IT5003 Semester 2 2024/2025

Data Structures and Algorithms

Tutorial 05
Hash Table
For Week 07 (Sat)/08 (Mon)

Document is last modified on: March 6, 2025

1 Introduction and Objective

In this tutorial, we will discuss Hash Table, **one possible efficient** (unordered) implementation of Table ADT. We will use `https://visualgo.net/en/hashtable` in this tutorial. We will contrast and compare two collision resolution techniques that each has its own strengths and weaknesses.

2 Tutorial 05 Questions

Hash Function Basics

Q1). Which of the following is the best (string) hash function?

1. `index = (rand() * (key[0]-'A')) % M`
2. `index = (key[0]-'A') % M`
3. `index = hash_function(key) % M`

where

- `rand()` is a function that returns a pseudo-random Integer $\in$ [0..`RAND_MAX`] (This value is library-dependent, but is guaranteed to be at least 32767 on any standard library implementation (In Python, we can use `from random import randint` library)).
- `key` is a string and we can subtract the ASCII values of two characters, e.g., 'C'-'A' = $67-65 = 2$
- `M` is the hash table size, usually a prime number
- `hash_function(v)` is as shown in `https://visualgo.net/en/hashtable?slide=4-7`

Q2). 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 (necessary condition to have good performing Hash Table implementation). Comment on the flaw (if any) of the following (integer) hash functions. Assume that for this question, the load factor α = number of keys N / Hash Table size M, is a small enough constant (e.g., $\alpha < 2$ for a typical Separate Chaining implementation or $\alpha < 0.5$ for a typical Linear Probing implementation) for all cases below:

1. $M = 100$. The keys are $N = 50$ positive even integers in the range of [0, 10 000].
   The hash function is h(key) = key % 100.

2. $M = 101$. The keys are $N = 50$ positive integers in the range of [0, 10 000].
   The hash function is h(key) = floor(sqrt(key)) % 101.

Hash Table Basics

Q3). 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 (SC or OA — LP/QP/DH; give some details)

The cases are:

1. A population census is to be conducted on every person in your (very large, e.g., population of 1 Billion) country. You can assume that no two person have the same name in this country. However, 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. Important consideration: Each year, everyone's age increases by one year, a bunch of new babies (age 0) are born and added into the database, some people unfortunately pass away and removed from the database. All these yearly changes have to be considered.

2. 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 exact precision needed is not known). 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 as the passing final marks can be adjusted as per necessary.

Q4). 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 (SC or OA:LP/QP/DH), 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

Q5). 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 answers.

1. At `https://visualgo.net/en/hashtable?slide=4-4`, Prof Halim writes about Perfect Hash Function (that was not discussed in lecture 6). Now let's try a mini exercise. Given the following strings, which are the names of Prof Halim'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. Prof Halim and his wife Grace are not planning to increase their family size so you can assume that $N = 5$ will not change.
2. Which 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 5

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

- IT5003: Quick review of Python set, dict, defaultdict, and Counter.
- Do a speed run of VisuAlgo online quiz that are applicable so far, e.g., `https://visualgo.net/training?diff=Medium&n=5&tl=5&module=hashtable`,
- Live solve another chosen Kattis problem involving Table ADT (that does *not* require ordering)

Problem Set 4

We will end the tutorial with **algorithmic** discussion of PS4 as this is the only lab session before PS4 due next Sat, 15 Mar 2025, 07.59am.