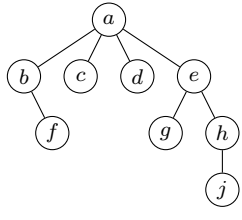
Lowest Common Ancestor

The *lowest common ancestor* of two nodes u and v in a tree T is the lowest (i.e., the node with the highest depth) node that has both u and v as descendants.

Here are some examples from the tree on the right:

- The lowest common ancestor of a and a is a
- The lowest common ancestor of b and e is a
- The lowest common ancestor of g and j is e
- The lowest common ancestor of e and j is e



(a) Briefly explain why there must exist a common ancestor for every pair of nodes.

(b) Assume that if v is a node, v has only the following fields:

- v.`parent` – parent node of v
- v.`children` – children nodes of v
- v.`depth` – depth of v

Provide an algorithm that, given two nodes u and v, finds and returns their lowest common ancestor.

```
Input: Two nodes u and v
Output: Lowest common ancestor of u and v
1 Procedure LowestCommonAncestor(u, v)
  |  // ···
```

(c) Now assume that the tree is a *binary search tree of unique integers*. If v is a node, then v has only the following fields:

- v.`element` – the integer stored in the node
- v.`left` – left child of v
- v.`right` – right child of v

Provide an algorithm that, given the root of the tree and the elements x and y, returns the lowest common ancestor of the nodes that have the elements x and y, where $x < y$. You can assume that both elements x and y exist in the tree.

```
Input: A node v, and two values x and y where x < y
Output: Lowest common ancestor of the nodes that have the elements x and y
1 Procedure LowestCommonAncestorBST(v, x, y)
  |  // ···
```

(d) What is the runtime complexity of your algorithm from (c) for a tree with n nodes in the worst case? What is the runtime complexity if you assume that the tree is balanced?

If you did not answer (c), you can provide the answer for your solution to (b) instead.