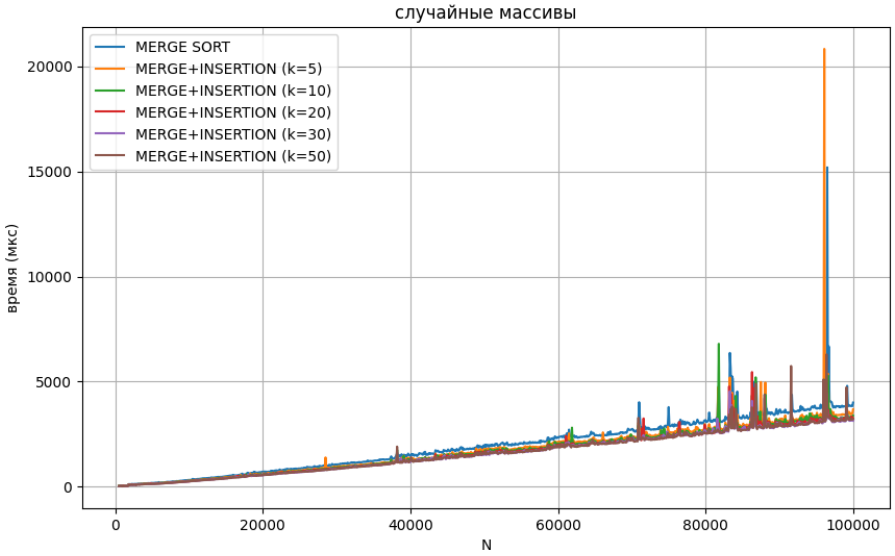
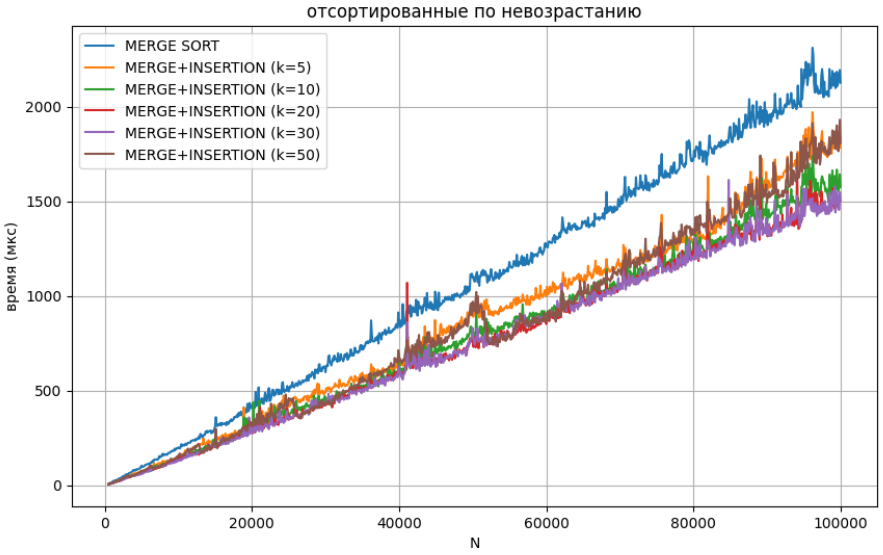
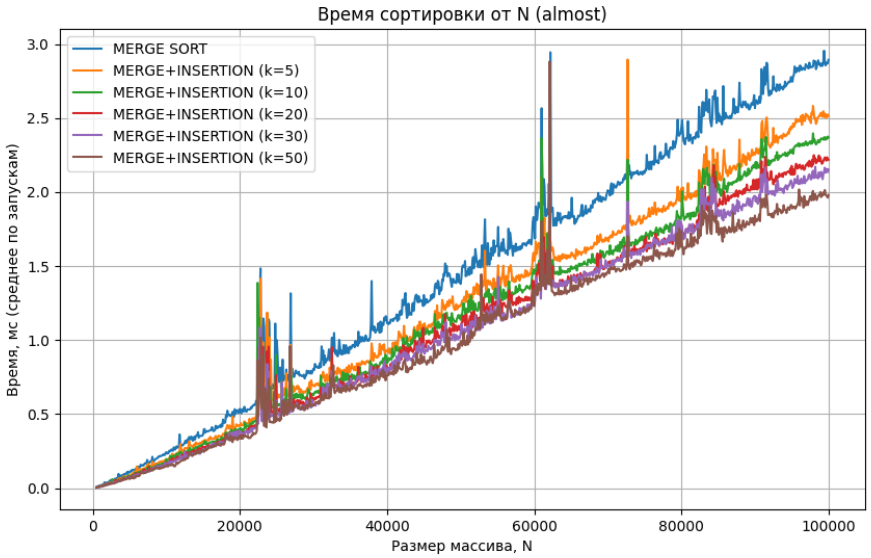

случайные массивы
MERGE SORT
MERGE+INSERTION (k=5)
MERGE+INSERTION (k=10)
MERGE+INSERTION (k=20)
MERGE+INSERTION (k=30)
MERGE+INSERTION (k=50)
время (мкс)
N


отсортированные по невозрастанию
MERGE SORT
MERGE+INSERTION (k=5)
MERGE+INSERTION (k=10)
MERGE+INSERTION (k=20)
MERGE+INSERTION (k=30)
MERGE+INSERTION (k=50)
время (мкс)
N


Время сортировки от N (almost)
MERGE SORT
MERGE+INSERTION (k=5)
MERGE+INSERTION (k=10)
MERGE+INSERTION (k=20)
MERGE+INSERTION (k=30)
MERGE+INSERTION (k=50)
Время, мс (среднее по запускам)
Размер массива, N

## Вывод:

1. **Характер роста времени.**
   Для всех рассмотренных алгоритмов (чистый MERGE SORT и гибрид MERGE+INSERTION) время работы растёт примерно как O(n log(n): на графиках при увеличении размера массива N кривые растут быстрее, чем линейно, но медленнее, чем квадратично.
2. **Сравнение MERGE и MERGE+INSERTION.**
   На всех трёх типах данных гибридный алгоритм в среднем работает быстрее классического MERGE SORT.
3. **Влияние порога переключения.**
   Порог, при котором рекурсивный MERGE SORT заменяется на INSERTION SORT, сильно влияет на скорость:
   - слишком маленькие пороги (например, k=5) дают небольшой выигрыш, потому что почти всё время продолжает тратиться на рекурсивные вызовы и слияния;
   - по полученным графикам для случайных и обратных массивов, видно, что наиболее выгодными оказались пороги порядка k≈20 - 30: для них время работы минимально на большинстве размеров N.
4. **Зависимость от структуры массива.**
   - Для случайных массивов и массивов, отсортированных в обратном порядке, выигрыш гибрида над чистым MERGE SORT есть, но не слишком большой: кривые близко друг к другу, особенно при малых N.
   - Для почти отсортированных массивов разница более заметна: алгоритм MERGE+INSERTION лучше использует локальную упорядоченность, и его кривые лежат ниже, чем у обычного MERGE SORT, на всём диапазоне N.
5. **Практический вывод.**
   Гибрид MERGE+INSERTION с порогом переключения порядка нескольких десятков элементов (20–30) является в среднем более эффективным, чем классический MERGE SORT, для всех трёх типов входных данных. Такой выбор позволяет сохранить асимптотику O(n log(n)), но уменьшить константы и улучшить реальное время работы.

ID: 349328300
github: https://github.com/Adaev1/Algorithms-HW3