

ФЕДЕРАЛЬНОЕ АГЕНСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ
СООБЩЕНИЯ Императора Александра I»

Кафедра «Информационные и вычислительные системы»

Дисциплина «Структуры и алгоритмы обработки данных»

ОТЧЁТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

Выполнил студент
Факультет: АИТ
Группа: ИВБ-211

Шефнер А.

Проверил:

канд. ист. наук доц. Забродин Андрей Владимирович

Санкт-Петербург

2023

Оценочный лист результатов ЛР № 1

Ф.И.О. студента _____ Шефнер Альберт _____

Группа _____ ИВБ-211 _____

№ п/п	Материалы необходимые для оценки знаний, умений и навыков	Показатель оценивания	Критерии Оценивания	Шкала оценивания	Оценка
1	Лабораторная работа№	Соответствие методике выполнения	Соответствует	7	
			Не соответствует	0	
		Срок выполнения	Выполнена в срок	2	
			Выполнена с опозданием на 2 недели	0	
		оформление	Соответствует требованиям	1 0	
			Не соответствует		
	ИТОГО количество баллов			10	

Доцент кафедры

«Информационные и вычислительные
системы»

«__» _____ 2023 г.

Забродин А.В.

Цели работы:

- Освоить основные алгоритмы сортировки

Задание

Разработать и реализовать следующие алгоритмы сортировок:

- Сортировка выбором
- Пузырьковая сортировка
- Сортировка вставками
- Сортировка слиянием
- Пирамидальная сортировка
- Быстрая сортировка
- лексикографическая (входными данными может быть журнал, где сделать сортировку по имени, отчеству, n-ой букве фамилии и т.д)

Сайт (Архив погоды с 1929 года (pogoda-service.ru)) с которого каждому студенту выдаётся массив по 6(по одной на каждую сортировку) странам за год.

В отчёте нужно представить диаграмму, отображающую скорость каждой сортировки. Данные брать из файла. Для каждого алгоритма вычислить его скорость в O-символике

Используемые средства

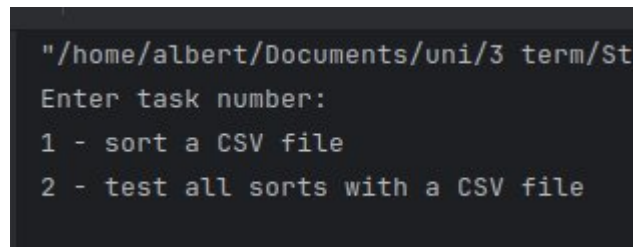
В качестве интегрированной среды разработки использовалась JetBrains CLion. Для работы в консоли с потоками ввода-вывода использовалась стандартная библиотека `<iostream>`. Для поддержки функционального программирования использовалась библиотека `<functional>`

Исходный код

Исходный код программы доступен по ссылке:

<https://github.com/n0emo/uni/tree/main/3%20term/Structures%20and%20Algorithms/Lab%201>

Поведение программы



```
"/home/albert/Documents/uni/3 term/St
Enter task number:
1 - sort a CSV file
2 - test all sorts with a CSV file
```

При запуске программы вы увидите небольшое меню:

- 1 – отсортировать CSV файл и записать результат в указанную директорию.
- 2 – Протестировать все функции сортировки на время по указанному CSV файлу.

Оценки сложности алгоритмов

1. **Сортировка выбором:** выбор элемента методом линейного поиска требует прохождения по всему массиву, что имеет сложность $O(n)$. Далее найденный элемент вставляется в нужную позицию за константное время $O(1)$. Эти две операции производятся для каждого элемента, поэтому итоговая сложность алгоритма: $O(n^2)$.
2. **Пузырьковая сортировка:** этот алгоритм включает в себя два вложенных друг в друга цикла, количество итераций которых линейно растёт с количеством элементов массива, поэтому итоговая сложность алгоритма: $O(n^2)$
3. **Сортировка вставками:** для каждого из элемента массива производится вставка в нужную позицию, которая имеет сложность $O(n)$, итоговая сложность алгоритма: $O(n^2)$.

4. **Сортировка слиянием:** делит массив на две равные части и сортирует их, после чего сливает обратно. Слияние двух отсортированных массивов имеет сложность $O(n)$, а производится оно будет $\log_2(n)$ раз (поскольку массив делится в 2 раза). Итоговая сложность: $O(n \cdot \log(n))$.
5. **Пирамидальная сортировка:** построение первоначальной максимальной кучи имеет сложность $O(n \cdot \log(n))$, далее n раз происходит перестановка и возврат максимальной кучи со сложностью $O(\log(n))$. Итоговая сложность алгоритма: $O(n \cdot \log(n))$.
6. **Быстрая сортировка:** разделение массива на 2 части имеет сложность $O(n)$. Таких разделений в среднем $\log_2(n)$ раз. Итоговая сложность алгоритма: $O(n \cdot \log(n))$.

Результаты тестов

Тесты проводились на CSV файлах people-100.csv, people-10000.csv и people-100000.csv. Вы можете найти их в репозитории проекта. Информация о ПО и оборудовании:

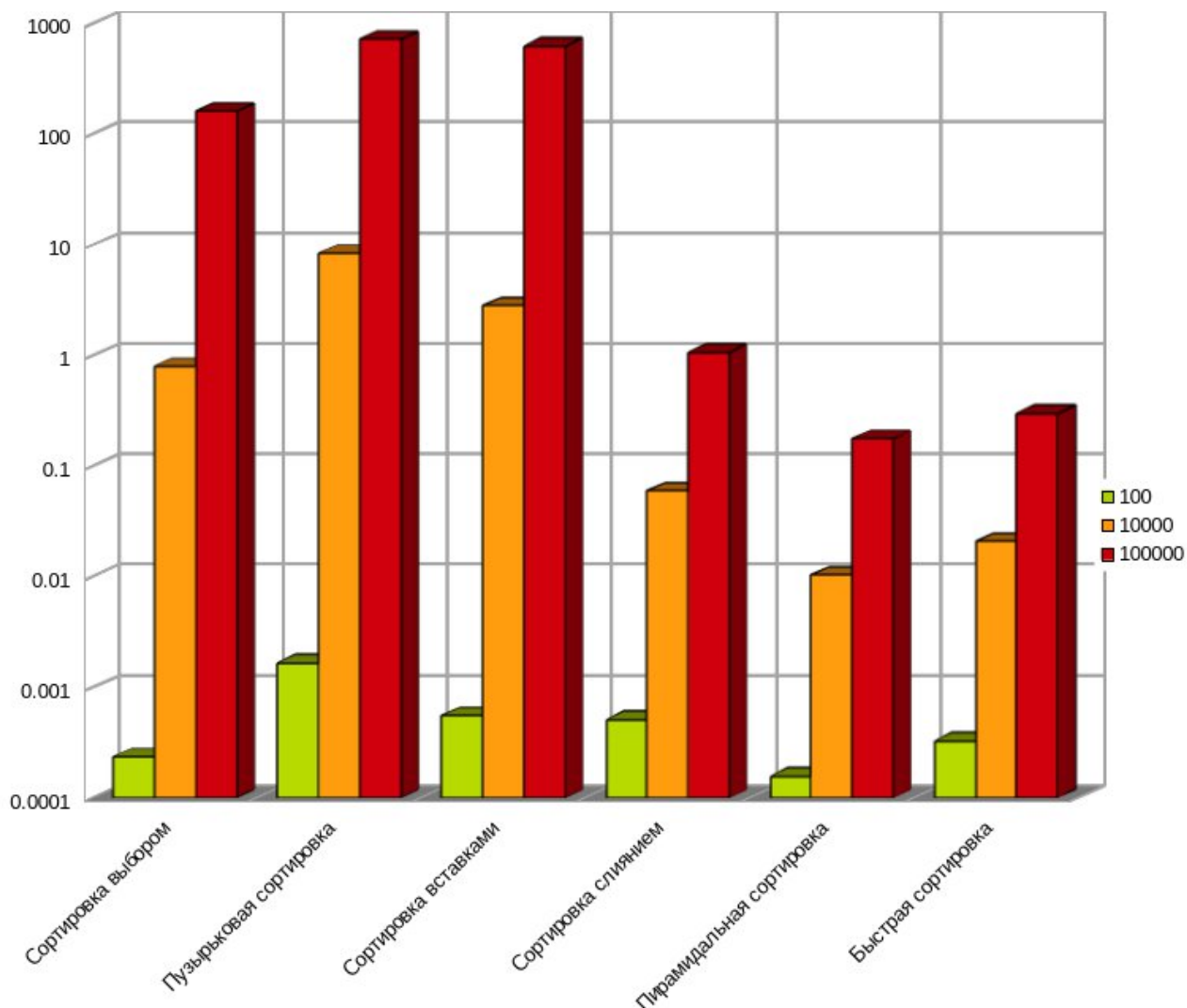
- Компилятор: LLVM
- Флаг оптимизации: -O3
- Процессор: AMD Ryzen 4700U
- ОС: Fedora Linux 38

Время, затраченное на сортировку

Алгоритм сортировки	Количество записей		
	100	10000	100000
Сортировка выбором	0.000234951 с	0.794204 с	160.005 с
Пузырьковая сортировка	0.00164801 с	8.40049 с	721.168 с
Сортировка вставками	0.000549454 с	2.77752 с	624.194 с
Сортировка слиянием	0.000508176 с	0.0589581 с	1.0649 с
Пирамидальная сортировка	0.000156181 с	0.0101833 с	0.177641 с
Быстрая сортировка	0.000324629 с	0.0205406 с	0.298199 с

Диаграмма

Диаграмма затраченного времени на сортировку в логарифмической шкале.



Вывод

Я изучил различные алгоритмы сортировки. Для моих данных и моего способа хранения наиболее быстрым алгоритмом оказался алгоритм пирамидальной сортировки. Это связано с тем, что перестановка двух записей при моём способе хранения это довольно дорогая операция, а пирамидальная сортировка совершает меньше всего перестановок. Так же, возможно, сыграли роль оптимизации LLVM, так как действительно сильно пирамидальная сортировка оторвалась от быстрой сортировки и сортировки слиянием только после компиляции в Release режиме с флагом оптимизации -O3.