CS2040C Semester 1 2018/2019

Data Structures and Algorithms

Tutorial 06 - Midterm Test Debrief, Table ADT 1, Hash Table

For Week 8 (Week Starting on 8 October 2018)

Document was last modified on: October 4, 2018

1 Introduction and Objective

In the first 20m of the tutorial, the tutor will discuss some selected questions in last week's CS2040C Midterm Test.

Then, we will continue our discussion about Non-Linear Data Structures. Next up is Hash Table, **one possible efficient** implementation of Table ADT (unordered). We will discuss the Hash Function and the next tutorial will cover Collision Resolution Techniques.

2 Midterm Test Discussion

This topic is open to the class. Please suggest and recommend some questions that you would like to ask the tutor regarding the Midterm test.

3 Tutorial 06 Questions

Hash Table Basics

Q1). A good hash function is essential for good Hash Table performance. A good hash function is easy/efficient to compute and will evenly distribute the possible keys. Comment on the flaw (if any) of the following hash functions. Assume that the load factor α = number of keys N / Hash Table size $M = 0.3$ (i.e. low enough) for all the following cases:

1. $M = 100$. The keys are positive even integers. The hash function is h(key) = key % 100.
2. $M = 1\,009$. The keys are valid email addresses. The hash function is h(key) = (sum of ASCII values of the last 10 characters) % 1 009. See `http://www.asciitable.com` for ASCII values.

3. $M = 101$. The keys are integers in the range of [0, 1 000]. The hash function is h(key) = floor(key * random) % 101, where $0.0 \leq \text{random} \leq 1.0$.

Q2). Hashing or No Hashing: Hash Table is a Table ADT that allows for `search(v)`, `insert(new-v)`, and `delete(old-v)` operations in O(1) average-case time, **if properly designed**. However, it is not without its limitations. For each of the cases described below, state if Hash Table can be used. If not possible to use Hash Table, explain why Hash Table is not suitable for that particular case. If it is possible to use Hash Table, describe its design, including:

1. The <Key, Value> pair
2. Hashing function/algorithm

The cases are:

1. A mini-population census is to be conducted on every person in your (not so large) neighbourhood. We are only interested in storing every person's name and age. The operations to perform are: retrieve age by name and retrieve name(s) by age.
2. A much larger population census is also conducted across the country, similarly containing only every person's name and age. The operation to perform is: retrieve names of people eligible for voting. Only people above legal age 17 years old (or older) are eligible for voting. However, we still need to store the rest of the data.
3. A different population census similarly contains only the name and age of every person. The operation to perform is: Retrieve people with a given last name.
4. A grades management program stores a student's index number and his/her final marks in one GCE 'O' Level subject. There are 100,000 students, each scoring final marks in [0.0, 100.0]. The operation to perform is: Retrieve a list of students who passed in ranking order (highest final marks to passing marks). A student passes if the final marks are more than 65.5. Whether a student passes or not, we still need to store all students' performance.

Table ADT

Q3). What is/are the main difference(s) between List ADT basic operations (see `https://visualgo.net/en/list?slide=2-1`) versus Table ADT basic operations (see `https://visualgo.net/en/hashtable?slide=2-1`)?