Министерство науки и образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе №1
по курсу «Логика и основы алгоритмизации
в инженерных задачах»
на тему «Оценка времени выполнения программ»

Выполнил:
студент группы 23ВВВ4
Костин К. А.

Приняли:
Демеев М. В.
Юрова О. В.

Пенза 2024

Общие сведения.

Для оценки времени выполнения программ языка Си или их частей могут использоваться средства, предоставляемые библиотекой **time.h**. Данная библиотека содержит описания типов и прототипы функций для работы с датой и временем.

Цель:

Научиться определать порядок сложности программы и оценивать время выполнения программы

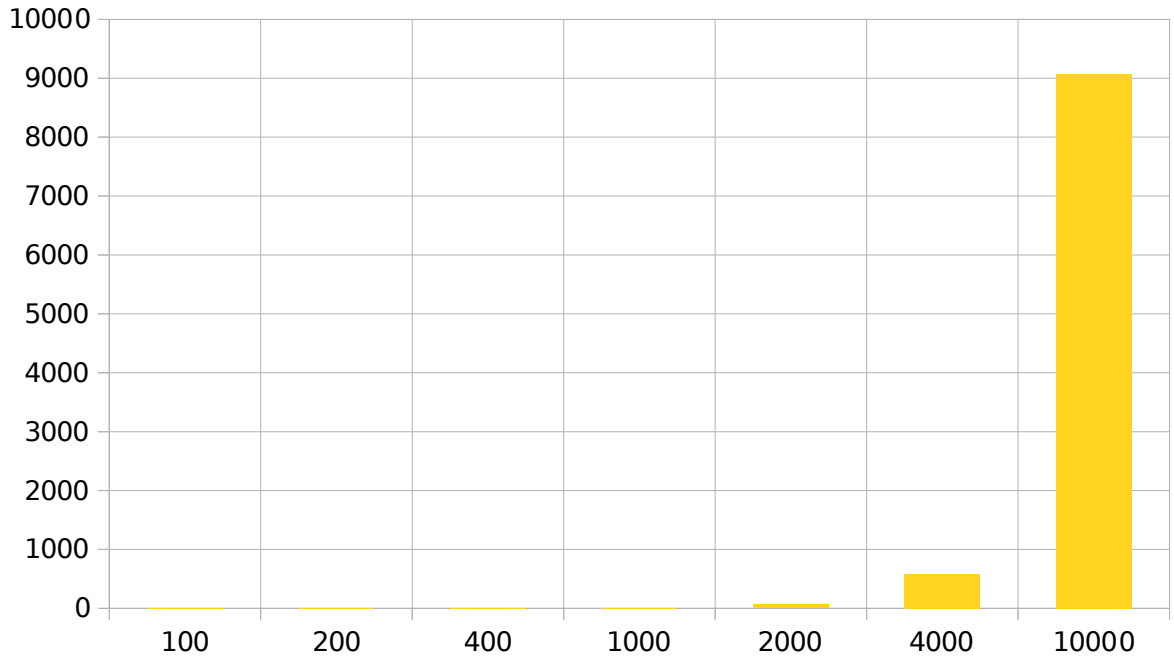
Задание 1:

1. Порядок сложности программы = O(n^3), так как в программе присутствует тройная вложенность циклов

2. Время исполенения программы должно увеличиваться кубически от увеличения размера матрицы

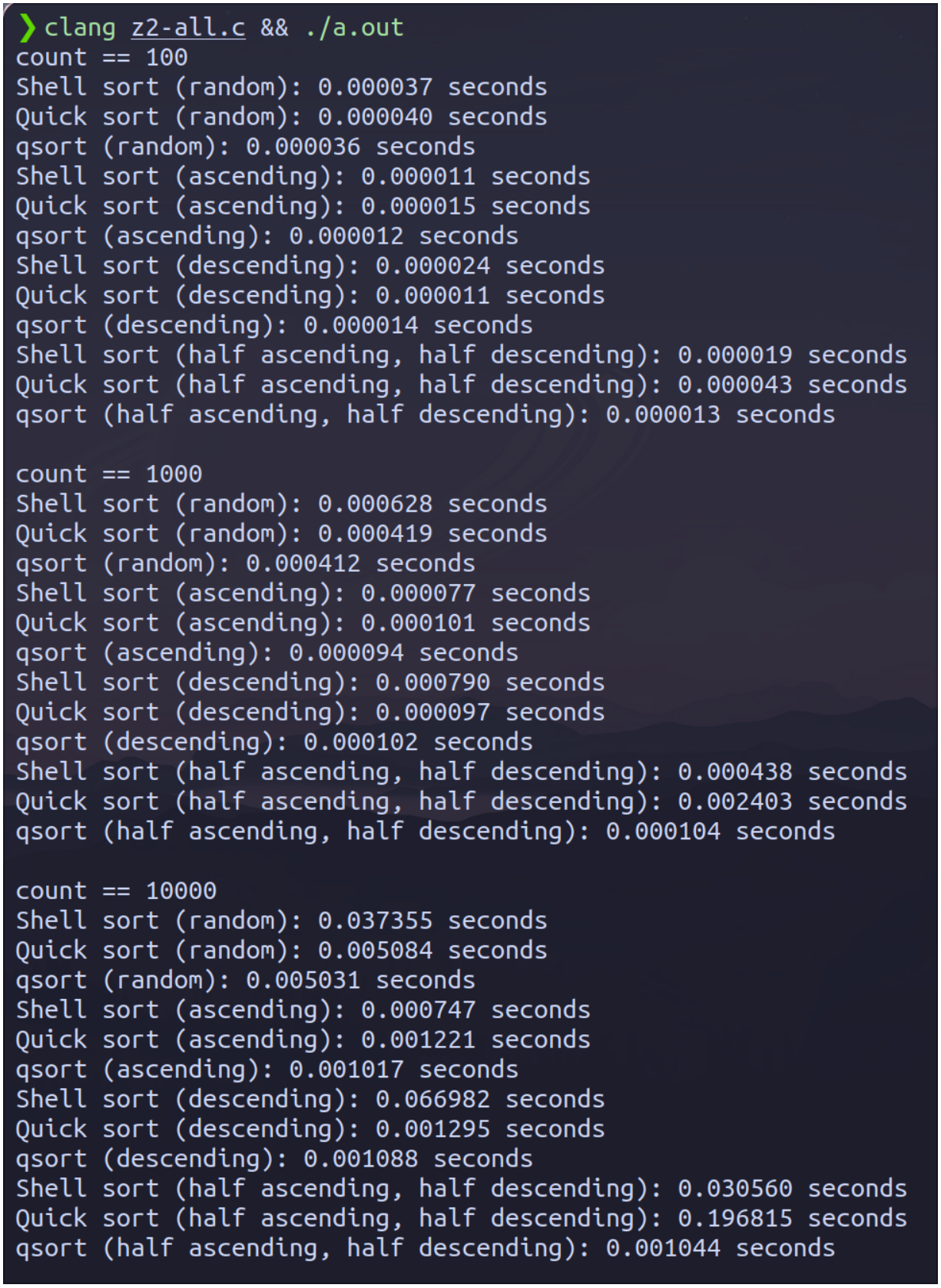
3. Результат работы программы совпал с ожидаемым результатом

```
❯ clang z1-3.c && ./a.out
Size: 100, Time spent: 0.002007 seconds
Size: 200, Time spent: 0.025941 seconds
Size: 400, Time spent: 0.189489 seconds
Size: 1000, Time spent: 3.340552 seconds
Size: 2000, Time spent: 63.988189 seconds
Size: 4000, Time spent: 580.806819 seconds
Size: 10000, Time spent: 9075.106546 seconds
```



Задание 2:

Все сравнения приведены на скриншете ниже:



Исходя из полученных данных, можно сделать несколько выводов о производительности различных алгоритмов сортировки (Shell sort, Quick sort и qsort) на разных типах данных и размерах массивов:

1. Shell sort:

Случайные данные: Shell sort показывает хорошую производительность на маленьких массивах (100 элементов), но значительно замедляется на больших массивах (1000 и 10000 элементов).

Возрастающие данные: Shell sort очень быстрый на возрастающих данных, независимо от размера массива.

Убывающие данные: Shell sort показывает хорошую производительность на маленьких массивах, но замедляется на больших массивах.

Полувозрастающие/полуубывающие данные: Shell sort показывает среднюю производительность, лучше, чем на случайных данных, но хуже, чем на возрастающих данных.

2. Quick sort:

Случайные данные: Quick sort показывает хорошую производительность на всех размерах массивов, особенно на больших массивах.

Возрастающие данные: Quick sort также показывает хорошую производительность, но немного медленнее, чем Shell sort на возрастающих данных.

Убывающие данные: Quick sort показывает хорошую производительность, сравнимую с возрастающими данными.

Полувозрастающие/полуубывающие данные: Quick sort значительно замедляется на таких данных, особенно на больших массивах.

3. qsort:

Случайные данные: qsort показывает хорошую производительность на всех размерах массивов, сравнимую с Quick sort.

Возрастающие данные: qsort также показывает хорошую производительность, но немного медленнее, чем Shell sort на возрастающих данных.

Убывающие данные: qsort показывает хорошую производительность, сравнимую с возрастающими данными.

Полувозрастающие/полуубывающие данные: qsort показывает хорошую производительность, значительно лучше, чем Quick sort на таких данных.

Выводы:

1. Shell sort хорошо подходит для возрастающих данных и маленьких массивов, но плохо справляется с большими массивами случайных данных.

2. Quick sort показывает стабильную хорошую производительность на случайных, возрастающих и убывающих данных, но значительно замедляется на полувозрастающих/полуубывающих данных.

3. qsort показывает стабильную хорошую производительность на всех типах данных и размерах массивов, что делает его универсальным выбором для сортировки.

Вывод: в ходе этой лабораборной работы были получены навыки определения сложности программы и оценки времяни работы программы