IT5003 Mar-May 2024

Data Structures and Algorithms

Tutorial+Lab 05
Table ADT: Binary Heap, Hash Table, and BST

Document is last modified on: January 6, 2024

1 Introduction and Objective

We need to catch up with long weekend last week. So, in the slightly longer first half of the session (and thus this session will likely take more time than usual), we will review:

- `https://visualgo.net/en/heap` data structure for Priority Queue ADT,
- `https://visualgo.net/en/hashtable` data structure for Table ADT, especially its Closed Addressing/Separate Chaining collision resolution technique versus what is likely used in Python set: Open Addressing/Linear Probing collision resolution technique, and
- `https://visualgo.net/en/bst` data structure, the unbalanced version, as another alternative data structure to implement Table ADT. However, since the full form (balanced BST) is only discussed briefly during recitation, this part is for theoretical interest only in this IT5003 module.

2 Questions

More Binary Heap

Q1). Give an algorithm to find all vertices that have value $> x$ in a Binary Max Heap of size n.
Your algorithm must run in $O(k)$ time where k is the number of vertices in the output.
Key lesson: This is a new algorithm analysis type for most of you as the time complexity of the algorithm does not depends on the input size n but rather the output size k :O...

Note that this question has also been integrated in VisuAlgo Online Quiz, so it may appear in future Online Quizzes :).

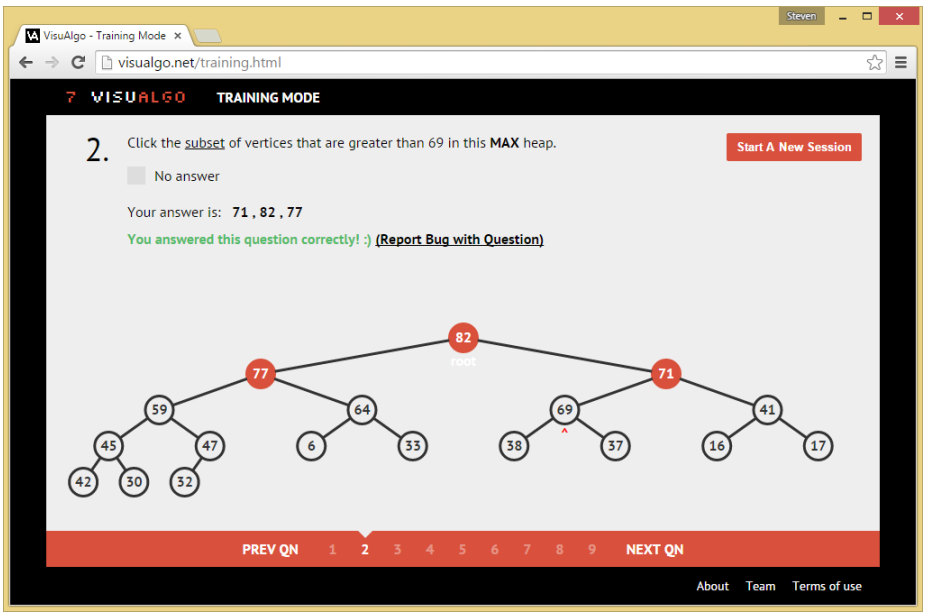

Figure 1: Also automated :)

Hash Function Basics

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 — only LP in IT5003 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.

Binary Search Tree

Q4). (Optional, only when many are still not comfortable with basic bBST operations): We will start this tutorial with a quick review of basic BST operations that is not necessarily balanced. The tutor will first open `https://visualgo.net/en/bst`, click Create → Random. Then, the tutor will ask students to Search for some integers, find Successor of existing integers, perform Inorder Traversal, Insert a few random integers, and also Remove existing integers.

Further 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 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. Which non-linear data structure should you use if you have to support the following three operations that can come in any order: 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:

- PS4 Quick Debrief,
- Do a sample 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,bst`. PS: Skip parts that are skipped for this sem's IT5003.
- Finally, live solve another chosen (short) Kattis problem involving Table ADT.

Problem Set 5

We will end the tutorial with high level discussion of PS5 A+B.