UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS212 - Data structures and algorithms

Practical 10 Specifications: Hashing

Release Date: 06-06-2022 at 06:00

Due Date: 10-06-2022 at 23:59

Total Marks: 54

Contents

1 General instructions:

- This assignment should be completed individually, no group effort is allowed.
- Be ready to upload your assignment well before the deadline as no extension will be granted.
- You may not import any of Java's built-in data structures. Doing so will result in a mark of zero. You may only make use of 1-dimensional native arrays where applicable. If you require additional data structures, you will have to implement them yourself.
- If your code does not compile you will be awarded a mark of zero. Only the output of your program will be considered for marks, but your code may be inspected for the presence or absence of certain prescribed features.
- All submissions will be checked for plagiarism.
- Read the entire specification before you start coding.
- You will be afforded five upload opportunities.

2 Plagiarism

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent) and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to http://www.library.up.ac.za/plagiarism/index.htm (from the main page of the University of Pretoria site, follow the Library quick link, and then choose the Plagiarism option under the Services menu). **If you have any form of question regarding this, please ask one of the lecturers, to avoid any misunderstanding.** Also note that the OOP principle of code re-use does not mean that you should copy and adapt code to suit your solution.

3 Outcomes

The aim of this practical is to implement a Cuckoo Hashing Structure with a cellar.

4 Introduction

Complete the task below. Certain classes have been provided for you alongside this specification in the Student files folder. A main has not been provided. Remember to test boundary cases. Submission instructions are given at the end of this document.

5 Task 1: Hashing

A hash function or structure is a method of optimizing the efficiency of searching for elements in a list structure. A hash structure usually consists of key value pairs where the value is stored at the index of the hashed value of the key. A hash function is a specialized function that will change the key such that it is a value within the size bounds of the list. A possible problem that might occur is one of collisions. A collision is when two keys have the same hash value. Multiple techniques have been invented to solve the problem of collision resolution. This practical will attempt to implement such a technique. In this practical you will need to implement a hash table with a cellar. You will be creating a Hash-map that accepts in two generic types. Please note that [] is square brackets.

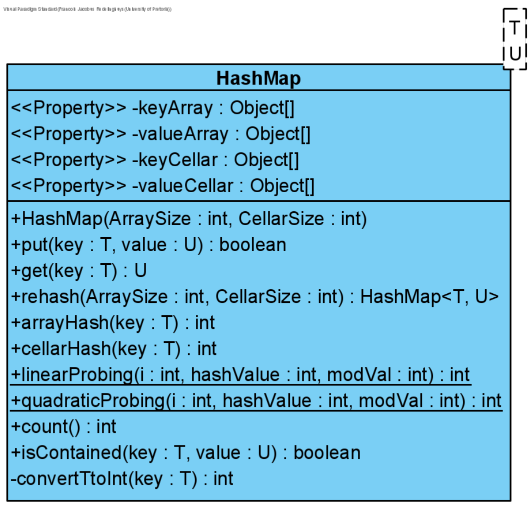
T
U
HashMap
<<Property>> -keyArray : Object[]
<<Property>> -valueArray : Object[]
<<Property>> -keyCellar : Object[]
<<Property>> -valueCellar : Object[]
+HashMap(ArraySize : int, CellarSize : int)
+put(key : T, value : U) : boolean
+get(key : T) : U
+rehash(ArraySize : int, CellarSize : int) : HashMap<T, U>
+arrayHash(key : T) : int
+cellarHash(key : T) : int
+linearProbing(i : int, hashValue : int, modVal : int) : int
+quadraticProbing(i : int, hashValue : int, modVal : int) : int
+count() : int
+isContained(key : T, value : U) : boolean
-convertTtoInt(key : T) : int

Figure 1: Hashmap class diagram

5.1 HashMap class

The class takes in two generic types namely T and U.

- Members:
  - keyArray: Object[]
    - This will be an array containing objects of type T.
    - This array will contain the keys for the hashmap.
  - valueArray: Object[]
    - This will be an array containing objects of type U.
    - This array will contain the values for the keys of the hashmap.
  - keyCellar: Object[]
    - This will be an array containing objects of type T.
    - This array represents the cellar for keys for the table of the hashmap.
  - valueCellar: Object[]
    - This will be an array containing objects of type U.
    - This array represents the cellar for the values of the keys in the cellar of the hashmap.
- Functions:
  - HashMap(ArraySize: int, CellarSize: int)
    - This is the constructor for the HashMap class.
    - The ArraySize parameter should be used to initialize the keyArray and valueArray array.
    - The CellarSize parameter should be used to initialize the keyCellar and valueCellar array.

– put(key: T, value: U): boolean

  * This function should attempt to add the passed in key and value into the hashmap.
  * If the function is able to add the passed in key and value into the hashmap, then the function should return true.
  * If the function is unable to add the passed in key and value into the hashmap, then the function should return false.
  * If the passed in key is already contained in the hashmap the function should return false.
  * The following algorithm describes how a key and value should be added to the hashmap:

```
if the keyArray[arrayHash] is empty
    add key, value
else
    if the keyCellar[cellarHash] is empty
        add key,value
    else
            Loop through the normal array and try to insert the key
               value pair at position i using
               quadraticProbing(i,arrayHash,keyArray size)
        else
            Loop through the cellar array and try to insert the key
               value pair at position i using
               linearProbing(i,arrayHash,keyArray size)

Could not add the key value
```

– get(key: T): U

  * This function should return the value associated with the passed in key.
  * The function should use the same algorithm as put to search for the value in the hashmap.
  * If the key is not associated with a value (i.e. no value can be found for the key) the function should return null.

– rehash(ArraySize: int, CellarSizeL int): HashMap<T,U>

  * This function should create a new HashMap with the passed in parameters.
  * The function should then try to add as many of the key value pairs in the current hashmap as possible.
  * Keys should be added to the new HashMap linearly starting with the first key in the keyArray to the last value of the keyCellar.
  * It can be assumed that ArraySize and CellarSize will always be a positive integer.

- arrayHash(key: T): int
  - This is the hash function for the keyArray array.
  - the hash function is defined as follows:

$$h(key) = |key\%length(keyArray)|$$

  - *Hint: use the convertToInt function to convert the key to an int.*
- cellarHash(key: T): int
  - This is the hash function for the keyCellar array.
  - the hash function is defined as follows:

$$h(key) = |length(keyArray) - key|\%length(keyCellar)$$

  - *Hint: use the convertToInt function to convert the key to an int.*
- linearProbing(i: int, hashValue: int, modVal: int): int
  - This is a static function.
  - This function will calculate a linear probing index based on the passed in parameters and on the following definition:

$$LP(i, hashValue, modVal) = (hashValue + i)\%modVal$$

- quadraticProbing(i: int, hashValue: int, modVal: int): int
  - This is a static function.
  - This function will calculate a quadratic probing index based on the passed in parameters and on the following definition:

$$QP(i, hashValue, modVal) = |hashValue{+}round((-1)^{i-1}) \times (floor\left(\frac{i+1}{2}\right))^2|\%modVal$$

- count(): int
  - This function should count the amount of key value pairs in the hashmap.
  - If there is no key value pairs in the hashmap the function should return 0.
- isContained(key: T, value: U): boolean
  - This function should determine if the passed in key is associated with the passed in value.
  - If the key and value is associated with each other the function should return true else the function should return false.
- convertTtoInt(key: T): int
  - It is not advised to change this function.
  - This is a helper function that will convert the key to an integer representation.
- All the getters should return the appropriate property.

6 Submission

You need to submit your source files on the Fitch Fork website (https://ff.cs.up.ac.za/).All methods need to be implemented (or at least stubbed) before submission. Only the following java files should be in a zip archive named uXXXXXXXX.zip where XXXXXXXX is your student number:

- HashMap.java

There is no need to include any other files in your submission. Your code should be able to be compiled with the following command:

```
javac *.java
```

You have 5 submissions and your best mark will be your final mark. Upload your archive to the Practical 10 slot on the Fitch Fork website. Submit your work before the deadline. **No late submissions will be accepted!**