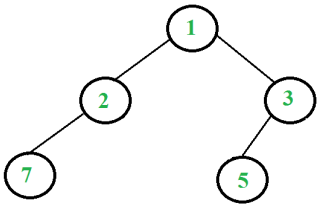
WORKSHEET WEEK 05: TREES AND HEAPS

5.1 Trees

Introduction Binary trees are often represented as arrays (where the array starts with the root node; followed by all the other nodes, displayed layer by layer. If any child of a node in this tree is missing, it is replaced by Λ (capital Lambda denoting an empty node) in the array. Once we reach the last non-empty node in the tree, this is the last element of the array. For example, the binary tree shown in this picture:



It is represented by the following array:

$$\texttt{int a}[\,] \;=\; \{1, 2, 3, 7, \Lambda, 5\};$$

Question 1.2.1: Assume that you have a binary tree that is represented by the following array:

$$\texttt{int a}[\,] \;=\; \{1, 2, 4, \texttt{a}, \Lambda, \Lambda, 6, \texttt{b}, \Lambda, \Lambda, \Lambda, \Lambda, \Lambda, \Lambda, \texttt{c}\};$$

Values a, b, c are the last three digits taken from your Student ID.

(A) Draw the binary tree represented by the above array in your answer. The tree should look nice: Draw left children to the left (and right children to the right) of their parents. Nodes on the same levels should be aligned.

(B) What is the number of internal nodes in this tree? The number of leaves in this tree?

(C) List the vertices of this tree in the post-order traversal order. (Only show real nodes in the post-order sequence (all Λ are technical symbols indicating absence of nodes; they are not part of the tree).

(D) Write pseudo-code for an algorithm GETPARENT(i) that receives the index i of some node in this array, returns the index of the parent of this node (or -1, if the node has no parent). All indices i are zero-based (in an array of length 10, $i \in \{0, \ldots, 9\}$).

(E) Assume that there is a different array (representing another binary tree) which does not contain any Λ values; all values there represent some nodes. Describe the property such trees must satisfy.

5.2 Heaps

Question 1.3.1:

(A) Assume that heap is implemented as a 0-based array (the root element is `H[0]`), and the heap supports DeleteMin(H) operation that removes the minimum element (and returns the heap into consistent state).

Find, if the heap property holds in the following array:

$$H[0] = 6, 17, 25, 20, 15, 26, 30, 22, 33, 31, 20.$$

If it is not satisfied, find, which two keys you could swap in this array so that the heap property is satisfied again. Write the correct sequence of array H.

Note: A *consistent state* in a minimum heap means that the key in parent does not exceed keys in left and right child.

(B) Assume that heap is implemented as a 0-based array (the root element is `H[0]`), and the heap supports DeleteMax(H) operation that removes the maximum element.

If the heap does not satisfy invariant (in a consistent max-heap, every parent should always be at least as big as both children), then show how to swap two nodes to make it correct.

$$96, 67, 94, 10, 67, 68, 69, 9, 10, 11, 50, 67.$$

Question 1.3.2 (Insert into a min-heap): Show what is the final state of a heap after you insert number 6 into the following minimum-heap (represented as a zero-based array):

$$9, 18, 28, 23, 20, 29, 33, 25, 36, 34, 23.$$

Question 1.3.3 (Delete maximum from a Max-Heap): Show what is the final state of a heap after you remove the maximum from the following heap (represented as a zero-based array):

$$96, 67, 94, 10, 67, 68, 69, 9, 10, 11, 50, 67.$$

Question 1.3.4 (Removing from Maximum Heap): Here is an array for a Max-Heap:

16	14	10	8	7	9	3	2	4	1

The image shows array used to store Maximum Heap (a data structure allowing inserts and removal of the maximum element). The array starts with the 0-th element (and any parent node in such tree should always be at least as big as any of its children).

(A) Draw the initial heap based on this array. Heap should be drawn as a complete binary tree.

(B) Run the command DeleteMax(H) on this initial heap. Draw the resulting binary tree (after the heap invariant is restored – any parent node is at least as big as its children). Draw the binary tree image you get.

(C) On the tree that you got in the previous step (B) run the command Insert(H, x), where $x = a + b + c$ is the sum of the last three digits of your student ID. Draw the binary tree image you get.

(D) Show the array for the binary tree you got in the previous step (C) (i.e. right after the DeleteMax(H) and Insert(H, x) commands have been executed).