Analysis of Algorithms

BLG 335E

Project 2

Res. Assist. Mustafa İzzet Muştu
mustu18@itu.edu.tr

Res. Assist. Erdi Sarıtaş
saritas21@itu.edu.tr

Instructors:
Assoc. Prof. Dr. İlkay Öksüz
Asst. Prof. Dr. Mehmet Baysan
Asst. Prof. Dr. Tuğçe Bilen

Faculty of Computer and Informatics Engineering
Department of Computer Engineering
Date of submission: 24.11.2024

1. Project Brief

Eleanor is a dedicated antique collector preparing her prized artifacts for an upcoming international exhibition. She has spent years collecting rare and fascinating artifacts from all over the world. She's passionate about history and wants her collection to tell a story visitors will remember. But she has a problem—her collection is so massive and diverse that she needs help organizing it!

Eleanor wants the display to tell a story about each item's history and uniqueness. To make her display unique, Eleanor hopes to arrange her artifacts not only by their age but also by their rarity—showcasing ancient and rare pieces for maximum intrigue. Each artifact in Eleanor's collection has a few important characteristics:

- **Age:** Representing how many years have passed since it was created. Eleanor believes older items deserve a higher spot in the display, but she also wants to highlight the rarest of these artifacts.
- **Type:** Every item has a type (e.g., pottery, sculpture, jewelry) and an origin (e.g., Roman, Egyptian, Aztec). These will help determine how unique each artifact is within its historical context.
- **Rarity:** Eleanor wants to emphasize how unique each item is within its historical context. For example, a rare artifact from Ancient Egypt would stand out more if it were the only one in her collection from that era.

You'll need to use these traits to sort Eleanor's collection for display. But here's the catch: rarity is a bit complex to determine because it depends on other nearby items regarding age and uniqueness. That means you'll need to do things in two steps: sort by age to see which items are close together historically, and then determine how unique each item is based on a "near-age window."

2. Implementation Details

To help Eleanor bring her vision to life, you'll need to sort and analyze her collection based on age and rarity. Here's the plan:

Step 1-) Sort the Collection by Age Using Counting Sort

Begin by organizing the artifacts by age, from the oldest to the most recent. Since ages are limited to a range (no item is older than a few tens of thousand years), counting sorting is an efficient way to handle this.

- *Why counting sort?* Counting sort is ideal here because it quickly sorts numbers in a small range and doesn't rely on item comparisons.

Step 2-) Calculate Rarity Scores Using Age Windows (with a Probability Twist)

Now that the items are sorted by age, you can calculate how rare each is by comparing them to other items in a "near-age window." This is how Eleanor defines an artifact's rarity:

- For each item, look at other items within a certain age range (e.g., ±50 years). Calculate a "rarity score" by seeing how common that item's type and origin are within this range.
- Use probability to estimate how likely it is to find similar items in the age window—Eleanor wants you to think like a historian, assessing how often they appear together.

The rarity score calculation can be as follows;

$$R = (1 - P)(1 + \frac{\text{age}}{\text{age}_{max}}) \tag{2.1}$$

where the probability P is calculated as follows;

$$P = \begin{cases} \frac{\text{countSimilar}}{\text{countTotal}} & \text{if countTotal} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Step 3-) Sort by Rarity Using Heap Sort

Now that each item has a rarity score, Eleanor wants you to organize them again, but this time by rarity. Use heap sort to arrange items by rarity.

- *Why heap sort?* Heap sort will allow you to sort based on rarity while keeping things efficient.

3. Deliverables

3.1. Code [70%]

You are given a codebase that is quite similar to the first homework. You will use the docker container as instructed in the first homework. You can utilize the same commands to build, run and test. The csv files are under "***data***" folder for your report and analysis. Test dataset for unit test is under "***data/test***" folder. In addition, there is an object file "***(io_op)***" for IO operations in tests. Please, do not delete it.

- **Counting Sort:** Implement the counting sort algorithm
- **Heap Sort:** Implement the heap sort algorithm
- **Sorting Validation & Benchmarking:** Validate your sorting implementations and analyze their efficiency across various cases (input size, input is sorted, etc.) in terms of time and iteration number
- **Scenario:** Execute the exhibition scenario following the given steps
- **Other:** File I/O, data manipulation/transfer, etc.

3.2. Report [30%]

- **Theoretical Background**: Explain briefly the algorithms used in this assignment
- **Technical Details:** Define your approach regarding sorting algorithm implementations and scenario execution
- **Benchmarking Sorting Algorithms:** Analyze and comment on the behavior and efficiency of sorting algorithms concerning a variety of inputs
- **Analyze Your Results:** Comment on the results you obtained from scenario execution, such as the total time required for the pipeline to process each dataset size.
- **Defining & Analyzing Different Rarity Score Calculations:** Develop other rarity score calculation formulas and analyze them (e.g., age window could be fixed sized, or getting the rarest element in each age period – without using age in the calculation–, larger age window, etc.)

3.3. Guidelines and Grading

Your homework will be graded under the following criteria:

- **Code:**
  - Counting Sort 15 %
  - Heap Sort 15 %
  - Sorting Validation & Benchmarking 10 %
  - Scenario 20 %
  - Other 10 %
- **Report:**
  - Theoretical Background 5 %
  - Technical Details 5 %
  - Benchmarking Sorting Algorithms 5 %
  - Analyze Your Results 5 %
  - Defining & Analyzing Different Rarity Score Calculations 10 %