IT5003 Oct-Dec 2023

Data Structures and Algorithms

Tutorial+Lab 04
Priority Queue

Document is last modified on: July 18, 2023

1 Introduction and Objective

This session marks the end of the first $\frac{3}{8}$ of IT5003: Review of basic Python, basic analysis of algorithms (worst case time complexity only), a few sorting algorithms (Bubble/Insertion/Selection/Merge Sort/(Randomized) Quick Sort and a bit of Counting Sort; but skipping Radix Sort), and a few linear Data Structures (DSes): Python List/Singly Linked List/Stack/Queue/Doubly Linked List/Deque).

This session marks the start of the next $\frac{1}{4}$ of IT5003: A few 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 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)`.

This part is open ended, up to the tutor. Focus on how Binary Heap can be used as a **Priority** Queue. Do not be long winded here. Perhaps focus more on the four new ones: Changing Mode, Partial Sort, Update Key (if index known), Delete Key (if index known).

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 :).

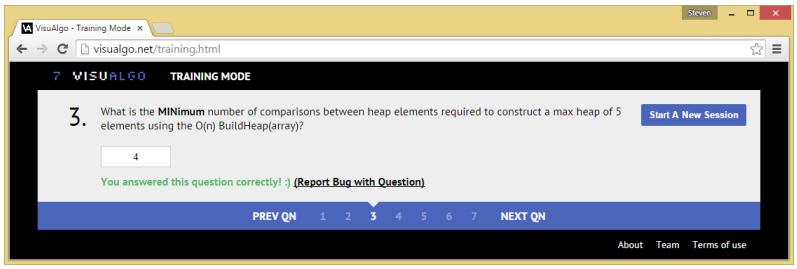

Figure 1: Now automated :)

Ans: Let's use an example for $n = 8$. The min and max answers are 7 and 11 respectively.

The case for minimum happens when the array to be converted into a Binary (Max) Heap already satisfies the Max Heap property (e.g., try input array {8,7,6,5,4,3,2,1} in VisuAlgo). The structure of an 8 vertices heap is shown below.

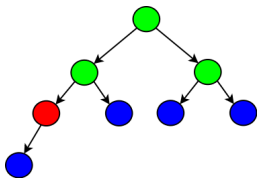

Figure 2: Minimum case, n=8

In this case except for the last internal vertex (the red vertex), which only does 1 comparison, the rest of the internal vertices (green vertices) do exactly 2 comparisons (with its left and right child) so the # of comparisons = 1+3*2 = 7.

The case for maximum happens where we have to call `ShiftDown` at each internal node all the way to the deepest leaf (note: contrary to intuition, try input array {1,2,3,4,5,6,7,8} in VisuAlgo does **not** really produce the maximum number of comparison as 1 will be shifted down to 8 then to 5 only, try input array {1,2,3,5,4,6,7,8} in VisuAlgo where 1 will be shifted down to 8, then 5, then 2. The structure of an 8 vertices heap is shown below and can be generated as follows: {5,6,2,7,4,3,1,8} (do you see the pattern on how to generate this test case?).
The red node requires 1 comparison
The purple node requires 2 comparisons
The green node requires 2+1 (2 comparisons at level 1, 1 comparison one level down at level 2)
Finally, the yellow node requires 2+2+1 (trace the longest path) comparisons
Total = 1+2+(2+1)+(2+2+1) = 11.

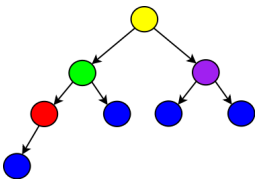

Figure 3: Maximum case, n=8

We have a bigger test case of this pattern (with integer values) in VisuAlgo (/heap). Click 'Create(A) - O(N)', 'Worst Case: Diagonal Entry' test case for a more general form of this test case.

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 :).

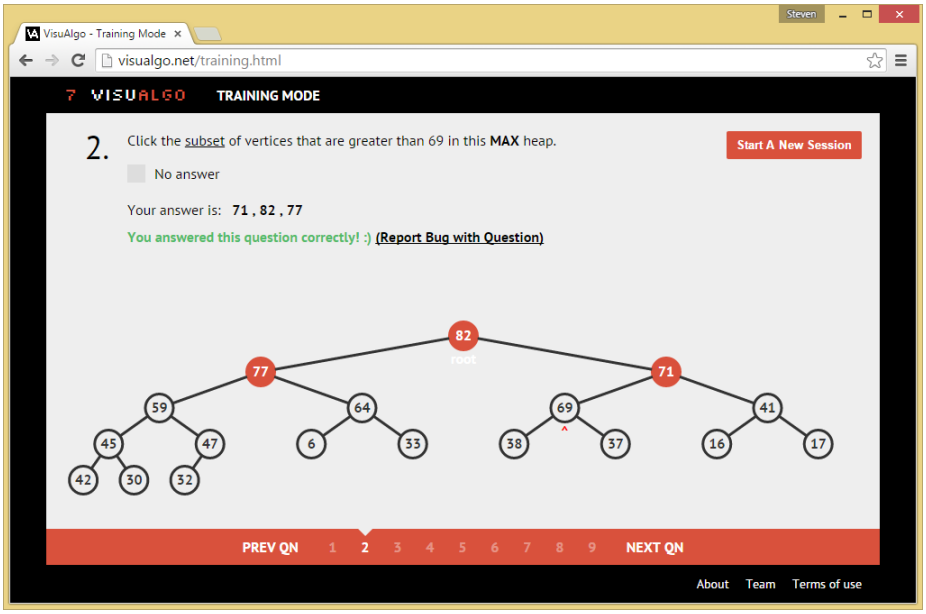

Figure 4: Also automated :)

Ans: The key insight is that the answers are at the top of the max heap... So we can perform a pre-order traversal (tutor will elaborate the concept of pre-order traversal as this may be new for many students; such graph traversal algorithms will only be discussed later) of the max heap starting from the root. At each vertex, check if vertex key is $>$ x. If yes, output vertex and continue traversal. Otherwise terminate traversal on subtree rooted at current vertex (as none of the vertex in

Algorithm 1 findVerticesBiggerThanX(vertex, x)

```
if (vertex.key > x) then
    output(vertex.key)
    findVerticesBiggerThanX(vertex.left, x)
    findVerticesBiggerThanX(vertex.right, x)
end if
```

this subtree will contribute to the answers), return to parent and continue traversal.

Analysis of time bound required:

The traversal terminates when it encounters that a vertex's key $\leq x$. In the worst case, it encounters $k * 2$ number of such vertices. That is, each of the left and right child of a valid vertex (vertex with key $> x$) are invalid. It will not process any invalid vertex other than those $k * 2$ vertices. It will process all k valid vertices, since there cannot be any valid vertices in the subtrees not traversed (due to the heap property). Thus the traversal encounters $O(k + 2 * k) = O(k)$ number of vertices in order to output the k valid vertices.

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

Ans: Simple, just insert the **negation** of those integers into another empty Binary Max Heap. For example: 5 (max), 4, 3, 2, 1 will now looks like this if negated: -1 (max), -2, -3, -4, -5.

For Python, the default of heapq is a min heap. We can convert it into a max heap for numbers by inserting their negatives.

Hands-on 4

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

- PS3 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=heap`.
- Finally, live solve another chosen Kattis problem involving a Priority Queue.

Do PS3 quick debrief according to the context of your lab group.

Do VisuAlgo Online Quiz sample run in medium setting.

`https://nus.kattis.com/problems/knigsoftheforest`.

/knigsoftheforest is a PQ simulation problem. First, we need to sort the moose. We insert the first k moose with 2011 as year of entry, then we simulate the strengths of new moose for the next n-1 years and see if Karl is ever become the alpha moose?

Problem Set 4

We will end the tutorial with high level discussion of PS4 A+B.

For PS4 A (/arraysmoothening), is it always beneficial to greedily remove an integer with the highest frequency (on how to count this, recall previous PS on counting frequencies using sorting - or using another data structure). If students are strong in mathematics, perhaps the tutor can show a quick exchange argument proof on why this is always the case (if there are other integers that can be removed at this step optimally, changing that integer with the integer currently with the highest frequency is always equally optimal). Then, a max PQ is used to maintain this ordering dynamically (as the frequency changes every time we remove the integer with the highest frequency). Again, the solution is short after students realize the easy way to convert Python min PQ to max PQ for integer keys.

For PS4 B (/janeeyre), the initial troublesome part is the input parsing (for other programming languages). But for Python users, we can just specify how we want to split the input (use double quote as the separator). Afterwards, this becomes a (min) Priority Queue simulation problem. Just read the problem description carefully to simulate the required book reading process until you read 'Jane Eyre' book.