CS2040C Semester 1 2018/2019

Data Structures and Algorithms

Tutorial 07 - Table ADT 2, Collision Resolution, Intro to Binary Search Trees

For Week 9 (Week Starting 15 October 2018)

Document was last modified on: October 11, 2018

1 Introduction and Objective

The purpose of this tutorial is to wrap up the discussion of Hash Table and introduce Binary Search Tree (BST). We will revisit the scenarios in Tutorial 06 and discuss what collision resolution techniques will best suit each case.

2 Tutorial 07 Questions

Basic Hash Table Stuff

Q1). 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 provide a quick summary on the different collision resolution techniques (LP, QP, DH, or SC).

Q2). (From Tut 06, now with Collision Resolution) 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 (including second hash function/algorithm for double hashing) (Main Focus)

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.

Basic Operations of Binary Search Tree

Q3). We will end this tutorial with a quick review (or introduction) of basic BST operations. Go to `https://visualgo.net/en/bst`, click Create → Random. Then, experiment with the operations below. While doing so, write down what the operation does and its time complexity. You may assume the height of the BST is H and it contain N nodes. Due to time constraint, the tutor might not cover all operations.

1. **Search(v)**: What does *Find Minimum* and *Find Maximum* do on VisuAlgo?

2. **Insert(v)**: Can you insert duplicate values into a Binary Search Tree (BST)? (Optional: What if you want to have duplicate values?)

3. **Remove(v)**: What happens if you remove a node which has *2 children*?

4. **Successor(v)**: What is the *successor* of a particular node?

5. **Pre-decessor(v)**: What is the *pre-decessor* of a particular node?

6. **Inorder Traversal**: What do you observe about the Inorder Travsersal of a Binary Search Tree (BST)?