CS2040S+IT5003 Semester 1 2024/2025

Data Structures and Algorithms

Tutorial+Lab 06
Table ADT 1: Hash Table
For CS2040S Only: UFDS

For Week 08

Document is last modified on: July 21, 2024

1 Introduction and Objective

In this tutorial, we will discuss Hash Table, **one possible efficient** implementation of Table ADT (unordered). We will heavily 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 06 Questions

Hash Function Basics

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

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

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).
- `key` is a string and we can subtract the ASCII values of two characters, e.g., 'C'-'A' = $67-65 = 2$

- `N` 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 $M \leq 10$ (i.e., small enough for our Separate Chaining or 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. $M = 100$. 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 (OA — LP/QP/DH or SC; 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 (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

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. 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. 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?

CS2040S Only (Skipped for IT5003): UFDS Review

Q6). Using `https://visualgo.net/en/ufds`, quickly review the `findSet(i)`, `isSameSet(i, j)`, `unionSet(i, j)` operations of the Union-Find Disjoint Sets (UFDS) data structure.

Q7). The basic UFDS data structure can be augmented to support extra operations. The first (and easiest) augmentation is to support `numDisjointSets()` query in $O(1)$ (instead of in $O(N)$). When we create a new instance of UFDS, we create N initially disjoint sets. Show how we can carefully track these information throughout various UFDS other operations!

Q8). The second (harder, but more versatile) augmentation is to support `sizeOfSet(i)` query in $O(1)$ (instead of in $O(N)$). This query reports the size of set that currently contains item i. Think of how to do this operation quickly and especially if two previously disjoint sets were merged into one!

Hands-on 5

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

- IT5003: Very quick review of Python set, dict, defaultdict, and Counter.
- CS2040S: Very quick review of Java HashSet and HashMap,

- 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`,
- IT5003 Only (as CS2040S needs more tutorial time for UFDS review): Live solve another chosen Kattis problem involving Table ADT (that does *not* require ordering) but has interesting time complexity analysis (to be fully understood in CS3230 later).

Problem Set 4

We will end the tutorial with **last-minute algorithmic** discussion of PS4.