Федеральное государственное бюджетное образовательное учреждение

высшего образования «Сибирский государственный университет телекоммуникаций и информатики»

Факультет ИВТ

Кафедра вычислительных систем

**Курсовая работа**

на тему «Обработка последовательной информации»

Вариант 1.1 «Сортировка методом пузырька, быстрая сортировка»

Выполнил:студент гр. ИВ-321Девятайкин П.С.

Проверил:к.т.н., доцент Кафедры ВС

Ефимов А.В.

Новосибирск, 2024

**Оглавление**

**Тема курсовой работы**

Сортировка методом пузырька, быстрая сортировка.

**Задание на курсовую работу**

Реализовать динамическую библиотеку сортировок. Алгоритмы сортировок выбираются в соответствии с вариантом задания. Проанализировать эффективность алгоритмов сортировки. Разработать демонстрационную программу, использующую созданную библиотеку.

**Критерии оценки**

* **Оценка «удовлетворительно»:** алгоритмы реализованы в виде простой программы без применения библиотек, данные поступают на вход с клавиатуры. Тесты проведены только на небольших последовательностях, нет анализа и сравнения алгоритмов. Не предусмотрено динамическое выделение памяти под входные данные.
* **Оценка «хорошо»:** работа выполнена в полном соответствии с заданием. Обязательно динамическое выделение памяти под входные данные.
* **Оценка «отлично»:** помимо выполнения условий задания предусмотрена сортировка произвольных данных (по аналогии с функцией qsort библиотеки GNU C Library – GLibC). Обязательно динамическое выделение памяти под входные данные.

**Указания к выполнению задания**

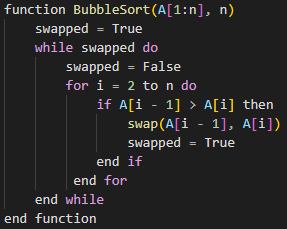
Алгоритмы сортировки необходимо реализовать в подпрограммах. Подпрограммы выносятся в отдельную библиотеку, которая компилируется как динамическая. Информация о создании и использовании динамических библиотек может быть найдена на ресурсе FirstSteps: <http://firststeps.ru/linux/general1.html>.

Эффективность сортировок оценивать по времени работы алгоритмов. По полученным результатам сформулировать выводы о преимуществах и недостатках каждого алгоритма. Сравнить полученные результаты с теоретическими оценками вычислительной сложности реализованных алгоритмов.

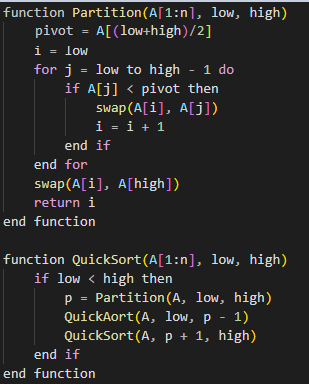
Экспериментальные измерения необходимо провести как для упорядоченных данных (по возрастанию и по убыванию), так и случайных последовательностей, размер которых составляет 28 – 215 элементов (с некоторым шагом). Построить графики полученных зависимостей. В случае, если время работы одного из алгоритмов превышает 15 мин., прекратить измерения по данному алгоритму и строить график не на всем интервале. **Анализ задачи**

1. В данной работе операции будут проводиться над целочисленным типом данных (int), который будет храниться в динамическом массиве.

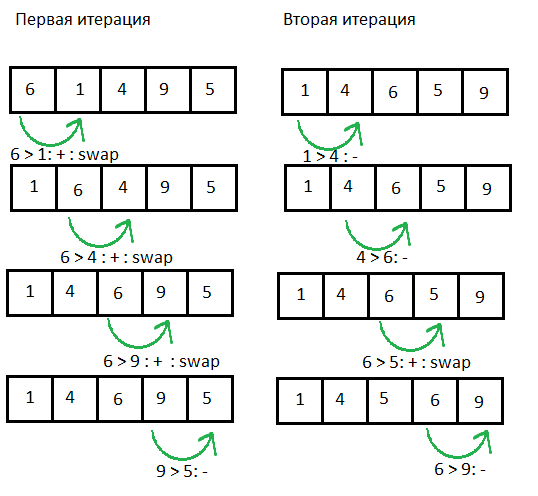
2. Псевдокод сортировки методом пузырька

****

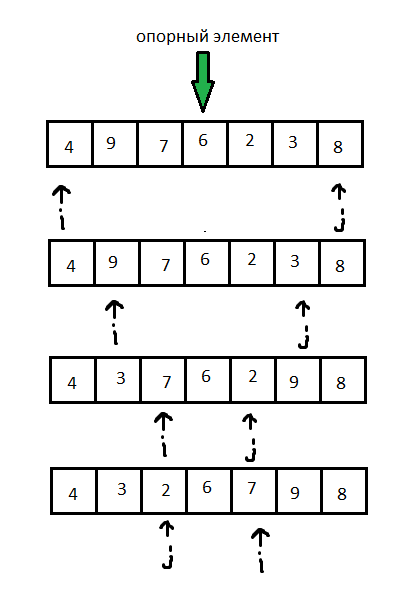
Псевдокод быстрой сортировки

****

Пример сортировки методом пузырька



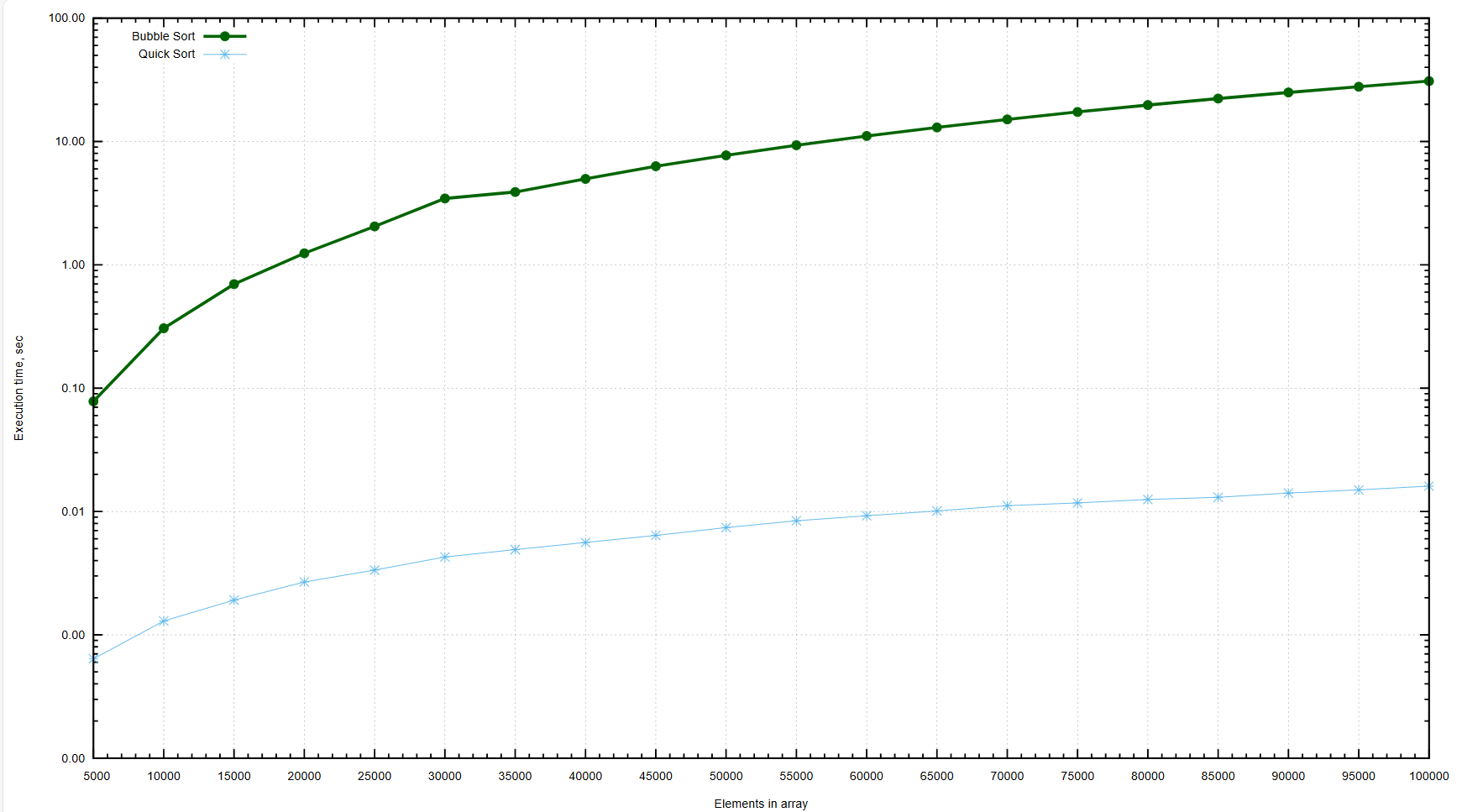
Пример сортировки методом быстрой сортировки



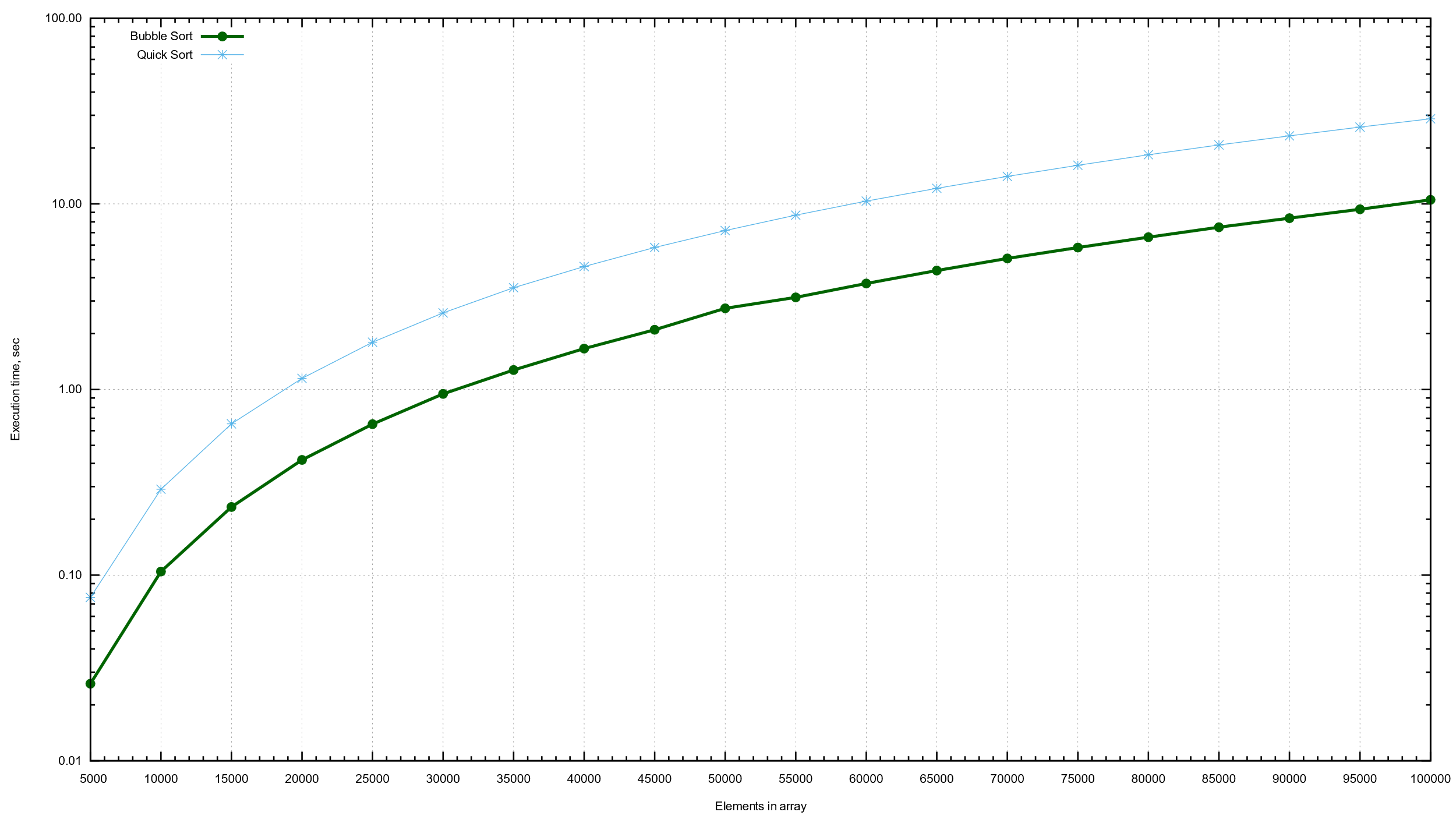
**Тестовые данные**

Было выполнено 3 теста, в каждом из которых сортировалась последовательность из элементов размером от 5000 до 100000 с шагом в 5000 элементов. Первый тест: программа выполнялась для случайной последовательности данных. Второй тест: программа выполнялась для данных упорядоченных по не убыванию. Третий тест: программа выполнялась для данных упорядоченных по не возрастанию. Для каждого теста были построены графики.

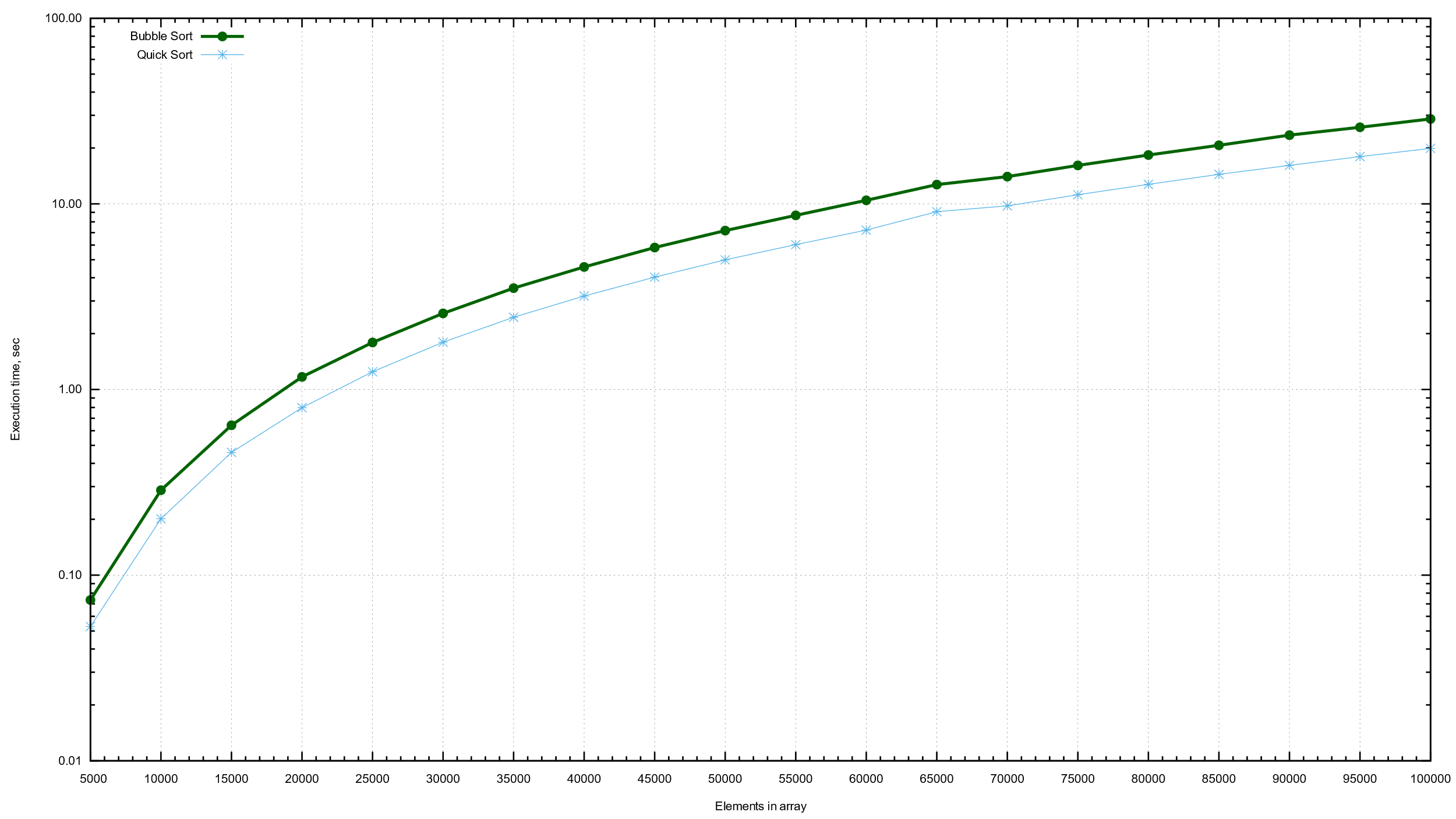
Тест 1. Случайная последовательность данных.

****

Тест 2. Данные упорядочены по не убыванию.



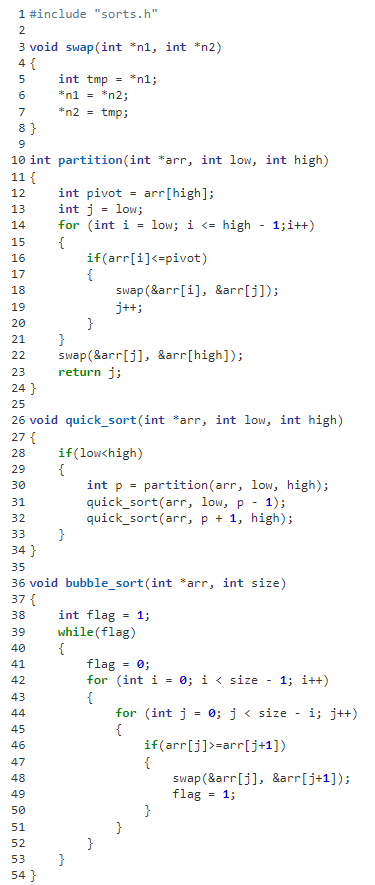
Тест 3. Данные упорядочены по не возрастанию.

****

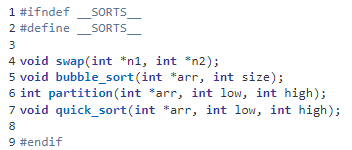
Проанализировав полученные данные, было установлено, то что метод сортировки пузырьком эффективен только в случае, если данные уже отсортированы.

**Листинг программы**

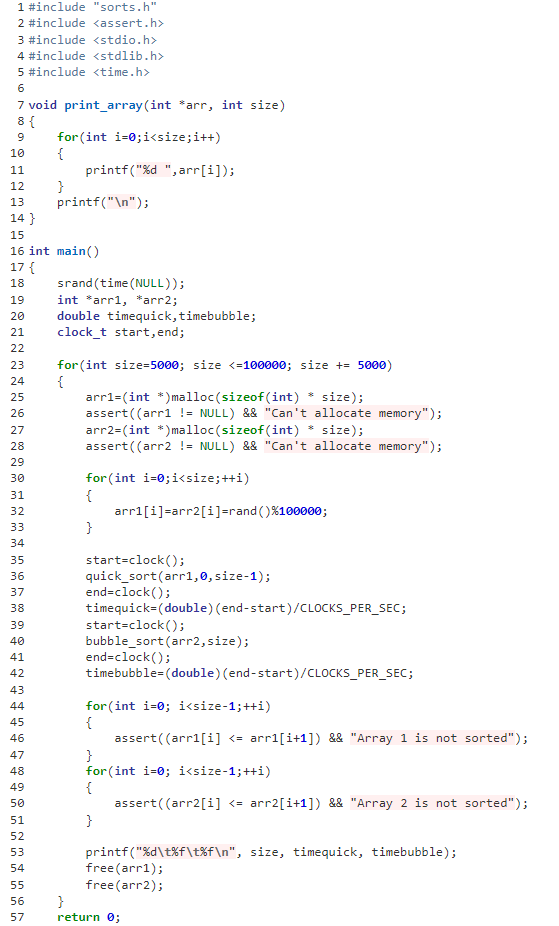
sorts.c - сортировки



sorts.h – заголовочные файлы



main.c – тестирование сортировок.



Для того чтобы получить динамическую библиотеку, необходимо скомпилировать код со следующими опциями:

gcc -Wall -g -shared -fpic -o libsorts.so sorts.c

Чтобы подключить библиотеку, нужно скомпилировать исполняемый файл, указав название библиотеки через опцию -l.

gcc -Wall -g -o main main.c -lsorts  
Получаем ошибку, т.к. линкер говорит, что не знает где лежит наша библиотека. Мы можем указать текущую директорию в которой лежит наша библиотека с помощью опции -L.

При попытке запуска программы ловим еще одну ошибку в которой говорится, что в процессе загрузки динамических библиотек отсутствует наша библиотека. По умолчанию в операционной системе есть некоторое количество стандартных директорий, где должны располагаться библиотеки. Так же есть возможность задавать дополнительные директории с библиотеками с помощью переменной окружения **LD\_LIBRARY\_PATH**. Добавим в **LD\_LIBRARY\_PATH** текущую директорию.

export LD\_LIBRARY\_PATH="$LD\_LIBRARY\_PATH:$PWD"

Видим, что наша библиотека подгрузилась. И теперь программа запускается и работает.