CS2040C Semester 1 2021/2022

Data Structures and Algorithms

Tutorial+Lab 06

Table ADT 1: Hash Table

For Week 08

Document is last modified on: July 30, 2021

1 Introduction and Objective

In this tutorial, we will continue our discussion about Hash Table, **one possible efficient** implementation of Table ADT (unordered). We will heavily use `https://visualgo.net/en/hashtable` in this tutorial.

2 Tutorial 06 Questions

Hash Table Basics

Q1). 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 is Hash Table 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
3. Collision resolution (OA: LP/QP/DH or SC; give some details)

(Choose 2 out of 4 to be discussed live): The cases are:

1. A mini-population census is to be conducted on every person in your (not so large) neighbourhood. No two person have the same name but there can be two or more person with the same

age. You can assume that age is an integer and within reasonable human age range [0..150] years old. We are only interested in storing every person's name and age. The operations to perform are: retrieve age by name and retrieve list of names (in any order) by age.

2. A much larger population census is also conducted across the country, similarly containing only every person's name and age. Again, no two person have the same name but there can be two or more person with the same age. You can assume that age is an integer and within reasonable human age range [0..150] years old. The operation to perform is: retrieve list of names (in any order) 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 (in full name, distinct) and the age of every person. The operation to perform is: Retrieve person's full name and his/her age given a last (sur)-name. Note that although the full names are distinct, their last (sur-)names may not.

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.

Basic Hash Table Stuffs

Q2). Quick check: Let's review all 4 modes of Hash Table (use the Exploration mode of `https://visualgo.net/en/hashtable`). During the tutorial session, the tutor will randomize the Hash Table size M, the selected mode (LP, QP, DH, or SC), the initial keys inside, and then ask student to `Insert(random-integer)`, `Remove(existing-integer)`, or `Search(integer)` operations. This part can be skipped if most students are already comfortable with the basics.

Hash Table Discussions

Q3). (Choose 2 out of 4 to be discussed live): The following topics require deeper understanding of Hash Table concept. Please review `https://visualgo.net/en/hashtable?slide=1`, use the Exploration Mode, or Google around to help you find the initial answers and we will discuss the details in class. For some questions, there can be more than one valid answer.

1. 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`)?

2. At `https://visualgo.net/en/hashtable?slide=4-4`, Steven mentions about Perfect Hash Function. Now let's try a mini exercise. Given the following strings, which are the names of Steven's current family members: {"Steven Halim", "Grace Suryani Halim", "Jane Angelina Halim", "Joshua Ben Halim", "Jemimah Charissa Halim"}, design any valid **minimal perfect**

hash function to map these 5 names into index [0..4] without any collision. Steven and Grace are not planning to increase their family size so you can assume that $N = 5$ will not change.

3. Thus far, which collision resolution technique is better (in your opinion or Google around): One of the Open Addressing technique (LP, QP, DH) or the Closed Addressing (Separate Chaining/SC) technique?

4. Which non-linear data structure should you use if you have to support the following three operations: 1). many insertions, 2) many deletions, and 3) many requests for the data in sorted order?

Hands-on 6

TA will run the second half of this session with a few to do list:

- Very quick review of C++ STL std::unordered_set and std::unordered_map,
- Do a(nother) sample speed run of VisuAlgo online quiz that are applicable so far, e.g., `https://visualgo.net/training?diff=Hard&n=5&tl=5&module=hashtable`.
- Finally, live solve another chosen Kattis problem involving Table ADT (that does *not* require ordering).

Problem Set 3

We will end the tutorial with **last-minute algorithmic** discussion of PS3.
We are now allowed to discuss 100+100 solutions in high level.