# Lab 15: Heaps

## 1.1 Information

**Topics:** Heaps

**Turn in:** This is another on-paper "lab". Turn in a text file, PDF file, scanned or photographed images.

---

## 1.2 About: Heaps

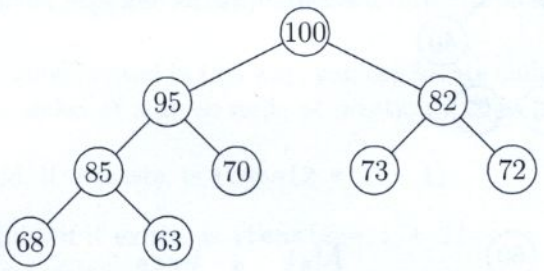



Figure 1.1: An example Heap

> "A heap container is a binary tree data structure that is not used for searching, [...] In a heap data structure the node keys are always larger than their child nodes, but their children nodes are not in any particular order. This differs from the binary tree, in which the left child is always the smaller node, while the right node is always the larger value when compared to their parents."
>
> From Data Structures and Algorithms for Game Developers, by Allen Sherrod, page 328

> "Three major characteristics make a heap data structure what it is: (a) A heap is a binary tree. (b) A heap is a complete data structure, meaning the rows of nodes are completely filled in from left to right without any gaps, [...] (c) Every node in a heap is larger than or equal to its child nodes."
>
> From Data Structures and Algorithms for Game Developers, by Allen Sherrod, page 329

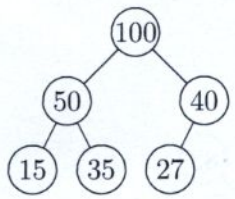
If a heap has the larger values as parents, it is known as a **maxheap**. Otherwise, if the parents have smaller values than their children, the heap is a **minheap**.

---

**Question 1** ____ / 2

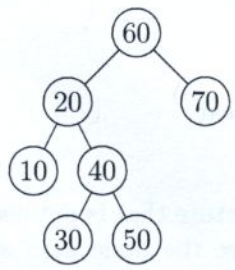
Identify which of the following are heaps. If a graph is not a heap, please specify what makes it invalid.

a. [1]




Heap (maxheap)

b. [2]




Not a heap
- Some children have larger values than the parents, but the root node is not the smallest value.
- There are gaps on the left.

[1] From Data Structures and Algorithms for Game Developers, by Allen Sherrod, page 328

[2] From Data Abstraction & Problem Solving with C++: Walls and Mirrors 7th ed, by Carrano and Henry, page 489

### 1.2.1 Implementation

Even though heaps are visualized as a binary tree, they are usually implemented with an array structure, with the root node (the node with the greatest value) being at index 0.

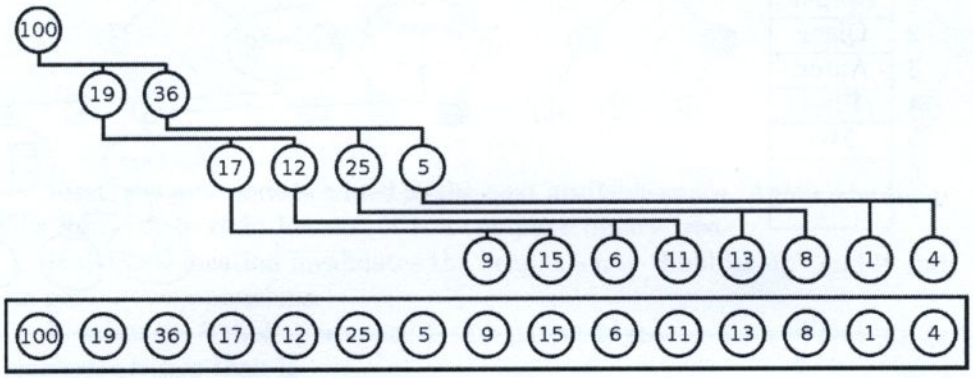



Figure 1.2: "Example of a complete binary max-heap with node keys being integers from 1 to 100 and how it would be stored in an array." by Maxiantor; downloaded from https://en.wikipedia.org/wiki/Heap_data_structure#Implementation

With a heap implemented in this way, you can locate children and parents by modifying the index of a given node at position $i$ (that is, `item[i]`)

- Its left child, if it exists, is `items[2 * i + 1]`
- Its right child, if it exists, is `items[2 * i + 2]`
- Its parent child, if it exists, is `items[(i - 1) / 2]`

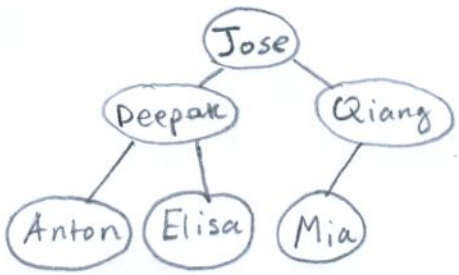
Remember that only the root in `items[0]` does not have a parent. [3]

[3]From Data Abstraction & Problem Solving with C++: Walls and Mirrors 7th ed, by Carrano and Henry, page 489

**Question 2** ____ / 2

Draw the heap that corresponds to the given array.

| | items |
|---|---|
| 0 | Jose |
| 1 | Deepak |
| 2 | Qiang |
| 3 | Anton |
| 4 | Elisa |
| 5 | Mia |
| 6 | |
| 7 | |

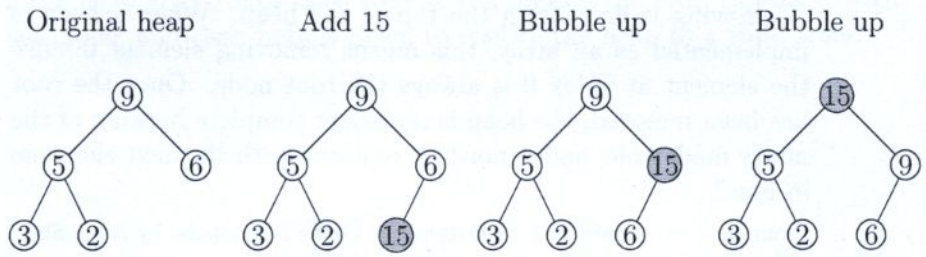



---

**Inserting into a heap**

> "Although a heap is weakly ordered, the purpose of the data structure is to allow for fast removal from the top of the heap. When an item is inserted into the data structure, it is initially placed on the bottom of the list. The element cannot stay at this position because its value might violate the heap condition that states that every child must be smaller than its parent. Thus, when an element is inserted into the container, it is moved up through the list until it finds an index where the element is smaler than its parent but larger than its children."
>
> From Data Structures and Algorithms for Game Developers, by Allen Sherrod, page 329

Example: Adding the number 15 to the following heap...




[4]

First, the new node is added to the next available space. Again, the Heap fills from left to right because it is a complete binary tree.

If the new location invalidates the heap, then it "bubbles up" and it and its parent swap positions.

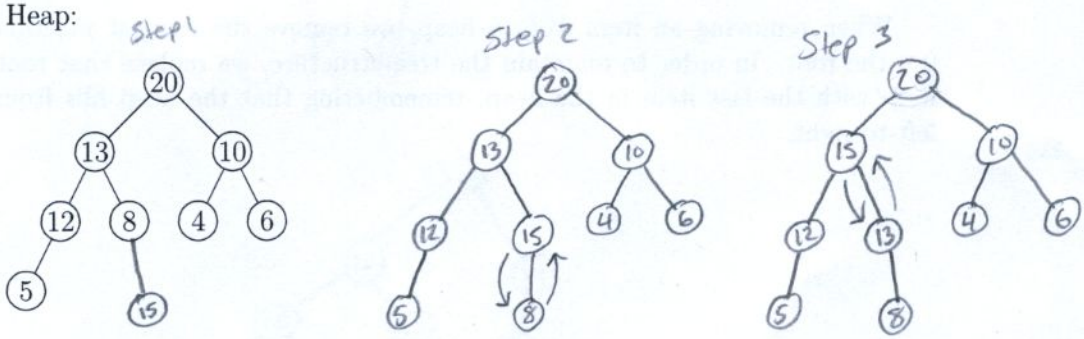
It continues bubbling up until (1) its parent is greater than it, and (2) its children are less than it.

---

**Question 3** ____ / 2

With the heap given, draw each step as you insert the new node and bubble it up to a valid location.

Heap:

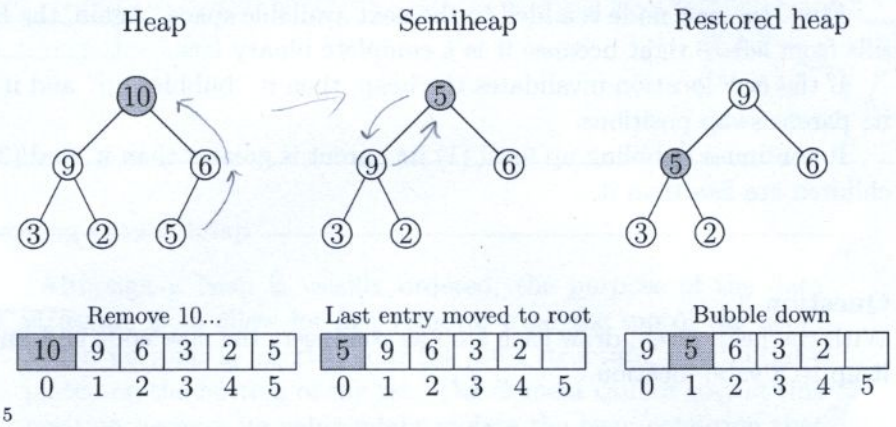



Inserting: 15

[4] From Data Abstraction & Problem Solving with C++: Walls and Mirrors 7th ed, by Carrano and Henry, page 523

### Removing from a heap

> "Removing is done from the top of the heap. When a heap is implemented as an array, this means removing element 0 since the element at index 0 is always teh root node. Once the root has been removed, the heap is no longer complete because of the newly made hole, and it must be replaced with the next elements in line."
>
> From Data Structures and Algorithms for Game Developers, by Allen Sherrod, page 329



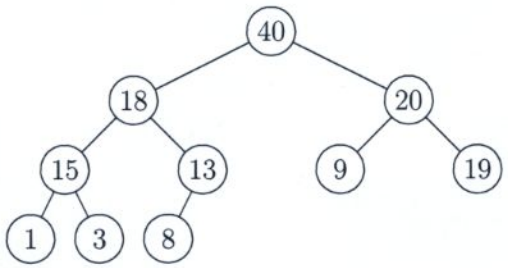

[5]

When removing an item from a heap, we remove the item at position 0 – the root. In order to maintain the tree structure, we replace that root item with the last item in the heap, remembering that the heap fills from left-to-right.

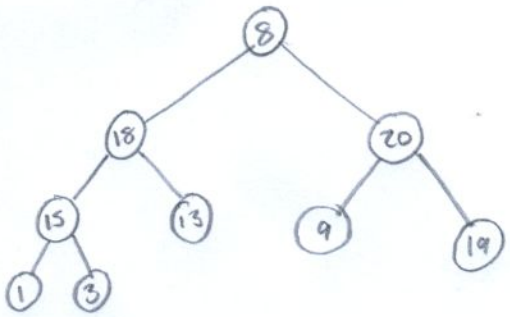
[5] From Data Abstraction & Problem Solving with C++: Walls and Mirrors 7th ed, by Carrano and Henry, page 521

**Question 4** ____ / 2

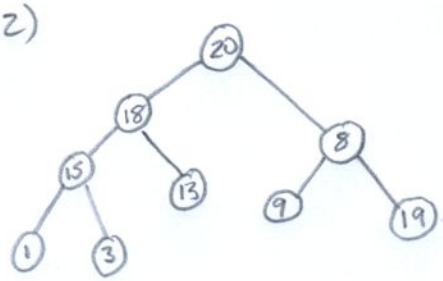
With the heap given, draw each step as you remove the root, replace it with the last node, and then bubble down to restore the heap to a valid state.




1)




2)




3)