Анализ работы сортировок в C++

Автор: Белорусов М.Д. РЛ6-21 МГТУ им. Н.Э.Баумана

Основной цикл for

```
int size = 1000;

    for (int i = 0 ; i < 56 ; i++) {

        cout << "Sorting (" << i + 1 << " out of 56)" << endl;

        arr = new int [size];



        for  (int sn=0 ; sn<8 ; sn++){
            if (stopsort[sn] == 0){
                new_arr(arr, size);

                filesort[sn] << size << "\t";

                auto start = chrono::high_resolution_clock::now();
                sortind[sn](arr, 0, size - 1);
                auto end = chrono::high_resolution_clock::now();

                unsigned int time =
chrono::duration_cast<std::chrono::nanoseconds>(end - start).count();

                filesort[sn] << time << "\n";

                if (time>1014041300){
                    stopsort[sn]++;
                }

                TestArray(arr, size, Nameofsort[sn]);

                cout << Nameofsort[sn] << "-OK" << endl;
            }
        }
        if (size < 50000)
            size += 1000;
        else
            size *=2;


        delete[] arr;
    }
```

В основной части программы используется цикл (for) для постепенного увеличения размера сортируемого массива . Производится 56 измерений, минимальный сортируемый массив имеет размер 1000 , а максимальный 3 200 000. Это сделано для более удобного отображения на логарифмической шкале.

Далее создаётся массив необходимого размера и удаляется он только в конце цикла.

Новый цикл необходим для перебора всех нужный сортировок. Все названий, файлы и указатели на функции сформированы в массив выше.

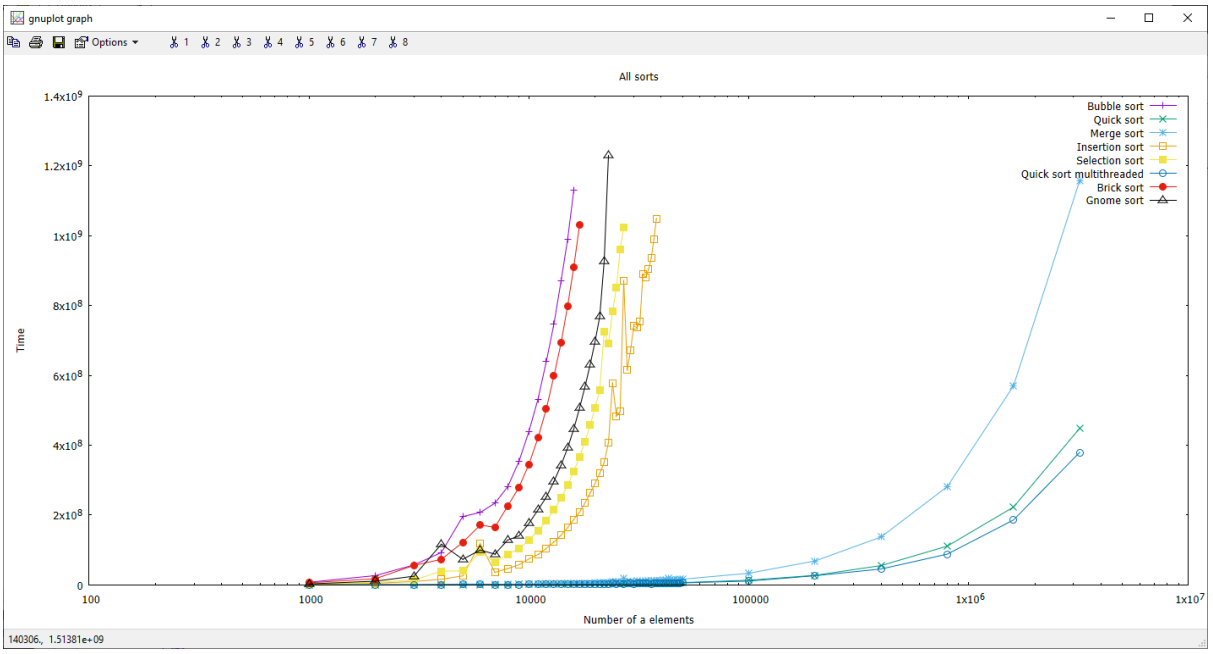
Пояснения к действиям в цикле:

1. Массив заполняется случайными значениями(В следующем цикле, отсортированный массив снова затрётся случайными значениями)
2. В текстовый файл добавляется координата X
3. Замеряется время выполнения программы (Используется библиотека chrono для более точный замеров)
4. Вызывается ссылка на необходимую функцию
5. Время выполнения программы записывается в переменную int (Особенности библиотеки chrono и типа auto)
6. В текстовый файл добавляется координата Y
7. Путём самописной функции определяется отсортированность массива.

Результаты работы программы

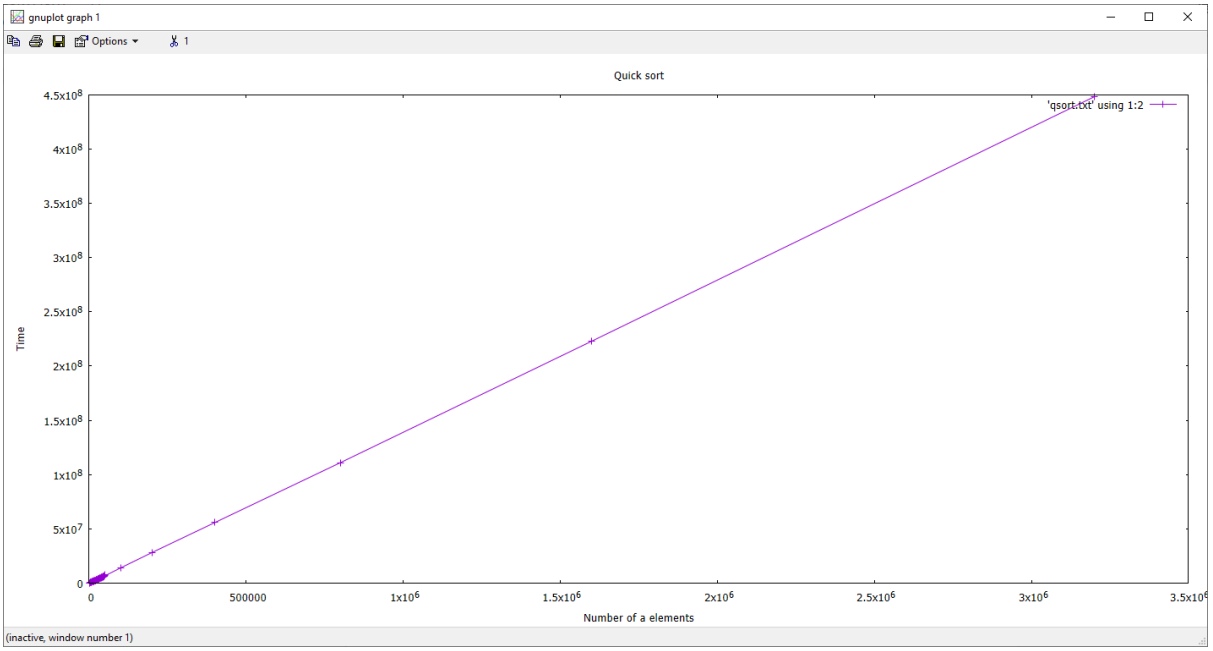
Для приведённый графиком бралось среднее значение из 3 замеров для более точного отображения эффективности программы.

График всех сортировок

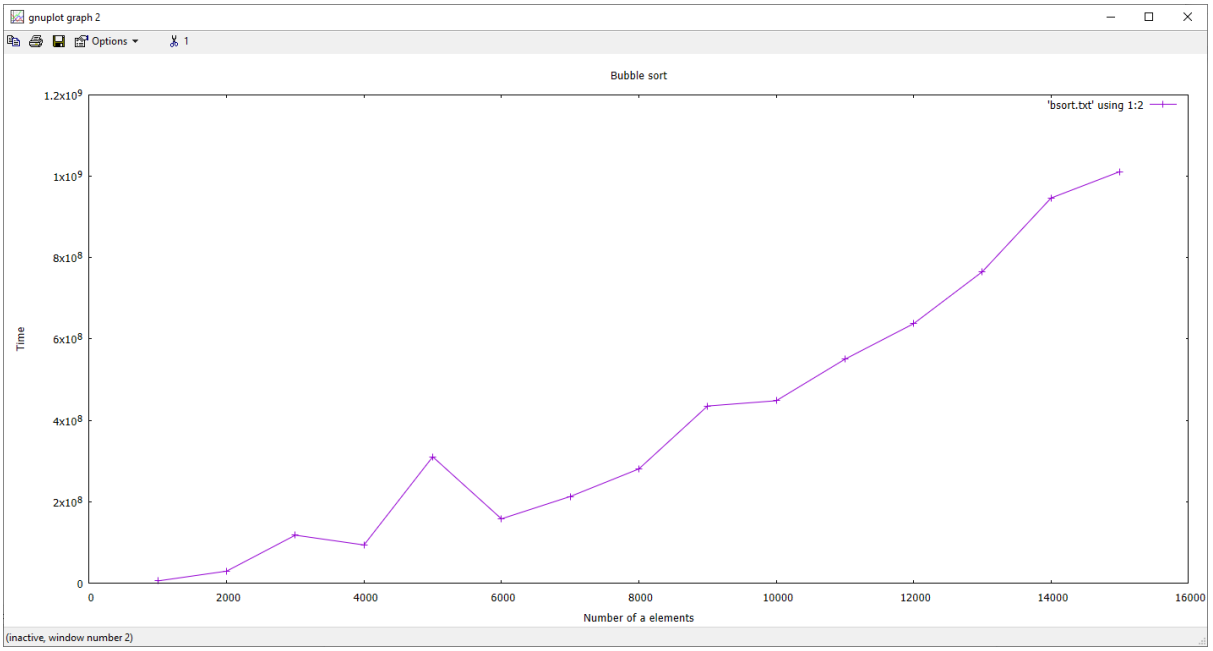


Отдельные графики

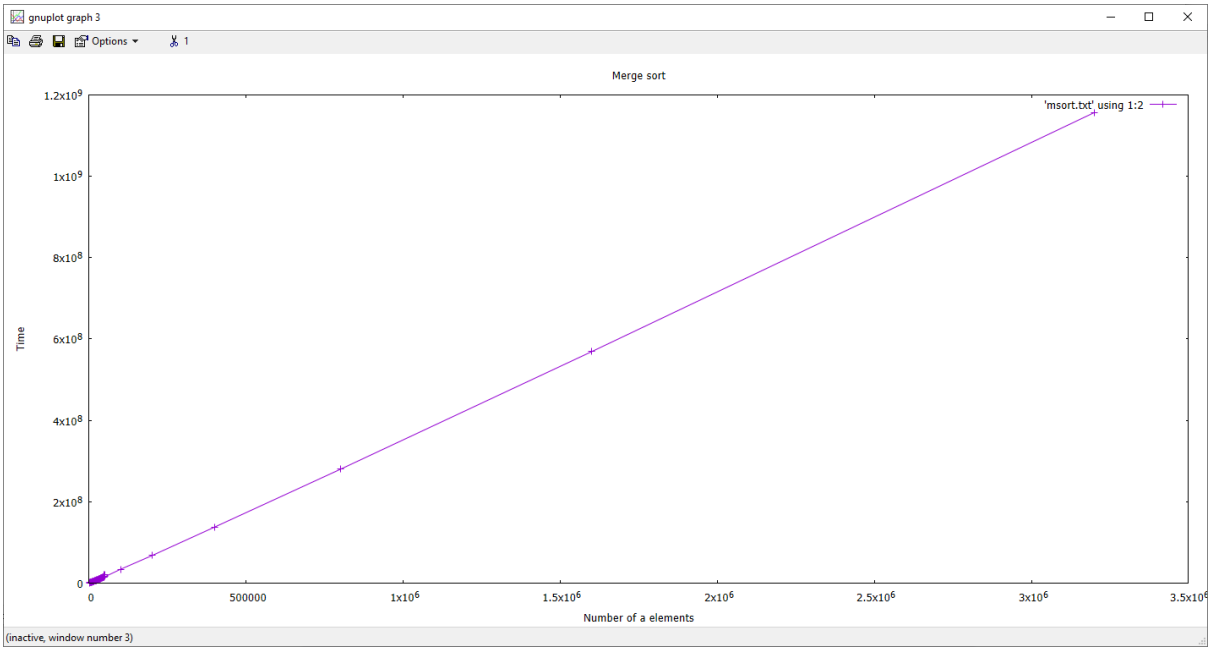
Быстрая сортировка



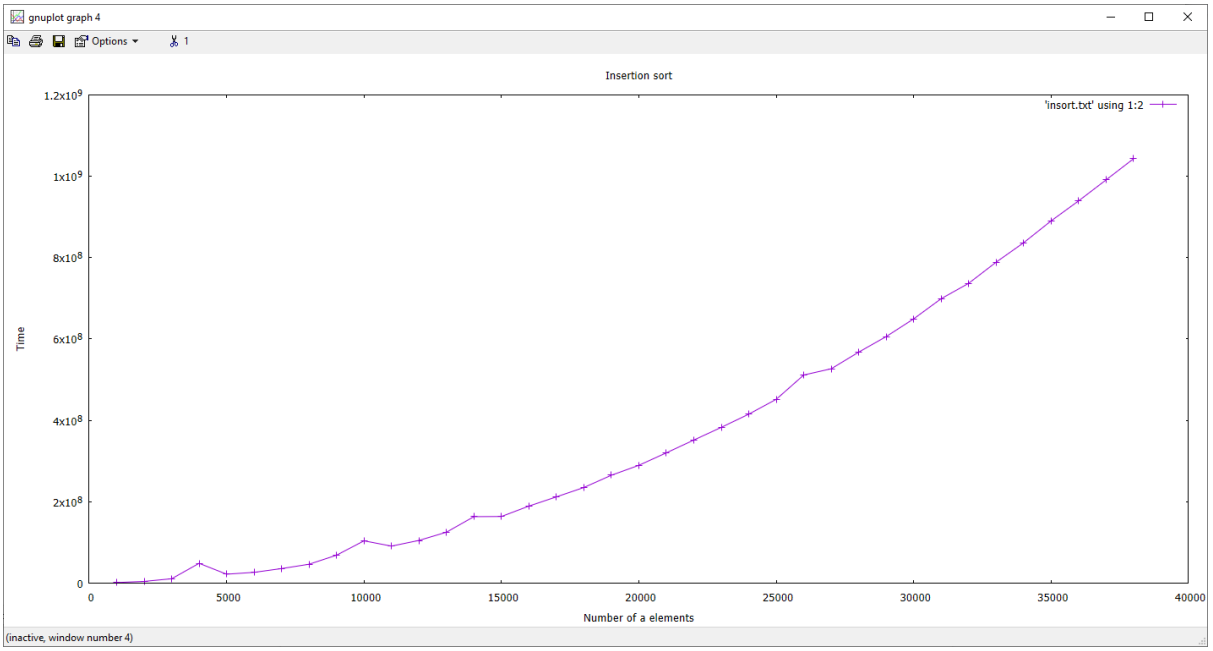
Сортировка пузырьком



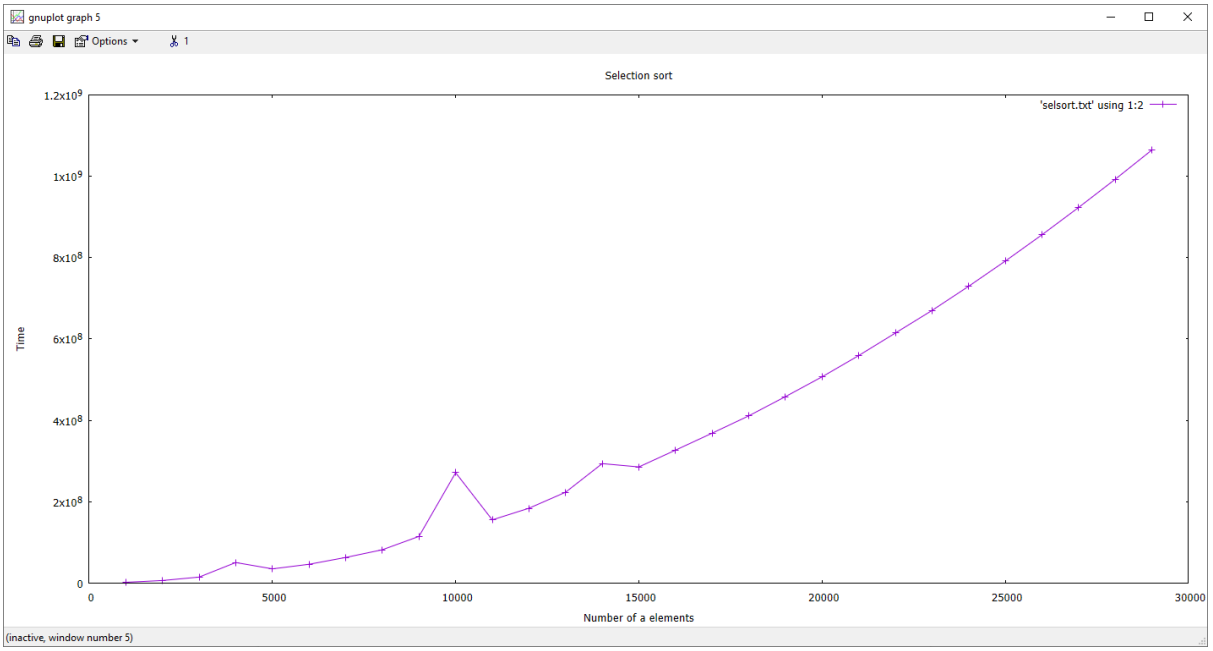
Сортировка слиянием



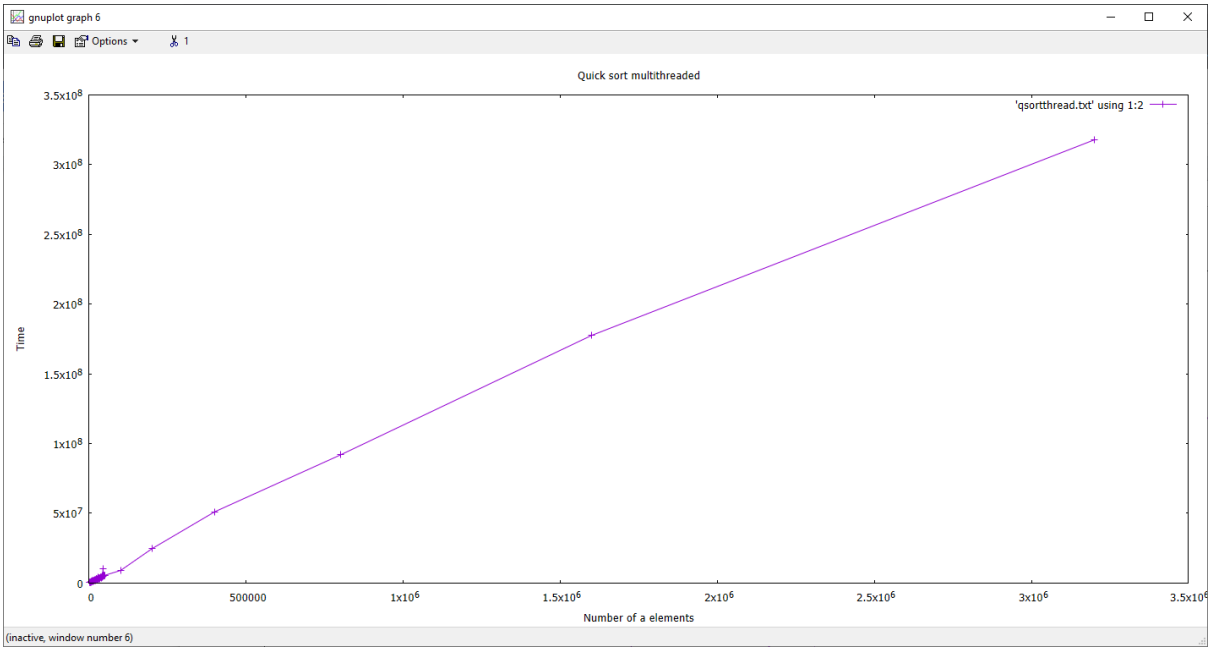
Сортировка вставками



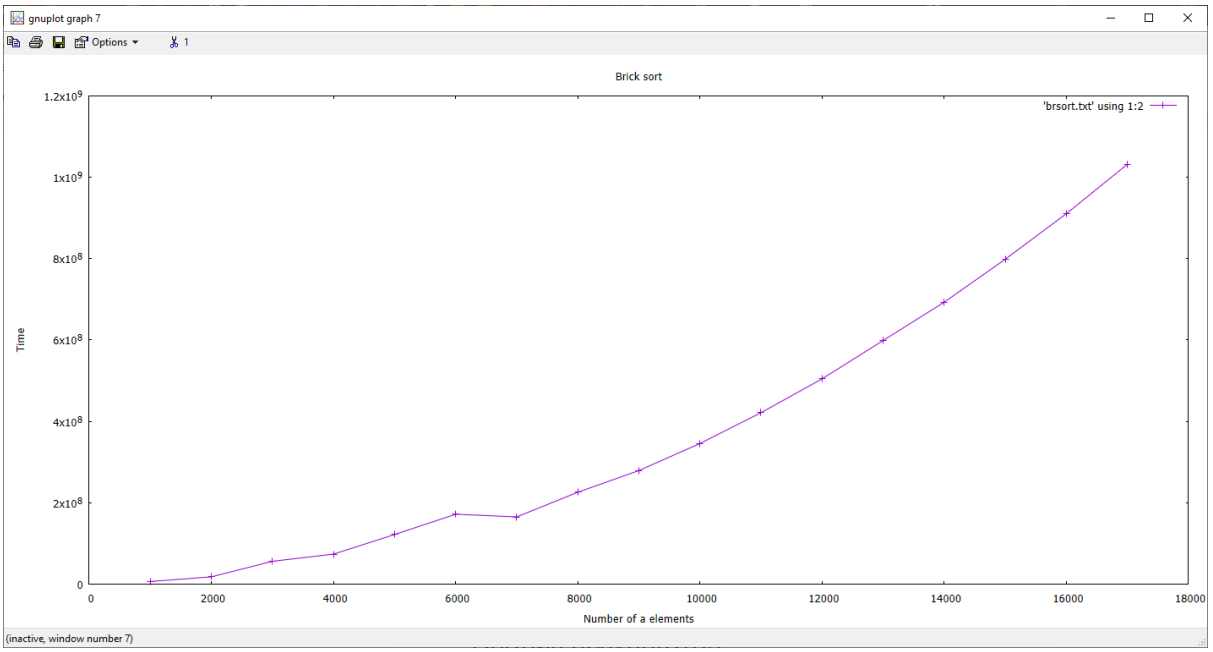
Сортировка выбором



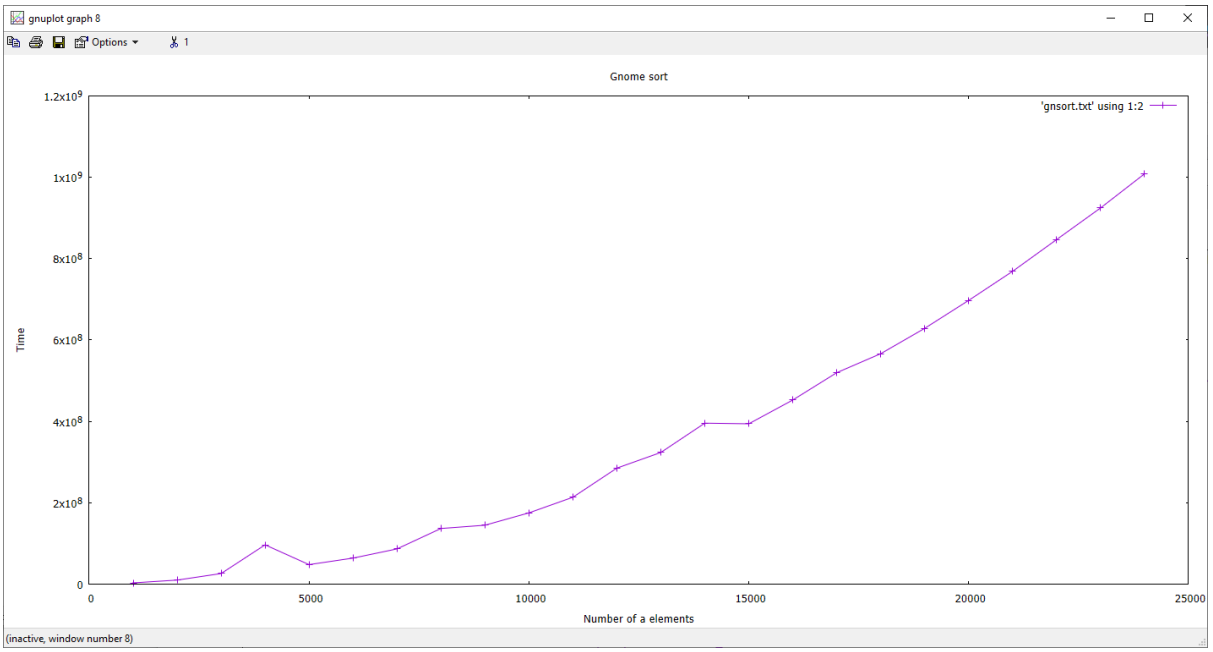
Быстрая сортировка многопоточная



Сортировка чёт-нечёт



Гномья сортировка



Анализ результатов

На малых значениях массива невозможно увидеть разницу из за недостаточной точности измерения. Однако если рассматривать Скорость работы сортировок на больших размерах массива то результаты получаются такими:

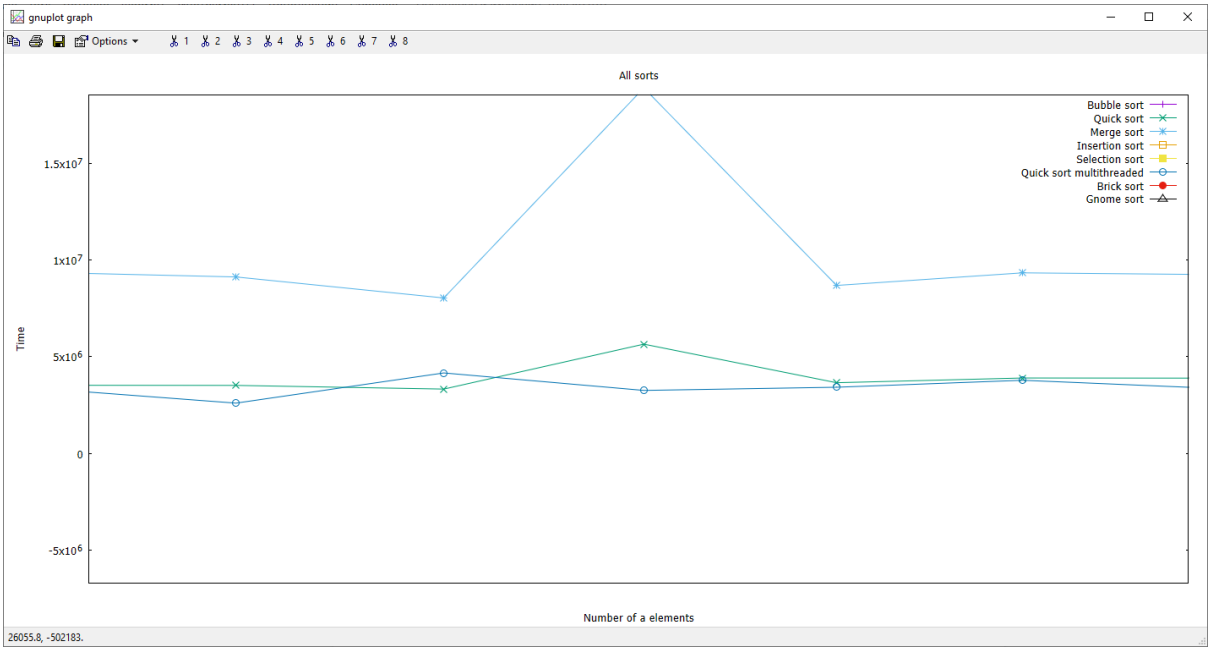
Список от самой медленной сортировки к самой быстрой:

1. Сортировка пузырьком
2. Сортировка чёт-нечёт
3. Гномья сортировка
4. Сортировка выбором
5. Сортировка вставками
6. Сортировка слиянием
7. Быстрая сортировка
8. Быстрая сортировка многопоточная

Примечания:

Быстрая многопоточная сортировка хоть и быстрее чем обычная, но не достаточно сильно. Это связано с неидеальносьтью нахождения опорного челена, который в некоторых случиях приводит к тому что одни поток выполняет сортировку, а второй простаивает. А так как время затрачиваемое на создание потока не маленькое, в некоторых случаях многопоточная сортировка будет выполняться дольше чем стандартная, хоть такие случаи и маловероятный.

Пример подобного случая



.

Дополнительный тест в режиме (от аккумулятора)