laSalle
RAMON LLULL UNIVERSITY

Advanced Programming and Data Structure

Project 1 of the First Semester - Recursive Sorting

Zaballos Zapatery

Digital Engineering School
La Salle - Ramon Llull University

CONTENTS

1 – INTRODUCTION

We have received a commission by the ancient and avowed "Zaballos Zapatery", a globally renowned soe shop that sells Premium quality footwear exclusively through their perfected local area network.

Specifically, they have contacted us because they're in the process of opening their first brick and mortar store and need a series of algorithms to organize their warehouse. However, the shop's owner is very meticulous when it comes to wasting resources and producing perfectly optimized designs, which is why we have been tasked with an initial investigation.

In short, for now we have been asked to analyse how different algorithms behave when it comes to future functionalities.

2 – Data Format

To make this task easier, the owner has provided us with some datasets that contain information regarding products currently being sold in the store. Each dataset is a text file with the following structure:

```
524288
Satterfield, Goldner and Orn Mike Raffone;113,59;23;52;2484;3,7
Thompson, Heller and Will Kent Cook;288,96;24;44;2271;4,5
Skiles-Rodriguez Rusty Keyes;188,77;20;38;2213;0,6
```

- **n_shoes:** Number of shoes in the file, ending in '\n'
- **shoe_1:** Ending in '\n'. Its data is separated by ';' and consists of the following fields:
    - **name:** String with the shoe's name.
    - **price:** Real storing the shoe's price in euros.
    - **min_size:** Integer storing the minimum shoe size sold for this model.
    - **max_size:** Integer storing the maximum shoe size sold for this model.
    - **weight:** Integer storing the shoe's weight in grams.
    - **score:** Real between 0 and 10 storing the average score that customers gave the shoe model via their reviews.
- **shoe_2:** Ending in '\n'
- ...
- **shoe_N:** Ending in '\n'

3 – FUNCTIONALITIES

Any shop needs their products to be properly organized. With this being the first brick and mortar store of the renowned Zaballos Zapatery, the owner expects the products to be restocked multiple times per day. Thus, we need the best possible algorithm to organize their products.

3.1 – Iterative sorting by name

The shop owner knows that, generally, smaller shoe producers send their datasets in an unsorted fashion, while larger brands already provide them with pre-sorted datasets (thanks to the efforts of their interns).

In this section the shoes will be alphabetically (A-Z) sorted by name. This will be achieved by implementing two of the simpler sorting algorithms: **Insertion Sort** and **Selection Sort**.

To compare their efficiency in different scenarios, we are provided with three different datasets:

- **ascending:** Alphabetically sorted dataset, by ascending names (A-Z).
- **descending:** Alphabetically sorted dataset, by descending names (Z-A).
- **random:** Unsorted dataset.

3.2 – Special recursive sorting

Next, we have been tasked with finding a new algorithm to design a new way to sort shoes, this time with the explicit goal of maximizing sales.

Given that we are interns and have no role in the decision-making process, we have been “asked” to implement and analyse the **Quick Sort** and **Merge Sort** algorithms. We must also design a way of combining **3 or more characteristics** of the shoes to create a custom sorting criterion.

To compare their efficiency we can use the **random** dataset.

4 – REQUIREMENTS

This section presents the project's main goals and describes the results expected from its realization.

4.1 – Objectives

This project has two main goals. On one hand, it serves as an opportunity to implement the recursive sorting algorithms studied in class, as well as an invitation to research other concepts in the same field.

On the other hand, this project also serves as an introduction to performance analysis techniques, meaning that it's important to carry out performance **comparisons** between the different algorithms, and to include their results in the report. Similarly, it's also important to make sure that any generated charts are mathematically correct.

4.2 – Code

The submitted project must correctly implement the different algorithms, but there are no restrictions in terms of user interaction. You can also leave any mechanisms for performance evaluation that you may have used in your code (for instance, the measure of execution time).

Regardless, it's recommendable that you **structure** your code in a way that's easy to modify in terms of behaviour (or the functionality being executed). This can be achieved in many ways, such as defining command line arguments, coding a small menu, using constants, commenting out code blocks in the main procedure...

You must use the provided **datasets**. Additionally, the solution to each problem must be logged to console in a clear and understandable format.

It's highly recommendable to **comment** the code at points that you consider to be relevant.

4.3 – Report

The report must be written in **formal tone**. It has the same importance or more than the implementation of the project itself, as it reflects the knowledge acquired with the project and in the subject. Therefore, its content should **prioritize quality over quantity**.

On a technical level, the use of LaTeX to write the report is proposed, but it won't influence the obtained grade.

The report should have the following sections, or an equivalent set that covers the same contents:

- **Cover** (with the group code and its members' full names and logins).
- **Numbered index**.
- Justification for the choice of **programming language**, with the corresponding advantages and possible disadvantages. The correctness of the arguments will be evaluated.
- Explanation of **how the algorithms operate**, with special emphasis in those that require of additional design or research by the students.
- **Result analysis** by using the provided datasets, including explanations about how the measurements where obtained. Relevant comparisons should be made between algorithms, and formally correct charts to support the analysis should be generated.
- **Observed problems** and their solutions.
- Total **dedication** in hours. Include a thematic breakdown into categories, for instance: comprehension, research, design, implementation, result analysis, documentation...
- **Conclusions**, both on a personal and (mainly) on a technological level.
- **Bibliography** following the ISO 690 or APA 7th standard.

The result analysis section is critically important to pass any project in this subject and will determine a substantial part of your mark. You must be capable of making correct performance measurements, as well as identifying relevant comparisons from them, and extracting substantial conclusions.

In order to analyse the cost of the algorithms with the provided datasets, it's recommended that you take measurements for subsets of them. For instance, a performance measurement can be taken every 100 elements, that is: Sorting the first 100 elements, then the first 200 elements, after that the first 300 elements, etc.

5 – CONSIDERATIONS

This section points out a series of details regarding the project and its development.

5.1 – Programming language

The project can be implemented in any programming language, to be chosen by the group. If it's developed in C, it must compile, execute, and work properly on the university servers (specifically, `matagalls`).

5.2 – Groups

The project must be developed in pairs. Groups must be established via the corresponding tool in the eStudy before the **29th of October at 23:55**.

Students that don't belong to any group won't be able to submit their work. The mechanisms to split groups are described in the subject's project regulations, which can be found in the Syllabus for the current academic year.

5.3 – Atlassian tools

Once this document is presented, and after a small amount of time for possible group modifications, you will be assigned a repository in the university's Bitbucket server for your git-based version control. Independently of the chosen language and the group's size, **you must regularly use it during the project's development**.

If you want to use other Atlassian tools such as Jira or Confluence, you can ask the subject's professors for them via email.

Note: Being responsible when using the tools at our disposal is an essential part of engineering. Thus, we ask you to **NOT** include datasets in the version control system. Be aware that you can use a `.gitignore` file to configure exceptions.

5.4 – Fraud detection

Academic fraud prevents students from learning and denotes a lack of respect towards classmates who have spent time and efforts in their work.

This project is classified as a **highly important academic activity**. Fraud attempts, whether partial or total, including plagiarism from classmates or from the internet, will be considered a premeditated action. Therefore, applying the university's copies regulation, it will be considered a **very serious misconduct**.

The subject's **AI regulations** forbid the use of AI-based generative tools during the development of this project, including in code and in the report. Consequently, their use will be considered academic fraud.

6 – DELIVERY

This section describes everything having to do with the project's delivery, from the format to the deadline and retake mechanisms.

6.1 – Format

Two different files must be submitted to the corresponding eStudy deliverable:

- The report in **PDF** format, including the contents described in chapter 4.3 of this document.
- A **ZIP** file containing:
    - `README`: Text file in **TXT** or **MD** format where **ALL** the necessary steps to test your code are explained in the most detailed way possible. For instance, it's recommended that you include information regarding the used language, its version, the IDE you use, instructions to compile / run, usage instructions…
    - Folder with the code or project structure generated by the IDE. **It must match the contents of your bitbucket repository**. Submitting a ZIP file inside a ZIP file is unnecessary and inefficient. We also ask you **not** to include the datasets in your submitted project.

Submissions where the report is only included inside the ZIP won't be accepted.

6.2 – Deadline

The project will be presented after establishing the knowledge base to start implementing it, and must be submitted before the **November 26th, 2023, at 23:55h** deadline. No submission will be accepted past this point.

6.3 – Retake

If the deadline is missed (NP) or the task is failed, the project can be submitted again with a maximum grade of 8 before the **January 14th, 2024, at 23:55h** deadline.

The last chance will be during the extraordinary call, before the **June 30th, 2024, at 23:55h** deadline, and with a maximum grade of 5.