CS2040S+IT5003 Semester 1 2024/2025

Data Structures and Algorithms

Tutorial+Lab 04

Priority Queue

For Week 06

Document is last modified on: July 21, 2024

1 Introduction and Objective

This tutorial marks the end of the first $\frac{1}{3}$ of CS2040/C/S and IT5003: Basic Java/C++/Python, basic analysis of algorithms (Big O worst-case time complexity only), various sorting algorithms, and various linear Data Structures (DSes): Array (Python List)/Linked List/Stack/Queue/Doubly Linked List/Deque.

This tutorial marks the start of the next $\frac{1}{3}$ of CS2040/C/S and IT5003: Various non-linear DSes. Today, we will discuss the Priority Queue (PQ) ADT with its Binary Heap implementation (use `https://visualgo.net/en/heap` to help you answer some questions in this tutorial).

2 Tutorial 04 Questions

Basic Binary Heap

Q1). Quick check: Let's review all basic operations of Binary Heap that are currently available in VisuAlgo (use the Exploration mode of `http://visualgo.net/en/heap`). During the tutorial session, the tutor will randomize the Binary Heap structure, ask student to compare Binary Tree versus (1-based) Compact Array mode, `Insert(random-Integer)` (you can try inserting duplicates, it is now allowed), perform `ExtractMax()` operations (once, K-times (i.e., partial sort), or N-times (i.e., `HeapSort()`)), the $O(N \log N)$ or the $O(N)$ `Create(from-a-random-array)`, `UpdateKey(i, newv)` and `Delete(i)`.

Q2). What is the minimum and maximum number of comparisons between Binary Heap elements required to construct a Binary (Max) Heap of arbitrary n elements using the $O(n)$ `Create(array)`?

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

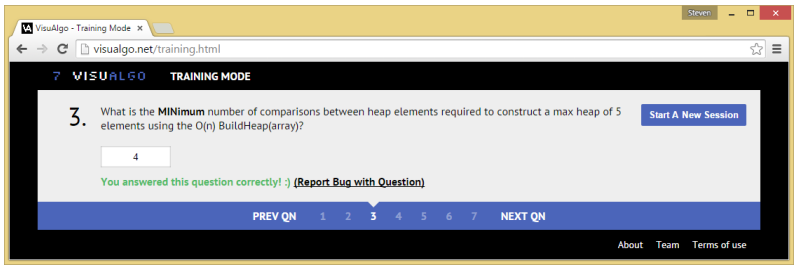

Figure 1: Now automated :)

More Binary Heap

Q3). 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 :).

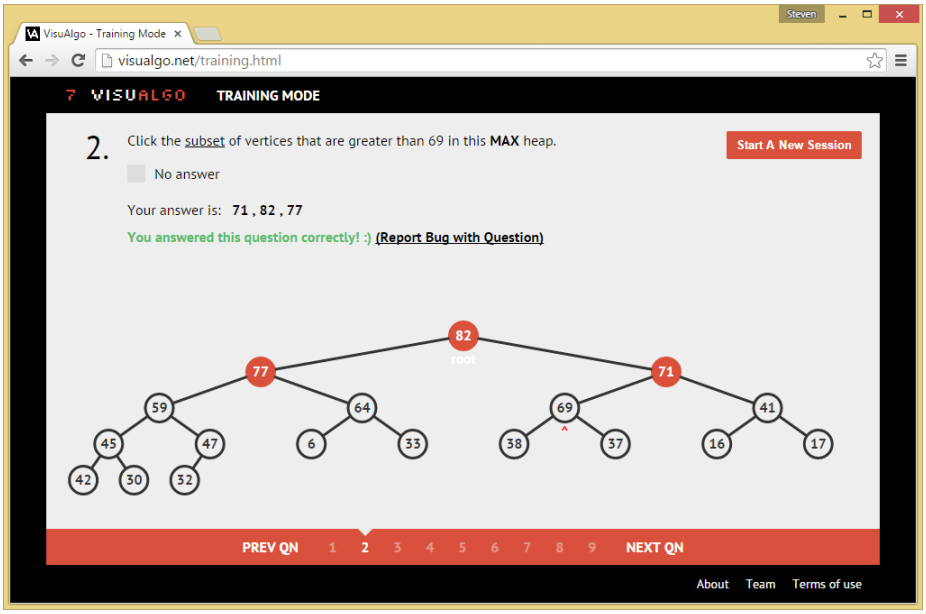

Figure 2: Also automated :)

Q4). Quick check: Show an easy way to convert a Binary Max Heap of a set Integers (as shown in VisuAlgo `https://visualgo.net/en/heap?slide=9-4`) into a Binary Min Heap (of the 'same' set of Integers) without changing the underlying data structure at all. Hint: modify the data.

Q5). The second largest element in a max heap with more than two elements (to simplify this question,

you can assume that all elements are unique) is always one of the children of the root. Is this true? If yes, show a simple proof. Otherwise, show a counter example.

Note that this kind of (simple) proof may appear in future CS2040/C/S written tests (unlikely asked to show proof in IT5003), so please refresh your CS1231/S (if you have taken that course) or just concentrate on how the tutor will answer this kind of question.

More ADT PQ Operations

Q6). There are two Binary Heap data structure features that are not available (yet) in C++ STL `std::priority_queue`, Python `heapq`, and Java `PriorityQueue`: `Increase/Decrease/UpdateKey(old_v, new_v)` and `DeleteKey(v)` where `v` is not necessarily the max element. These two operations are not yet included in VisuAlgo (the hidden slide `https://visualgo.net/en/heap?slide=3-1`) although the version where index i is known, is now included in VisuAlgo (see `https://visualgo.net/en/heap?slide=9-5` and `https://visualgo.net/en/heap?slide=9-6`).

Q6a). Given `https://www.comp.nus.edu.sg/~stevenha/cs2040c/demos/BinaryHeapDemo.cpp` (that is a Binary Max Heap), what should we modify/add so that we can implement `DecreaseKey(old_v, new_lower_v)` – notice that we never IncreaseKey?

Q6b). Given `https://www.comp.nus.edu.sg/~stevenha/cs2040c/demos/BinaryHeapDemo.cpp` (that is a Binary Max Heap), what should we modify/add so that we can implement `DeleteKey(v)` where `v` is not necessarily the max element?

Hands-on 4

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

- IT5003: Very quick review of Python heapq
- CS2040S: Very quick review of Java PriorityQueue Class
- 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=heap`,
- Finally, live solve another chosen Kattis problem involving priority queue ADT.

Problem Set 3

We will end the tutorial with **another round of algorithmic** discussion of PS3 that will due soon.