Our Solution(s) Run Code

Solution 1

```
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

class AncestralTree {
  constructor(name) {
    this.name = name;
    this.ancestor = null;
  }
}

// O(d) time | O(1) space - where d is the depth (height) of the
function getYoungestCommonAncestor(topAncestor, descendantOne, 
  const depthOne = getDescendantDepth(descendantOne, topAncesto
  const depthTwo = getDescendantDepth(descendantTwo, topAncesto
  if (depthOne > depthTwo) {
    return backtrackAncestralTree(descendantOne, descendantTwo,
  } else {
    return backtrackAncestralTree(descendantTwo, descendantOne,
  }
}

function getDescendantDepth(descendant, topAncestor) {
  let depth = 0;
  while (descendant !== topAncestor) {
    depth++;
    descendant = descendant.ancestor;
  }
  return depth;
}

function backtrackAncestralTree(lowerDescendant, higherDescenda
  while (diff > 0) {
    lowerDescendant = lowerDescendant.ancestor;
    diff--;
  }
  while (lowerDescendant !== higherDescendant) {
    lowerDescendant = lowerDescendant.ancestor;
    higherDescendant = higherDescendant.ancestor;
  }
  return lowerDescendant;
}

exports.AncestralTree = AncestralTree;
exports.getYoungestCommonAncestor = getYoungestCommonAncestor;

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```
// This is an input class. Do not edit.
class AncestralTree {
  constructor(name) {
    this.name = name;
    this.ancestor = null;
  }
}

function getYoungestCommonAncestor(topAncestor, descendantOne, d
  // Write your code here.
}

// Do not edit the line below.
exports.AncestralTree = AncestralTree;
exports.getYoungestCommonAncestor = getYoungestCommonAncestor;

```

Custom Output Submit Code

Run or submit code when you're ready.