CS310 Data Structures Fall 2019

Programming Assignment 3

Comparison of Dictionary Implementation using Hash tables with chaining and Hash tables with linear Probing.

(100 Points)

For this programming assignment you are required to author original code to implement the Dictionary/Map ADT using Hash tables.

The hash tables have to implemented using two collision handling methods:

1. Separate Chaining (using LinkedLists)
2. Linear Probing.

Constraints for the Hash table:

1. Set the size of the hash table to be 50, use a static unordered array.
2. Do not increase the size of the hash table if the hash table gets full. We are comparing Linear Probing and Chaining for your hash function for a limited set of data.
3. You are required to write your own Hashing Function and override the hash code method to write your own method. Do not use the inbuilt java hashcode method.

Generic Data Class (Node) – DataClass.java

The value of you dictionary Node will be the DataClass.java, written as follows:

1. K Key
2. V Value
3. public int hashFunction() – Generates the index based on key.hashCode()

This class will also serve as a Node, hence include any appropriate references necessary to handle Separate chaining with LinkedLists here.

Key.java will be separate class to handle the Key String read from the file. This file will override the hashCode() method.

MapInterface.java – Interface class for Dictionary/Map

Contains the description of the methods for Maps listed in the class slides.

ChainingHashTable.java – Hash table Implementing Separate Chaining

1. Implements the MapInterface.java
2. Define all methods in MapInterface by implementing Separate chaining for collisions.
3. You are allowed to use java.util.LinkedList class for chaining.
4. Write the hash function (compression + call of hashCode() on the key) in this class, that compresses the hashcode to index.
5. Write Iterators for Keys and Values.
6. Main method that uses this class, locally.

ProbingHashTable.java – Hash table Implementing Linear Probing

1. Implements the MapInterface.java
2. Define all methods in MapInterface by implementing Linear Probing for collisions.
3. Write the hash function in this class, that compresses the hash code to index.
4. Write Iterators for Keys and Values.
5. Main method that uses this class, locally.

Main Methods

1. Both main methods must read from a text file to access the key, value pair as done in Programming Assignment 1.
2. The main method must demonstrate the working of all the methods of the MapADT.
3. The flow of the main methods must be as follows:
    a. Read all Key value pairs from java file.
    b. Add the records in the Hash table using appropriate collision handling techniques.
    c. Print the Hash table for all the non-null indexes.
    d. Demonstrate Deletion, search as well.

Text Files

1. Key must be a 5-character alpha-numeric String.
2. Value can be any datatype of your choice.
3. One record per line of the text file, Key and Value separated by a space.

Read Me

Your Read Me must include the answers to the following questions:

1. Explanation of you hashCode () function. How are you converting the 5-character Alphanumeric String to an Integer?
2. What is your Compression method on the hashcode? Why did you choose this compression method?
3. Which strategy for collision handling works better with your Hash Function? Support your answer with snapshots of output from the code, showing which method results in a more even dispersal of the key, value pairs.

Files to be submitted

1. DataClass.java
2. Key.java
3. MapInterface.java
4. ChainingHashTable.java
5. ProbingHashTable.java
6. Records.txt
7. ReadMe.pdf

Put all these files in a ZIP File and submit it on Backboard. No JAR file is necessary.

Programming Prompt

To give you a head start, an incomplete version of DataClass.java and Key.java are included.