Анализ работы сортировок в C++

Автор: Белкин А.А.

Группа: РЛ6-21

Основной цикл (for):

```
for (int size = 1000; size < MAX_ARRAY_SIZE; size+=1000)
{
    my_array = new int[size];

    for (int i = 0; i <= 6; i++)
    {
        if (sortCompleted[i])
        {
            continue;
        }

        memcpy(my_array, randArray, size * sizeof(int));

        files[i] << size << "\t";
        unsigned long long time = 0;
        for (int j = 0; j < ITERATIONS_COUNT; ++j)
        {
            auto begin = chrono::high_resolution_clock::now();
            pSort[i](my_array, size);
            auto end = chrono::high_resolution_clock::now();
            time += chrono::duration_cast<std::chrono::nanoseconds>(end - begin).count();
        }
        time /= ITERATIONS_COUNT;
        if (time >= MAX_SORT_TIME)
        {
            sortCompleted[i] = true;
        }
        files[i] << time << "\n";
    }
    delete[] my_array;
}
```

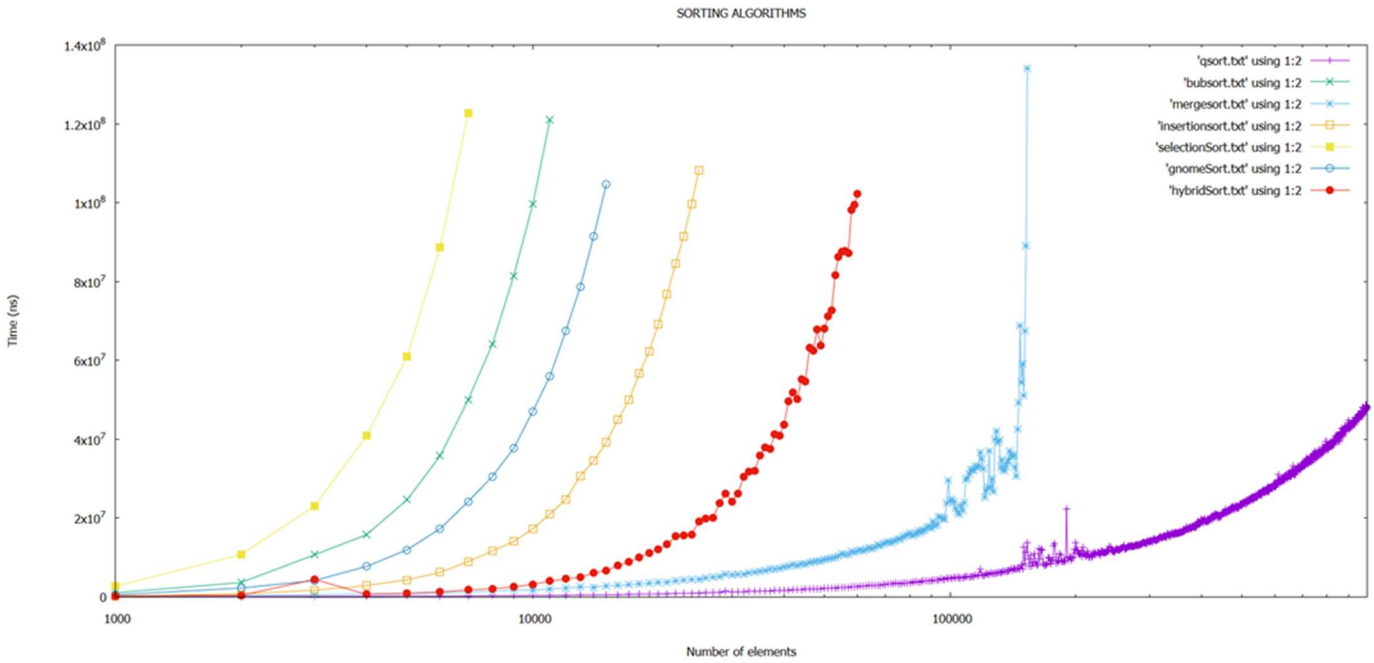
В основной части программы используется цикл (for) для постепенного увеличения размера сортируемого массива. Ограничения по размерам массива: минимальный размер – 500 элементов, максимальный размер – 1000000 элементов, заключённый в MAX_ARRAY_SIZE. Также используется ограничение по времени выполнения одного цикла сортировки, которое должно не превышать 100000000 наносекунд (0.1 секунда). Данное ограничения заключено в MAX_SORT_TIME. Для более удобного отбражения результатов работы программы основный график работы всех алгоритмов сортировки представлен с использование логарифмической шкалы.

Алгоритм работы цикла (for):

1. Проверка стадии работы алгоритма сортировки (окончено выполнение или нет)
2. Если сортировка окончила своё выполнение, то переход к следующему алгоритму сортировки
3. Заполнение массива my_array случайными значениями посредством копирования в него содержимиго массива randArray, заранее заполненного случайными числами
4. Передача в текстовый файл размера текущего массива
5. Создание и инициализация переменной time, используемой для измерения времени работы алгоритма сортировки
6. Выполнение сортировки и измерение времени работы. Для большей точности данная операция выполнятеся 5 раз (заключено в ITERATIONS_COUNT)
7. Проверка времени работы алгоритма сортировки. Если время работы превышает 0.1 секунды, то данный алгоритм сортировки считается окончившим работу и больше не используется

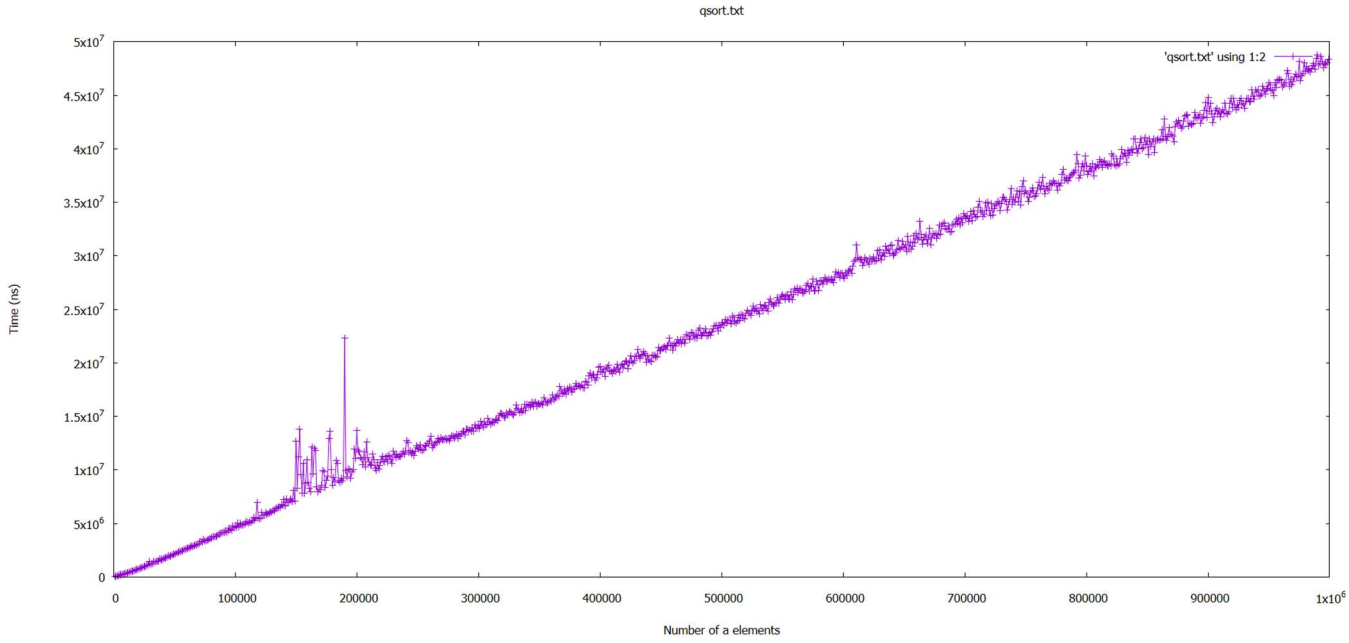
Результаты работы программы:

Общий график для всех сортировок:

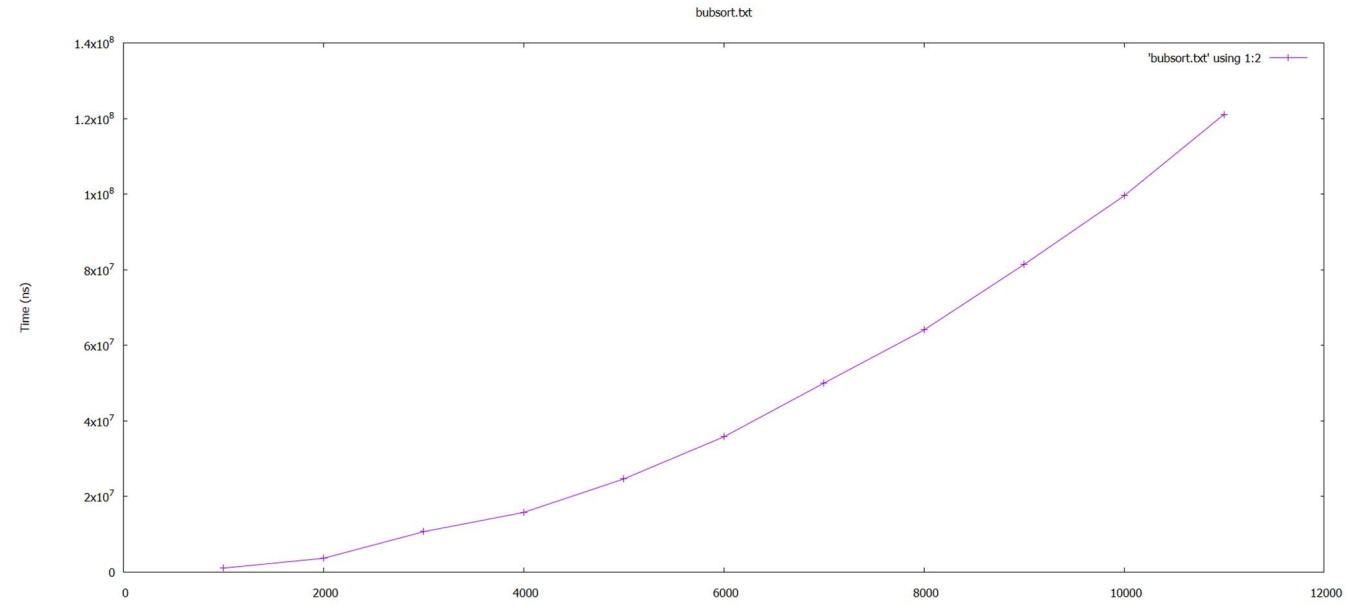


Отдельные графики сортировок:

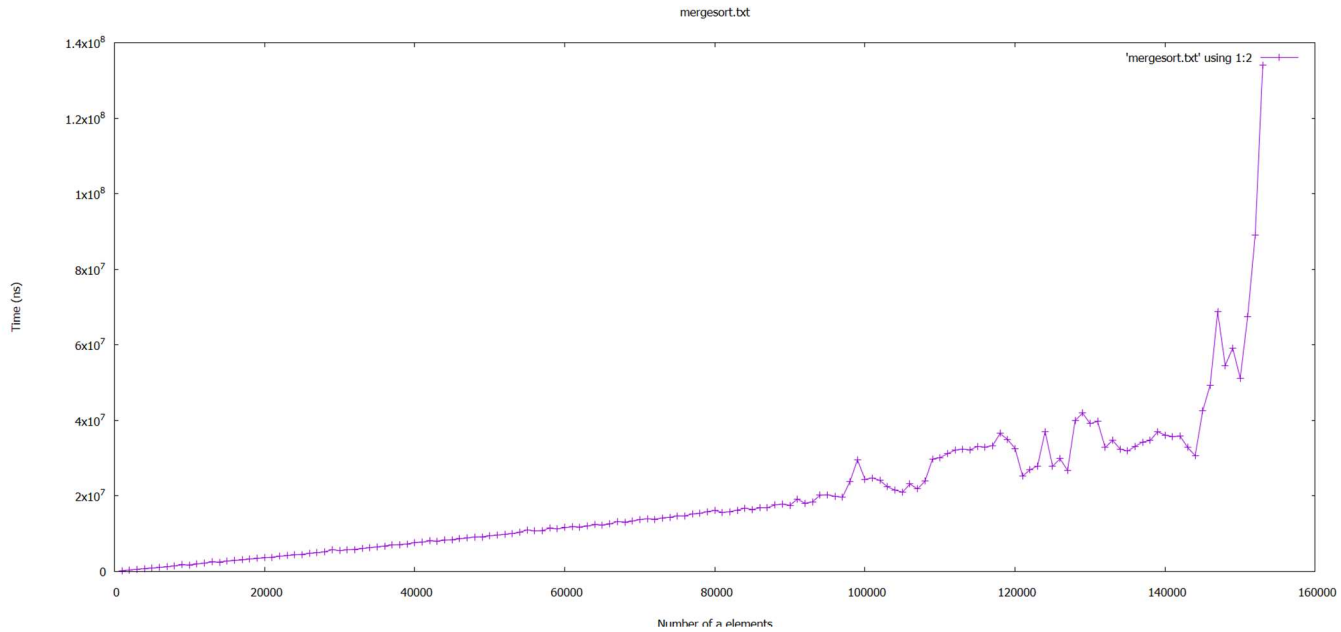
Быстрая сортировка:



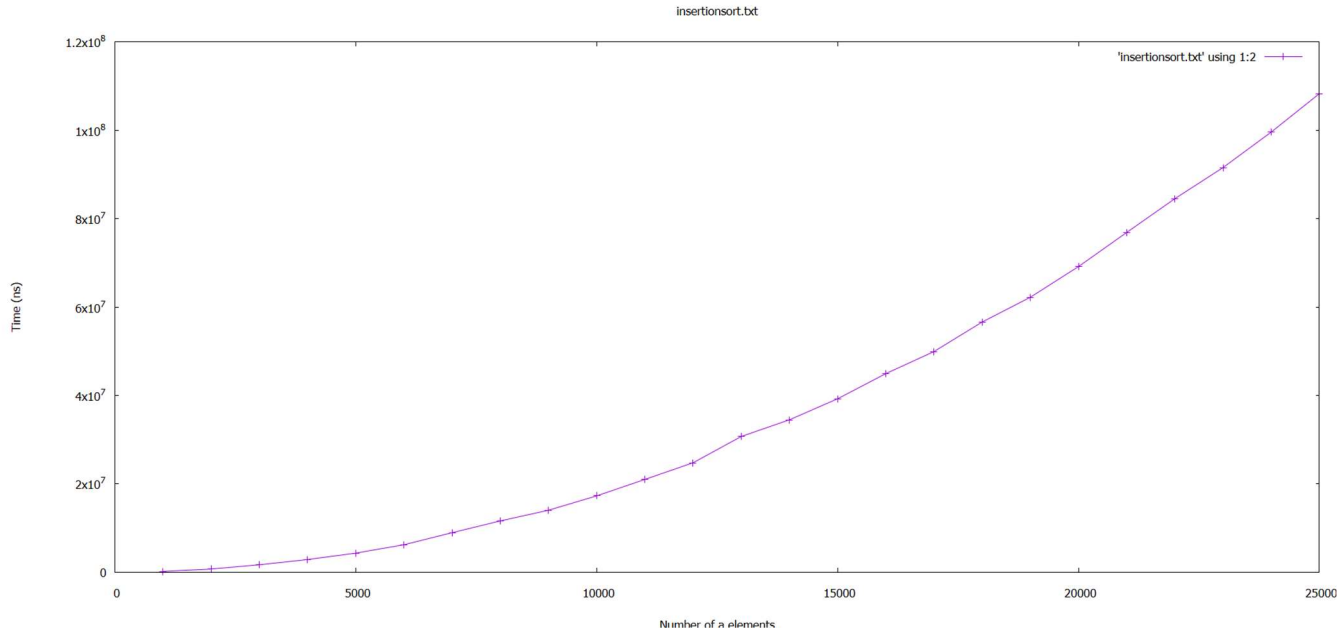
Пузырьковая сортировка:



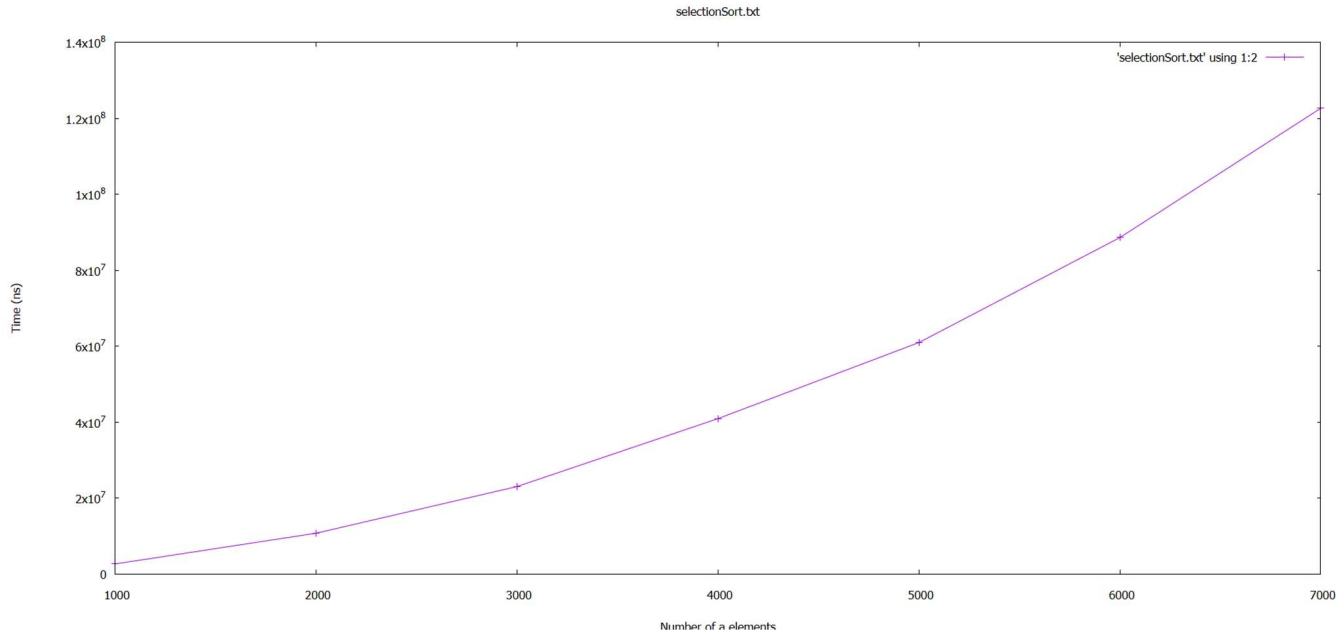
Сортировка слиянием:



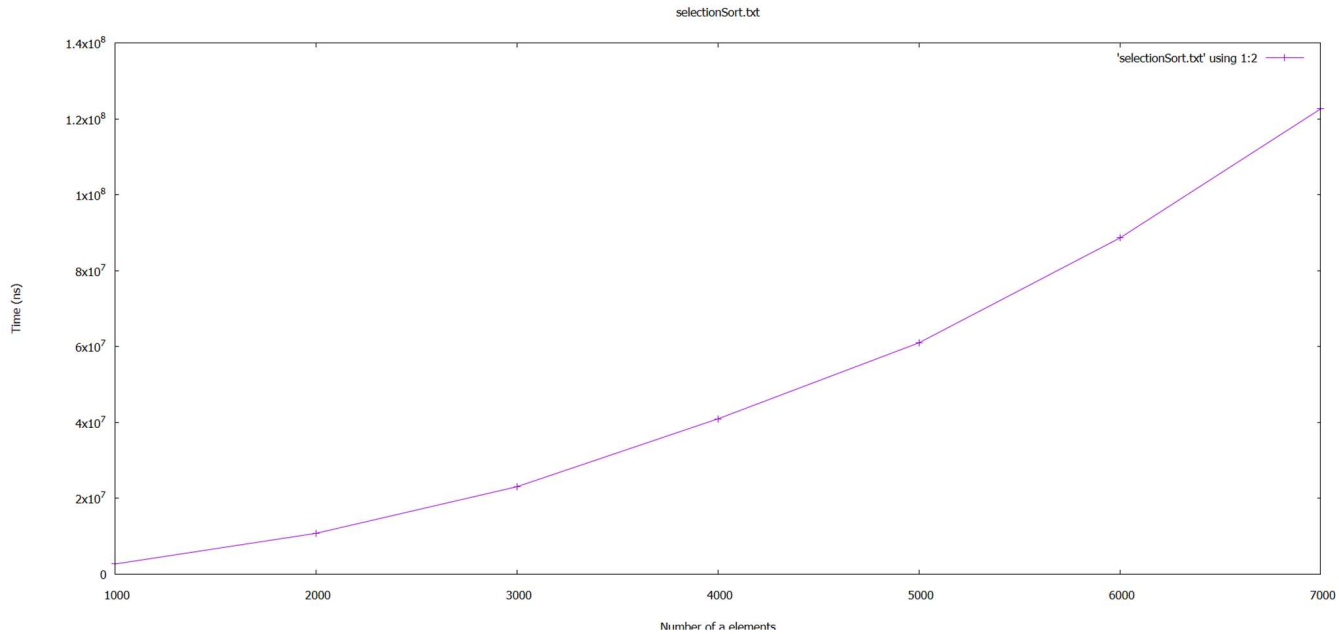
Сортировка вставками:



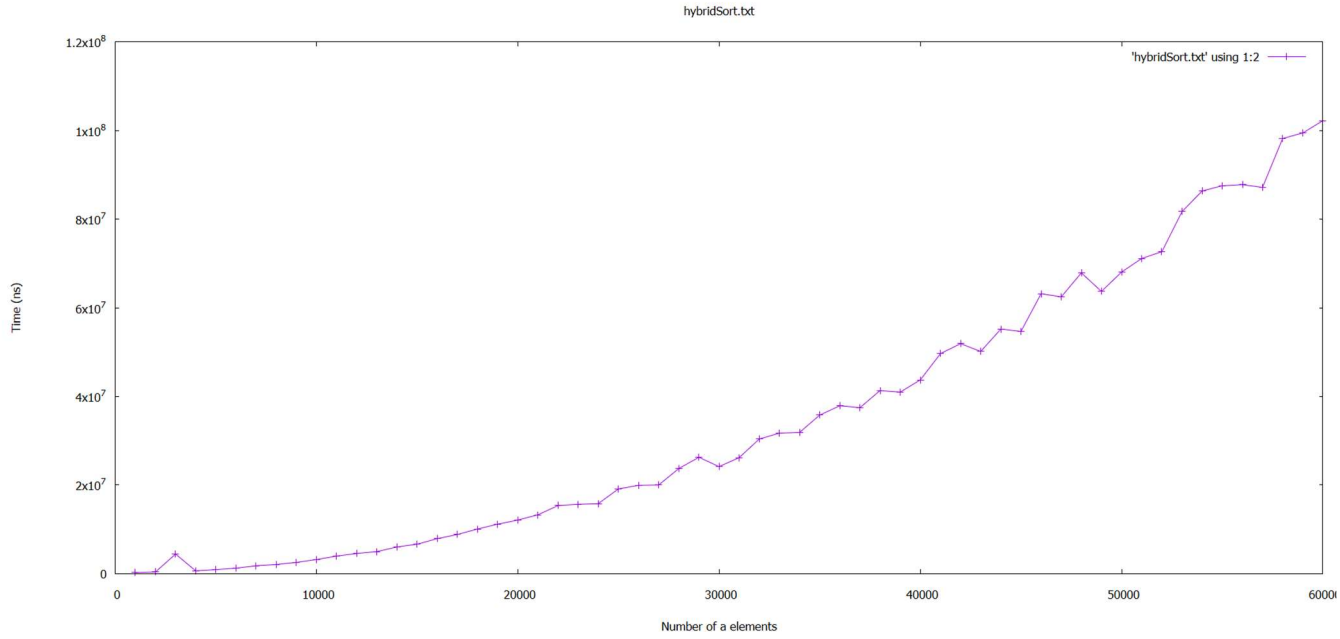
Сортировка выбором:



Гномья сортировка:



J-сортировка:



Анализ результатов:

Проводить анализ, основываясь на данных, полученных при использовании малых размеров сортируемого массива невозможно из-за погрешностей измерения времени работы, поэтому при анализе результатов работы программы я буду отталкиваться от результатов, полученных при использовании больших обьёмов данных.

Список скоростей работы сортировок (от самой медленной до самой быстрой):

1. Сортировка выбором
2. Пузырьковая сортировка
3. Гномья сортировка
4. Сортровка вставками
5. J-сортировка (кучей + вставками)
6. Сортировка слиянием
7. Быстрая сортировка

Примечание:

При работе в режиме экономии заряда: Сортировка слиянием не справляется с массивами больших размеров, особенно при увеличенном числе повторений, что иногда приводит к сильным сбоям в работе программы и прекращению её выполнения.