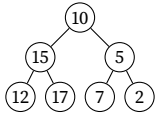
Level-order

A *level-order* traversal visits the nodes in a tree by *depth*. That is, the root is visited first, then all the children of the root, then all the children of the root's children, and so on.

A *level-order* traversal of the following tree can, for example, visit the nodes in the order $10, 15, 5, 12, 17, 7, 2$.



(a) Assume that the tree is a *binary heap*, represented by an array A with n elements. Give an algorithm that returns a list with the elements from A in *level-order*.

```
Input: A binary heap, represented by an array A with n elements
Output: A list with the elements in the heap in level-order
1 Procedure LevelOrderHeap(A)
  |   // ···
```

(b) Assume that the tree is a *binary search tree* with n nodes, where each node v has the following fields:

- v.`element` — the value stored in the node
- v.`left` — the left child of v

- v.`right` — the right child of v

Give an algorithm that returns a list with the elements from the tree in *level-order*.

```
Input: The root v of a binary search tree with n nodes
Output: A list with the elements in the search tree in level-order
1 Procedure LevelOrderBST(v)
  |  // ···
```

(c) State the runtime complexity for your algorithms in (a) and (b).

(d) Consider the following algorithm:

```
Input: The root v of a binary search tree
Output: A new binary search tree
1 Procedure ConstructBST(v)
2 |  B ← empty binary search tree
3 |  nodes ← LevelOrderBST(v)
4 |  for u in nodes do
5 |  |  B ← Insert(B, u)
6 |  return B
```

What can you say about the relation between the input tree and the output tree?