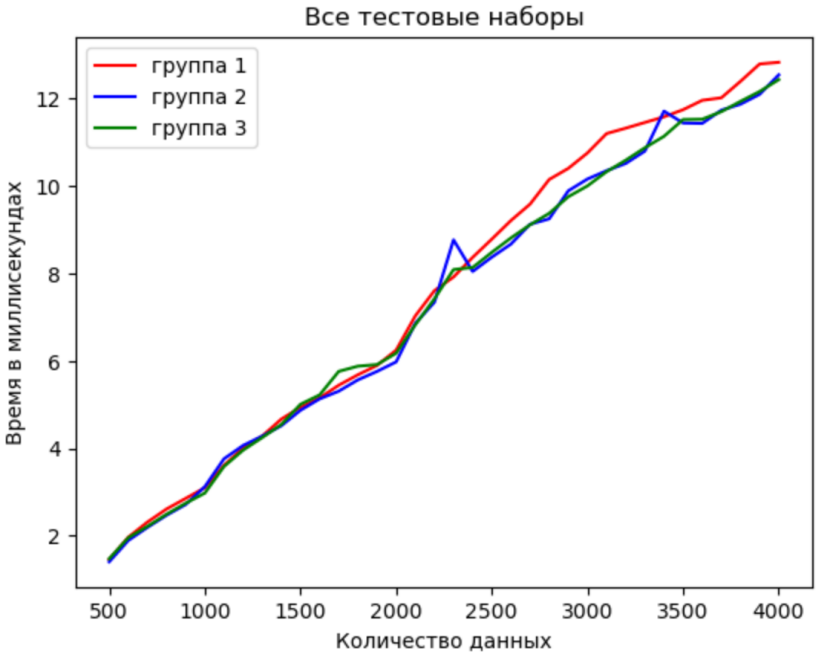
В этой задаче на проверку предоставлен архив состоящий из:

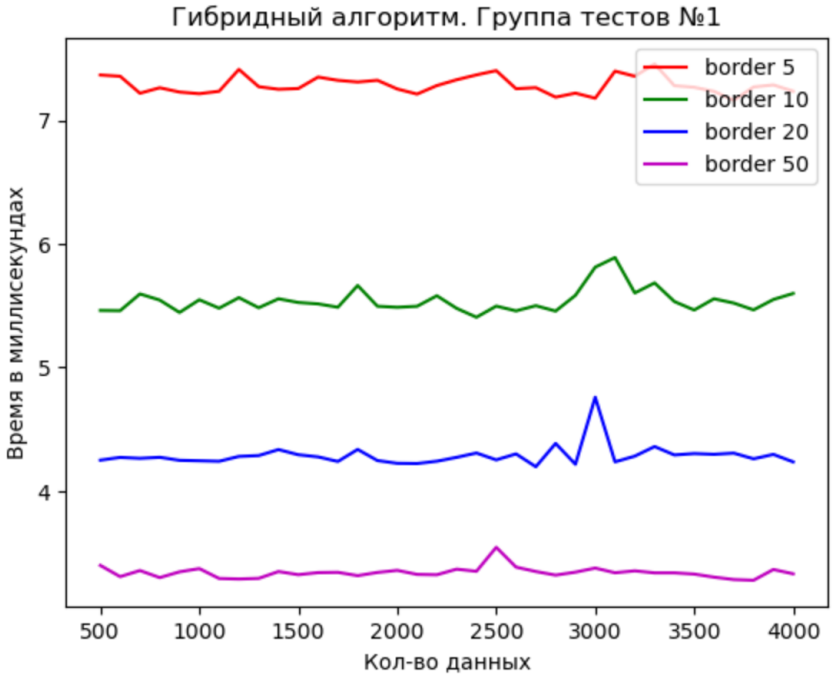
1. этого файла
2. папки cpp_generator. Рассмотрим ее поподробнее
   (a) main.cpp содержит ГЕНЕРАТОР тестовых данных (метод GenerateRandomVector). В методе main() происходит вызов данного метода и создание трех групп тестов: первая группа - 4000 случайных чисел, вторая группа - 4000 отсортированных по невозрастанию чисел, третья группа - почти отсортированные.
   (b) mergeInsertionSort.cpp содержит реализацию гибридного алгоритма Merge + Insertion Sort
   (c) mergeSort.cpp содержит реализацию нативного алгоритма Merge Sort
3. файлов data1.txt, data2.txt, data3.txt - сгенерированные наборы данных
4. файлов test_merge_1.txt (аналогично для групп 2 и 3) - результаты тестирования i-х групп, записанных в формате "кол-во данных;время". Внимание! Для большей точности результатов все тесты были совершены по 5 раз (соответственно в файле 5 раз содержатся данные для одного размера входных данных) - при обработке бралось среднее значение по одному размеру.
5. файлов test_mergeInsertion_1.txt (аналогично для групп 2 и 3) - результаты тестирования i-х групп, записанных в формате "порогПерехода;время". Аналогично тестирования проводились 5 раз.
6. файла draw_statistics.ipynb, который на основе данных, сгенерированных генератором на C++ и сохраненных в файлы, нарисовал статистику в Python. Кроме того, используя pandas были посчитаны средние значения для тестов с одинаковыми размерами входных данных (для большей достоверности значений).
7. папки graphs. Рассмотрим подробнее:
   (a) набора графиков для mergeSort - файлы вида Merge_GroupI
   (b) набора графиков для mergeInsertionSort - файлы вида MergeInsertion_GroupI_BorderI. Всего 12 графиков: для трех тестовых групп по 4 значения границы (5, 10, 20, 50).
   (c) дополнительно прикрепил сравнения всех групп merge и всех границ mergeInsertion по группам.

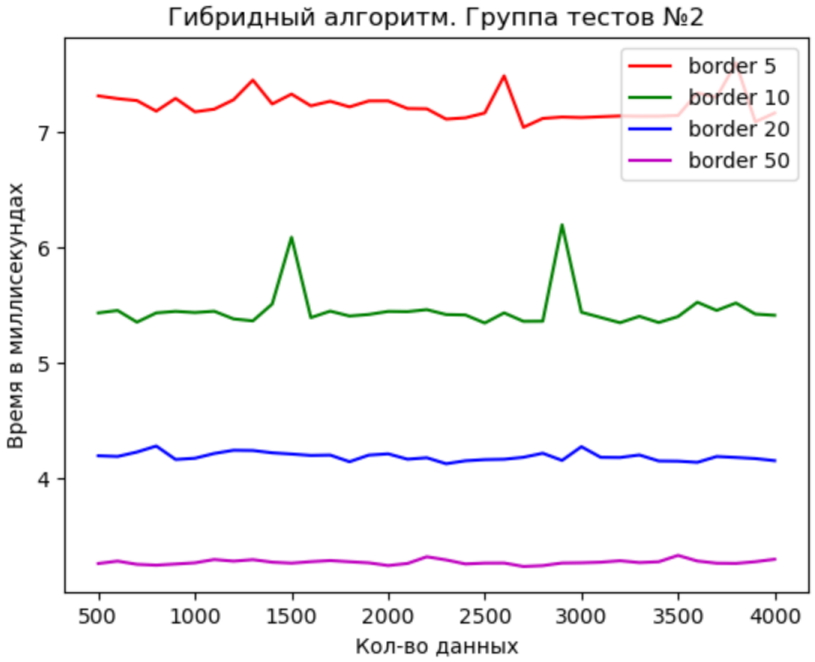
Важно! Если есть необходимость в просмотре файла с созданием статистики (ipynb), удобнее это делать, посмотрев на моем гитхабе

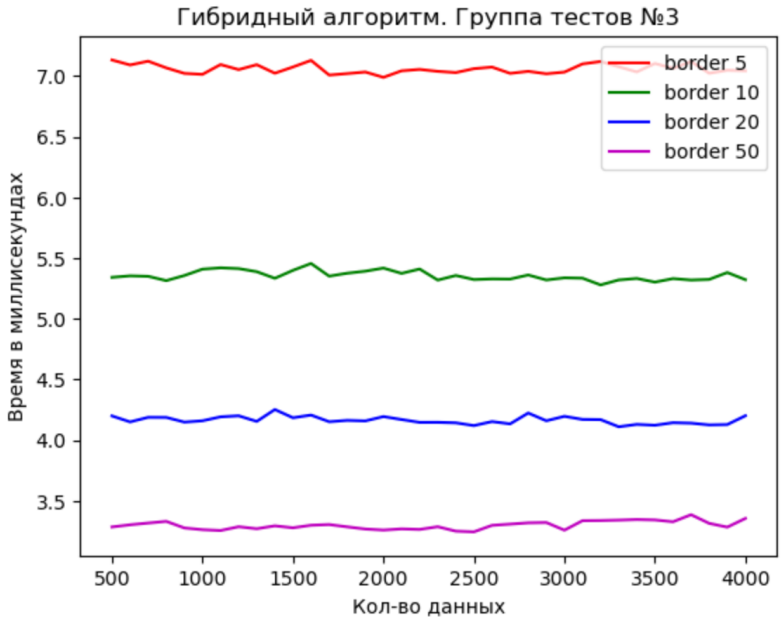
2 3. Опишите полученные вами результаты и представьте выводы о сравнении временных затрат двух рассматриваемых алгоритмов.

Посмотрим на дополнительно реализованные графики (их и основные графики можно найти в папке graphs):









Можно сделать выводы, что:

1. При любой рассмотренной границе по переходу в Insertion Sort гибридный алгоритм работает быстрее нативного Merge Sort. Это подтверждается тем, что максимальное выявленное время при эксперименте у MergeInsertion Sort - около 8 секунд (для тестовой группы №1), в то время как максимальное время нативного алгоритма на том же первом наборе данных - примерно 12 секунд. Разница есть и на других наборах данных: на 3-м наборе данных при 4000 элементов время выполнения почти 12 секунд, у гибридного алгоритма время выполнения колеблется от примерно 7 секунд при границе в 5 элементов и доходит аж до 3.5 секунд при границе в 50 элементов (разница почти в 4 раза)!

2. Нативная реализация Merge Sort показывает результаты с самыми большими выбросами при 1-м наборе данных (совершенно случайных) - здесь про выбросы имеется в виду значительное отклонение от регрессионной линии тренда. Результаты 2-й и 3-й групп данных несильно отличаются из-за небольшого объема данных.

3. Что касается гибридного алгоритма, то он показал лучшие значения времени при границе перехода в $n = 50$ элементов на всех тестах (причем улучшение во времени являются значительным: по сравнению с границей в 5 элементов прирост в 2 раза).