Our Solution(s) Run Code

Solution 1

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

package main

type AncestralTree struct {
  Name     string
  Ancestor *AncestralTree
}

// O(d) time | O(1) space - where d is the depth (height) of the
func GetYoungestCommonAncestor(topAncestor, descendantOne, desc
  depthOne := getDescendantDepth(descendantOne, topAncestor)
  depthTwo := getDescendantDepth(descendantTwo, topAncestor)
  if depthOne > depthTwo {
    return backtrackAncestralTree(descendantOne, descendantTwo,
  }
  return backtrackAncestralTree(descendantTwo, descendantOne, d
}

func getDescendantDepth(descendant, topAncestor *AncestralTree)
  depth := 0
  for descendant != topAncestor {
    depth++
    descendant = descendant.Ancestor
  }
  return depth
}

func backtrackAncestralTree(lowerDescendant, higherDescendant *
  for diff > 0 {
    lowerDescendant = lowerDescendant.Ancestor
    diff--
  }
  for lowerDescendant != higherDescendant {
    lowerDescendant = lowerDescendant.Ancestor
    higherDescendant = higherDescendant.Ancestor
  }
  return lowerDescendant
}

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```go
package main

type AncestralTree struct {
  Name     string
  Ancestor *AncestralTree
}

func GetYoungestCommonAncestor(top, descendantOne, descendantTwo
  // Write your code here.
  return nil
}

```

Custom Output Submit Code

Run or submit code when you're ready.