Hashing: Extra Credit

Due: Sunday July 23 2023 11:59pm PST

Overview

This is a **purely optional, extra credit** assignment that is worth **10 extra credit points** in the homework category. It is totally up to you how much, if any, of this assignment you implement. That being said, we'd highly recommend you attempt this problem set even if you don't need the extra credit.

Instructions

Hashing is a concept that has many applications in the field of Computer Science. In lecture, we discussed how we can use hashing to build a data structure that supports inserting, deleting and finding keys efficiently. This is possible because the hash and compression functions tell you where in the hash table a record lives. If you ever used Python dictionaries or HashMaps or HashSets in Java, you were taking advantage of hashing!

In this assignment, you will implement the core functionality of a Java hashmap. We have provided some basic test cases that you can use to perform sanity checks on your code. We also have hidden test cases that we will not be releasing. See the Testing instructions for more details.

For this assignment, you **cannot import the java.util.Map or java.util.List packages**. **You will receive no credit if you do so**. The only pre-built data structure you can use is an array to represent the buckets of the HashTable.

Setting Up

Download the starter code from the Assignments repo on Github. You can do so by running the following command: **git pull origin main**. This will pull the hashing directory to your local Assignments repo. Make a copy of the hashing folder and move it into your own workspace. For this assignment **you do not need to add any jar files for this assignment from the 'jars' folder from Canvas.**

Problems

In **MyMap.java**, fill in the bodies of the following four functions mentioned below. Do not change the method signatures or parameters. You can implement any helper classes you deem necessary, but be sure they are part of the **map package (sorry, don't get this confused with**

java.util.Map). Do not change the package name. The test cases will fail if you do. We have provided a default constructor in the MyMap.java file. All of our test cases call the empty constructor.

- **void put(K key, V value)**: inserts a record with (key, value) into the map. Implement separate chaining to handle collisions. **You cannot use any pre-built Java Lists to handle separate chaining.** Feel free to use and modify the implementations of the SList/DList or AList that we covered in class earlier in the semester.

- **V get(K key):** return the value associated with the key. Return null if the key does not exist.

- **boolean contains(K key)**: Return a boolean indicating whether the map contains the key.

- **V remove(K key)**: Remove the key and return the associated value from the data structure. If the key does not exist, return null.

- **void resize()**: Double the number of buckets and rehash existing keys when the load factor exceeds .75. Recall the load factor is the # of entries / # of buckets. For this assignment, you are not responsible for reducing the number of buckets.

Autograder

On Gradescope, we have set up an autograder under the assignment HashMaps: Extra Credit. Create a zipped version of the **map** folder and upload it to the assignment. Be sure you upload MyMap.java and any other supporting files you created as part of the **map** package. You are responsible for ensuring the autograder can process your submission.

Testing and Grading

Since this is an extra credit assignment, we will use the results of Gradescope to verify the correctness of your solution and determine your score. There will be no partial credit. If your score shows up as 3/10 on Gradescope, you will have earned 3 extra credit points in the homework category.

Late Policy

You cannot use slip days for this assignment. If the assignment is turned in late, you will receive no credit.

Submission

Upload your final code to your private Github repo. The folder that contains your code for this assignment should be called **hashing.**