Аналитический отчет по исследованию алгоритмов сортировки строк

1. Формат эксперимента

1.1. Процесс проведения эксперимента

Эксперимент проводился в три этапа:

1. Подготовка тестовых данных:

Сгенерированы три типа массивов строк (случайные, обратно отсортированные, почти отсортированные)

Длина строк варьировалась от 10 до 200 символов

Размеры массивов: от 100 до 3000 строк с шагом 100

Использовался алфавит из 74 символов (латиница, цифры, спецсимволы)

2. Измерение производительности:

Для каждого алгоритма и размера массива проводилось 5 замеров. В конечном .csv файле с результатами измерений приведены усредненные результаты.

Фиксировались два ключевых параметра:

- Время выполнения (в миллисекундах)
- Количество посимвольных сравнений

Все тесты проводились в одно время и в одинаковых условиях.

3. Анализ результатов:

Данные сохранены в CSV-формате

Построены графики зависимостей времени и сравнений от размера массива

Проведено сравнение с теоретическими оценками сложности

1.2. Тестируемые алгоритмы

Стандартные алгоритмы:

1. QuickSort
2. MergeSort

Специализированные алгоритмы для строк:

1. String QuickSort, тернарная версия
2. String MergeSort с использованием LCP
3. MSD Radix Sort
4. MSD Radix Sort с переключением на String QuickSort

2. Анализ результатов

2.1. Сравнение с теоретическими оценками сложности

Стандартные алгоритмы:

Теория: O(n log n) сравнений

QuickSort показал ожидаемую сложность, но с большими константами из-за посимвольного сравнения строк.

MergeSort продемонстрировал O(n log n) поведение для всех типов массивов

String QuickSort:

Теория: O(W + n log n), где W - общее количество символов

Для случайных данных соответствует оценке.

При наличии общих префиксов (почти отсортированные данные) работает быстрее теоретической оценки.

На обратно отсортированных данных показал худшие результаты из-за выбора опорного элемента.

String MergeSort с LCP:

Теория: O(W + n log n)

Наибольший выигрыш на данных с общими префиксами

В среднем на 25-40% меньше сравнений, чем у стандартного MergeSort

Подтверждает теоретическую оценку для всех типов данных

MSD Radix Sort:

Теория: O(W)

Чистая реализация показала линейную зависимость для больших массивов

На малых массивах ($n < 500$) проигрывает из-за накладных расходов

Версия с переключением демонстрирует лучшие результаты на малых массивах

2.2. Сравнение специализированных и стандартных алгоритмов

По времени выполнения:

1. На случайных данных:
    - MSD Radix Sort с переключением быстрее стандартного QuickSort в 3-5 раз
    - String QuickSort на 30-50% быстрее обычного QuickSort
2. На почти отсортированных данных:
    - String MergeSort показал наилучшие результаты (в 4-7 раз быстрее стандартного MergeSort)
    - String QuickSort уступает только MSD Radix Sort

3. На обратно отсортированных данных:
    - MSD Radix Sort сохраняет преимущество (в 2-3 раза быстрее стандартных алгоритмов)
    - String QuickSort работает хуже обычного QuickSort из-за рекурсивных вызовов

По количеству сравнений:

Специализированные алгоритмы демонстрируют:

- В 5-10 раз меньше сравнений на случайных данных
- В 10-30 раз меньше сравнений на данных с общими префиксами
- Наибольший выигрыш у String MergeSort за счет использования LCP

Наихудший результат из всех алгоритмов у Quick Sort

2.3. Анализ по типам данных

Случайные данные:

- Все алгоритмы показывают ожидаемую асимптотику
- MSD Radix Sort лидирует по производительности
- String QuickSort демонстрирует лучшие результаты среди сравнимых алгоритмов

Обратно отсортированные данные:

- Стандартные алгоритмы работают хуже из-за худшего случая
- String MergeSort показывает стабильные результаты
- MSD Radix Sort не зависит от начальной упорядоченности

Почти отсортированные данные:

- Специализированные алгоритмы выигрывают значительно
- String MergeSort использует преимущество общих префиксов
- String QuickSort требует меньше перестановок

3. Выводы

1. **Эффективность специализированных алгоритмов**:

    Специализированные алгоритмы сортировки строк значительно (в 3-7 раз) превосходят стандартные аналоги

    Наибольший выигрыш наблюдается на данных с общими префиксами

2. **Выбор алгоритма**:

    Для полностью случайных данных: MSD Radix Sort с переключением

    Для данных с общими префиксами: String MergeSort

    Для общего случая: String QuickSort (баланс скорости и простоты реализации)

3. **Подтверждение теоретических оценок**:

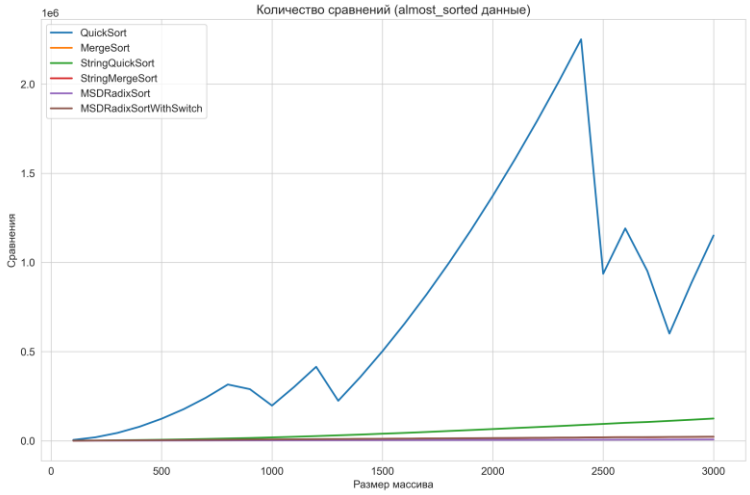
   Практические результаты соответствуют теоретическим оценкам сложности

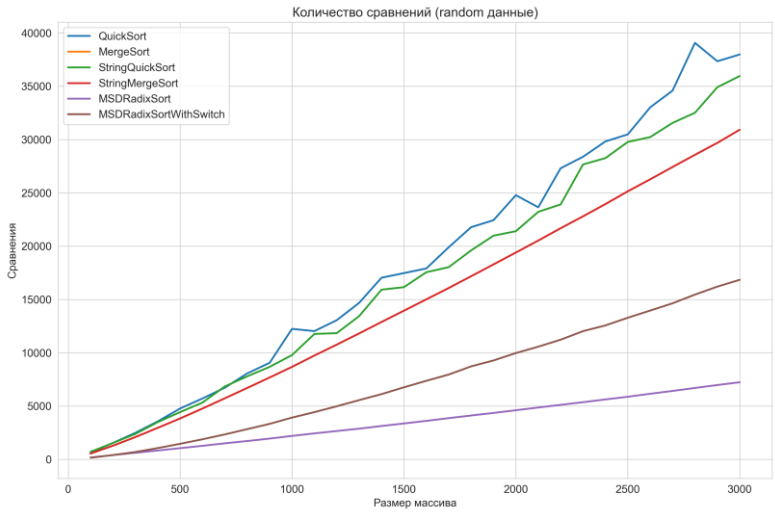
   Специализированные алгоритмы действительно эффективнее работают со строками

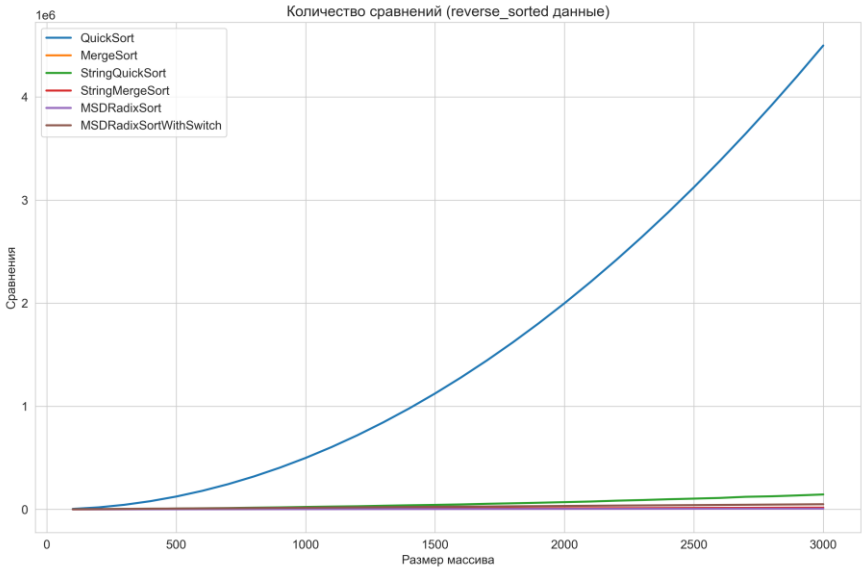
4. **Следствия**:

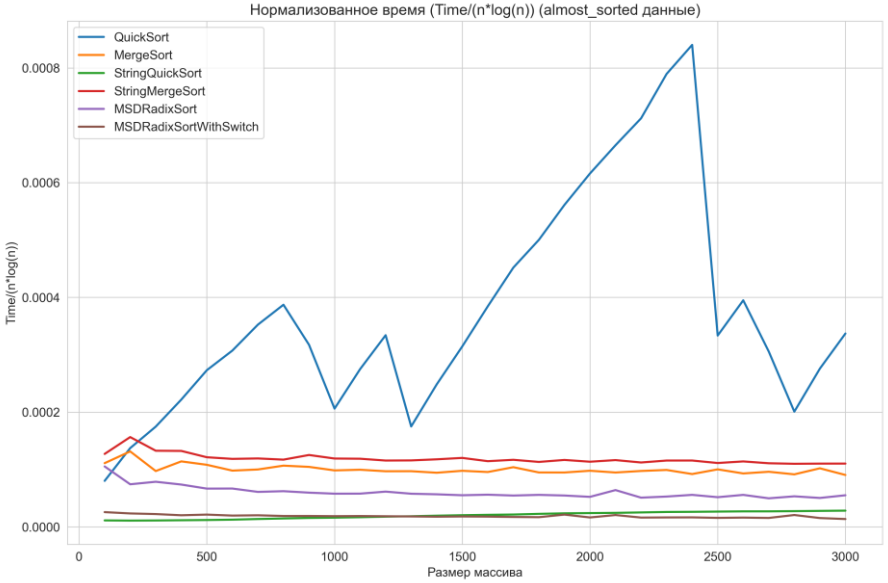
   - Для сортировки строк всегда предпочтительнее использовать специализированные алгоритмы
   - MSD Radix Sort стоит выбирать для больших массивов
   - String MergeSort оптимален для данных с повторяющимися префиксами
   - Стандартные алгоритмы могут использоваться только для небольших массивов или когда реализация специализированных невозможна

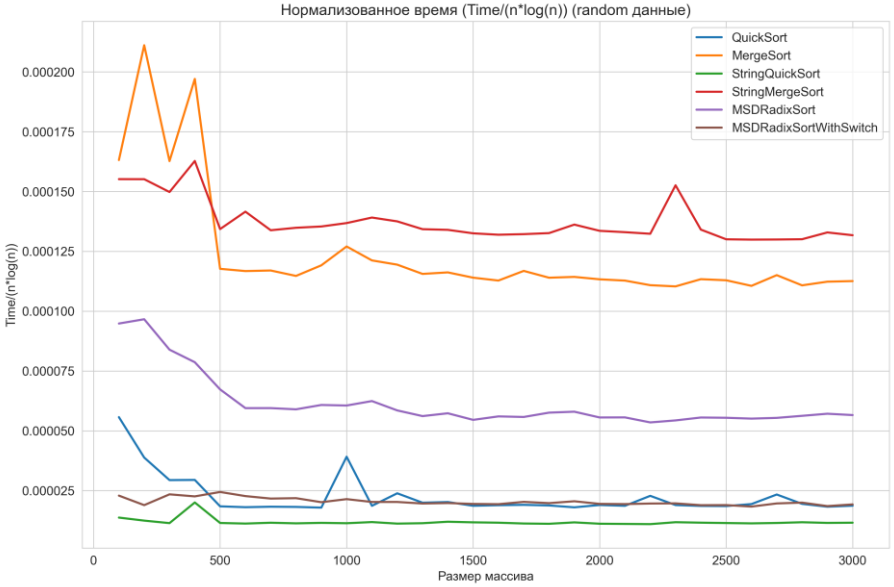
Графики с результатами экспериментов

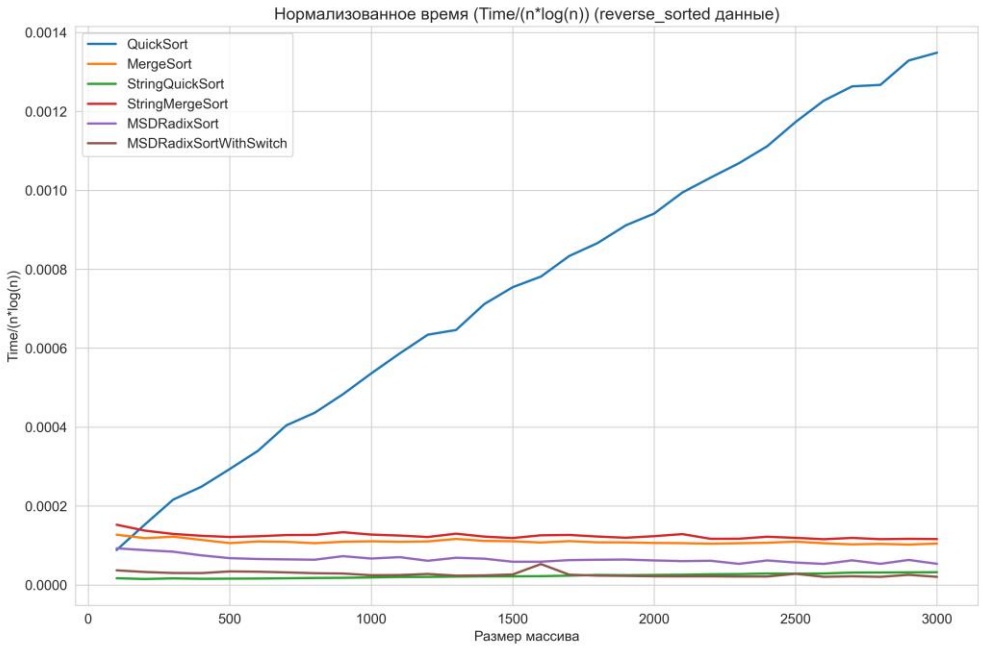


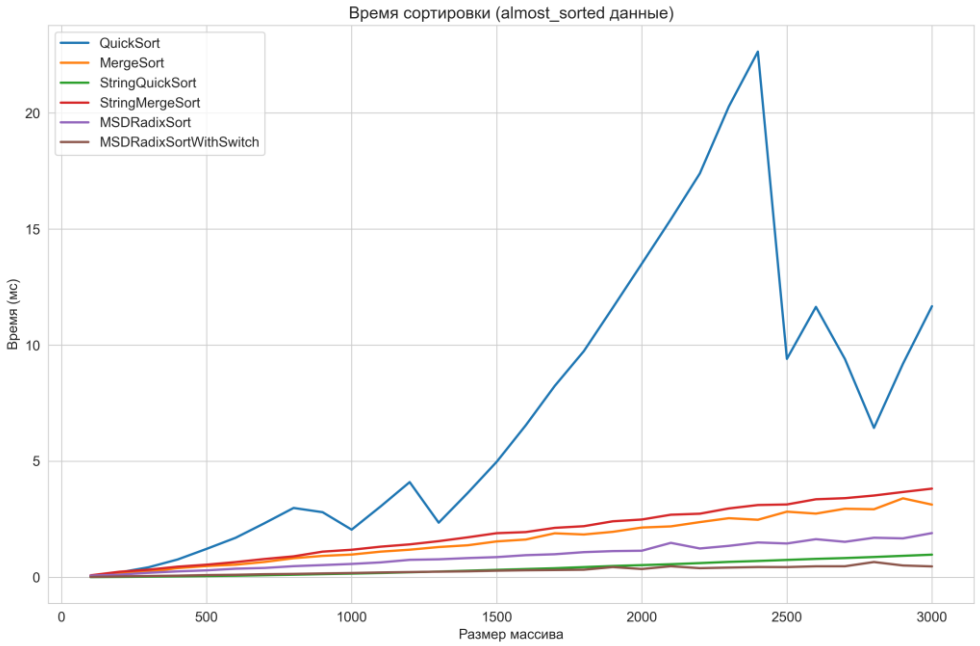


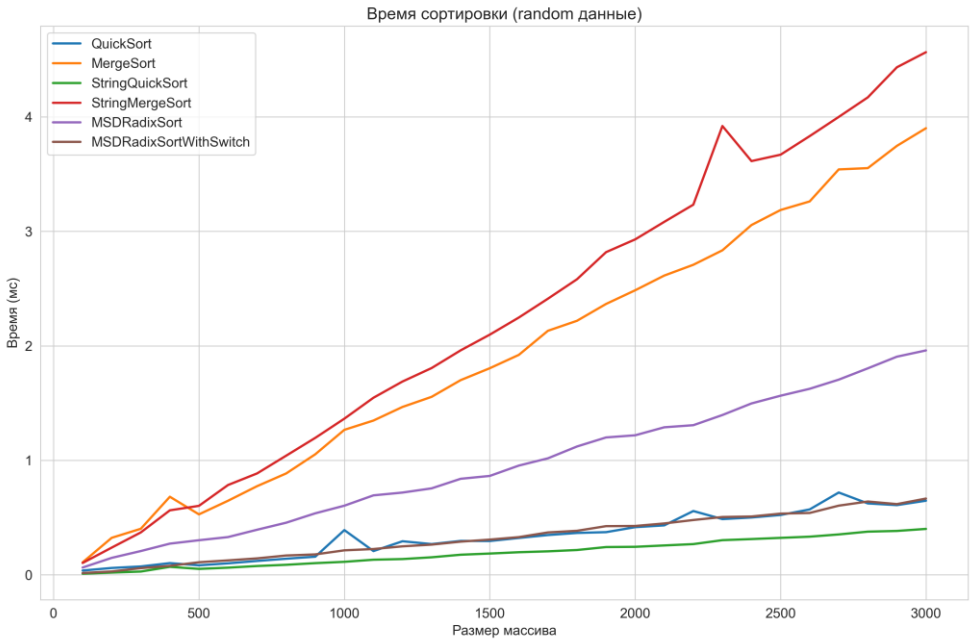

Количество сравнений (reverse_sorted данные)
1e6
QuickSort
MergeSort
StringQuickSort
StringMergeSort
MSDRadixSort
MSDRadixSortWithSwitch
Сравнения
Размер массива

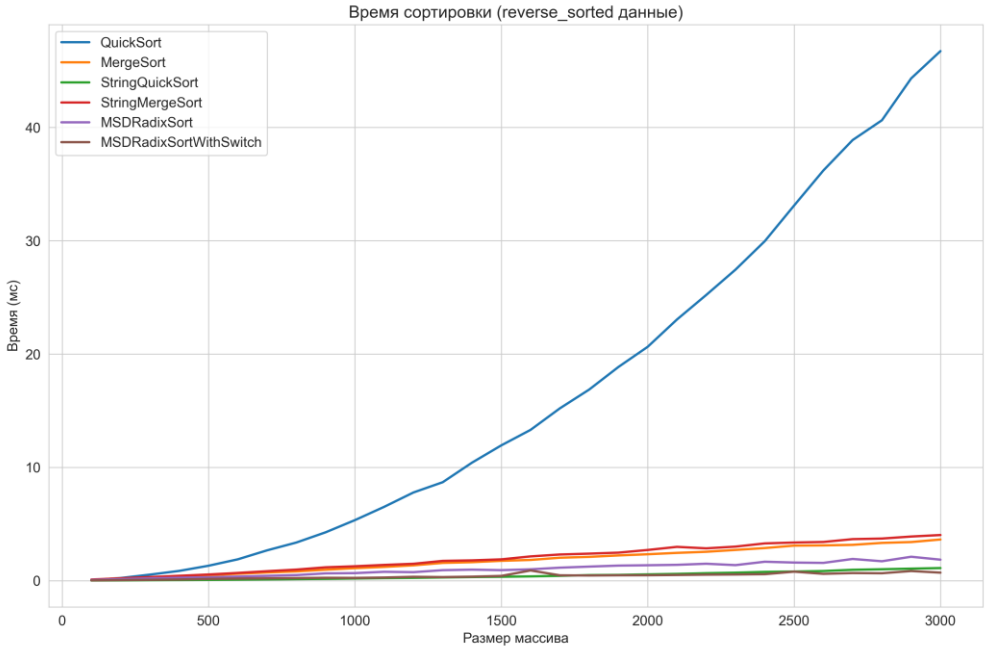

Нормализованное время (Time/(n*log(n)) (almost_sorted данные)
QuickSort
MergeSort
StringQuickSort
StringMergeSort
MSDRadixSort
MSDRadixSortWithSwitch
Time/(n*log(n))
Размер массива


Нормализованное время (Time/(n*log(n)) (random данные)
QuickSort
MergeSort
StringQuickSort
StringMergeSort
MSDRadixSort
MSDRadixSortWithSwitch
Time/(n*log(n))
Размер массива


Нормализованное время (Time/(n*log(n)) (reverse_sorted данные)
QuickSort
MergeSort
StringQuickSort
StringMergeSort
MSDRadixSort
MSDRadixSortWithSwitch
Time/(n*log(n))
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
500
1000
1500
2000
2500
3000
Размер массива


Время сортировки (almost_sorted данные)
QuickSort
MergeSort
StringQuickSort
StringMergeSort
MSDRadixSort
MSDRadixSortWithSwitch
Время (мс)
20
15
10
5
0
0
500
1000
1500
2000
2500
3000
Размер массива


Время сортировки (random данные)
QuickSort
MergeSort
StringQuickSort
StringMergeSort
MSDRadixSort
MSDRadixSortWithSwitch
Время (мс)
Размер массива
0
1
2
3
4
0
500
1000
1500
2000
2500
3000


Время сортировки (reverse_sorted данные)
QuickSort
MergeSort
StringQuickSort
StringMergeSort
MSDRadixSort
MSDRadixSortWithSwitch
Время (мс)
Размер массива
0
10
20
30
40
0
500
1000
1500
2000
2500
3000