Отчёт по Д39 Кокорев Артём Вячеславович БПИ235

Вся нужная информация:

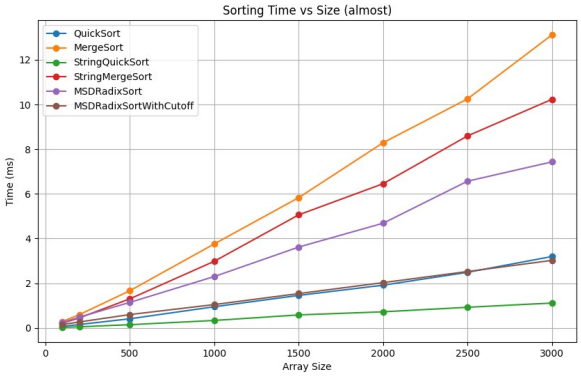
ID посылок: A1m – 321303240, A1q – 321303250, A1r – 321303261, A1rq – 321303292
Ссылка на репозиторий: https://github.com/w1sq/hse_algo/tree/main/HW9
Всё остальное в этом же файле :)

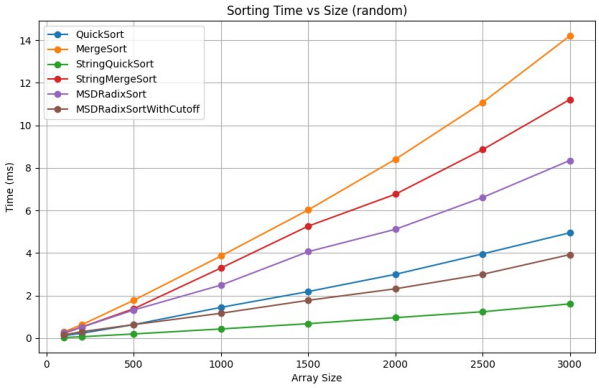
Порядок выполнения:

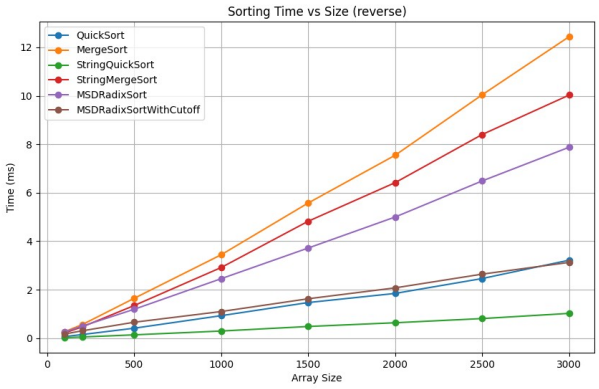
1) Написал все сортировки и заслал в контест
2) Дописал классические сортировки
3) Написал инфраструктуру на C++(файлы main.cpp StringGenerator.cpp, StringSortTester.cpp)
4) Протестировал работу бинарника из main.cpp
5) Дописал Python-скрипт для многочисленного запуска, обработки данных и визуализации
6) Написал этот отчет

Перейдём к результатам(оригиналы картинок в github):

Время:

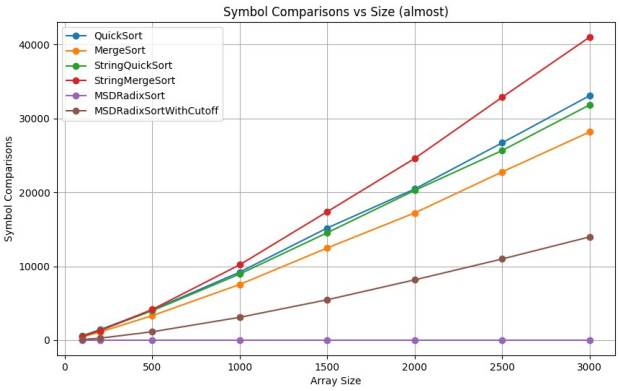


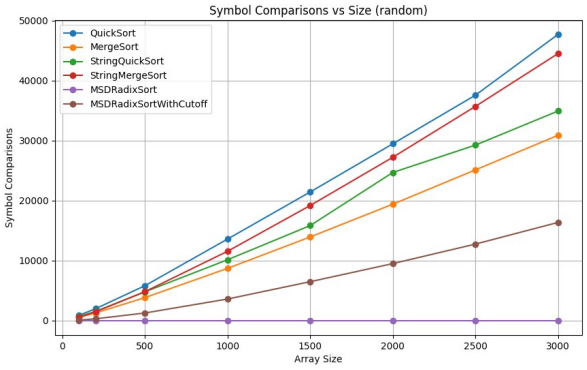


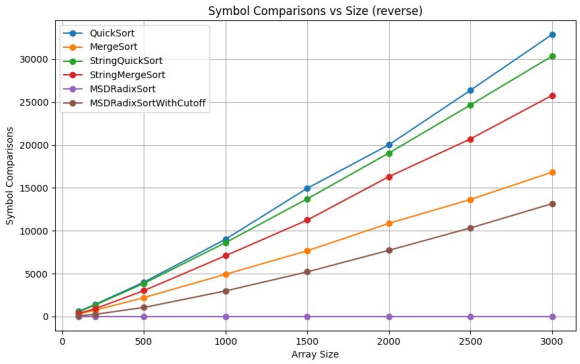


Показатели времени довольно предсказуемы. MergeSort всегда оказывается самым медленным, даже с оптимизацией. Стоит заметить, что pivot в QuickSort выбирается в середине, если бы он выбирался в конце или в начале — на reverse датасете он бы всегда проигрывал и улетал в космос и по времени и по сравнениям. Поэтому для наглядности сравнения(спасая масштаб графика) было принято решение ставить его в центр.

Сравнения:







Так же ничего необычного. RadixSort не делает никаких сравнений и пользует хэшмапу под капотом. Но по времени работает медленнее чем RadixSortWithCutoff, получается очень наглядный трейдофф скорости на память. Все остальные сортировки ведут себя ± похоже в количестве сравнений.

Рассчёт теории:

n — число строк, L — средняя длина строки, M — мощность алфавита (74)

1) String QuickSort

В среднем O(nlogn) В худшем — O(n, L)

2)String MergeSort (LCP)

Базово O(nlogn) сравнений

С учётом LCP: O(nlogn + LCP), в худшем — O(n, L)

3)MSD Radix Sort без Cutoff

Линейно по данным — O(n, L + M)

Посимвольных сравнений - 0

4)MSD Radix Sort с Cutoff

Линейный этап + для сегментов m ≤ M QuickSort за O(mlogm)

Общая оценка - O(n, L + MlogM)

Сравниваем с реальными значениями на рандомном датасете и 3000

Эмпирические данные (random, n=3000)

Алгоритм	Время, мс	Сравнений
QuickSort	4.95	47,724
MergeSort	14.20	30,900
String QuickSort	1.61	34,946
String MergeSort	11.21	44,526
MSD Radix Sort	8.35	0
MSD Radix SortWithCutoff	3.93	16,383

1) String QuickSort — время чуть выше nlogn из-за LCP-вставок

2)String MergeSort (LCP) — порядок приблизительно nlogn, но свою роль играют LCP и двойное копирование

3)MSD Radix Sort без Cutoff — 0 сравнений и линия по времени, всё чётко

4)MSD Radix Sort с Cutoff — совсем немного сравнений из-за ограничения алфавитом, проигрывает quicksort по скорости, видимо из-за инициализации внутренних структур данных, а объем 3000 маловат