A4-TRHASH Challenge

Stuart Thiel

June 20, 2023

Introduction

This assignment is worth **5%** of your grade. This is an **individual assignment** (so individual, I put *your* student ID on it) and you should not share your assignment with anyone else. The **assignment is due June 25th at 11:59pm, Montreal time**.

All submissions must go through Moodle. I like EAS, but the new VPN rules suck.

Q1) TRHASH Challenge

1.1 General Instructions

This assignment involves understanding and implementing Hashing. You will be asked to discern the strategies used to generate a provided hash class and then construct your own corresponding class.

The hash class utilized in this assignment is called `MyHash`, and you are expected to create your own subclass of this class by overriding the necessary methods. Note that this superclass is part of the `main.stuff` package.

1.2 Programming Task

Your main task is to develop a program named `EfficientHash.java` in the package `ca.concordia.comp352.hashing`. This program should subclass `MyHash` and override the appropriate methods to implement your hash function.

Your program will be compared to the Totally Reliable Hash (TRHASH) based on a dataset provided as an input file. The goal is to create a hash function that outperforms TRHASH. The provided `main.stuff.CompTime` utility will compare your hash function to TRHASH based on the provided dataset.

Please ensure that your implementation correctly hashes the input it receives. You may use Java's built-in libraries (but nothing from java.collections) and structures to help create your function, but the actual hashing algorithm must be your own.

After creating your function, you'll need to compare its performance against the `TotallyReliableHASH` function. To do this, you can use the provided `CompTime` Java program, which will measure the time taken by each function to hash the same data and output the results.

1.3 Additional Requirements

The Driver program will receive three command-line arguments, which are strings:

1. The fully qualified name of your hash function.
2. The filename of the source file containing the data to be hashed.

3. The filename of the source file containing the data to be looked for in the created hash.

The CompTime program will receive three command-line arguments, which are strings:

1. The fully qualified name of a baseline hash function.
2. The fully qualified name of your hash function.
3. The filename of the source file containing the data to be hashed.

It is important to note that the programs accept filenames as input. You should not hardcode filenames or file content in your program.

To compile the code you can run:

```
javac main\stuff\*.java
javac ca.concordia.comp352.hashing.EfficientHash.java
```

To run your code:

```
java main.stuff.Driver ca.concordia.comp352.hashing.EfficientHash dataE6.
    txt test100.txt
```

To run the provided Totally Reliable Hash:

```
java main.stuff.Driver main.stuff.TotallyReliableHASH dataE6.txt test100.
    txt
```

To compare your hash implementation to the TRHASH implementation:

```
java main.stuff.CompTime main.stuff.TotallyReliableHASH ca.concordia.
    comp352.hashing.EfficientHash dataE6.txt
```

See how much better you can do!

In addition to the programming task, you will also need to answer three theory questions related to the assignment.

1.4 Theory Questions

Please provide your answers to the theory questions in comments at the top of your `MyHash.java` file. Your responses should not exceed 200 words each. The questions are:

1. What factors can lead to an inefficient hash table and how did you mitigate them in your design? Give specific examples from your subclass implementation.
2. How does the size of the hash table affect its performance in terms of both time and space complexity? How did you choose the size of your hash table?
3. Discuss the time complexity of your hash table operations: insertion, deletion, and lookup. How do these complexities compare to those of the TRHASH version?

1.5 Evaluation

The zip file for this assignment includes not only the assignment PDF but also a `DebugRunner.class` file and a `config.xml` file. The `DebugRunner` can be executed using Java to provide feedback on your

code. The `config.xml` file serves as a partial grading tool, allowing you to assess your progress in the assignment.

After considering feedback from previous assignments this semester, it appears that the most straightforward method of utilizing `DebugRunner` is by copying the contents of your `bin` folder into the directory containing `DebugRunner.class` and `config.xml`. Once done, you can execute the command line demonstrated in the examples to run `DebugRunner`.

The assignment grade is split into two components: a programming section (60% of your grade) and a theory section (40% of your grade). The programming component is further divided into two parts: 60% is based on how your code performs against the DebugRunner test (which includes your tests and a mystery test), and the remaining 40% is based on the code quality as determined by a marker's review of your class. The theory component of your grade will be derived from your responses to the theory questions.

1.6 Guidance

Understanding Hashing

This assignment centers around a core topic: Hashing. It is crucial that you comprehend this concept thoroughly before proceeding with the tasks at hand.

Hashing is a technique or process of mapping keys, values, or both to a particular space using a hash function. Hash functions are used in various algorithms and structures like Hash Tables, which are utilized for rapid data retrieval.

A good hash function should:

1. Be deterministic: The same input will always produce the same output.
2. Distribute values evenly across the hash space.
3. Be fast to compute.

Test Your Hash Function

Remember, it's not enough to simply create a hash function. The performance of your function will be evaluated against the provided `TotallyReliableHASH`. Hence, it is crucial that your hash function is optimized and efficient.

Write Comprehensive Comments

Make sure you thoroughly comment your code. Your comments will help others understand your thought process and the logic behind your implementation. Also, remember to answer the theory questions within comments at the top of your `MyHash.java` file.

1.7 Submission Instructions

After completing the task, you should have one source file named `MyHash.java`. Please submit this file via the course assignment submission portal.

Your submission should be a single `40254326.zip` file that includes:

1. Your `EfficientHash.java` source file in the appropriate package folder structure

Ensure your submission adheres to the academic integrity policy of the university.

Good luck with your assignment! Remember, understanding the concept of Hashing is crucial, so make sure you take the time to understand it fully. If you have any issues, don't hesitate to seek help during the consultation hours or on the discussion forums.