

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 #include <vector>
4 using namespace std;
5
6 class AncestralTree {
7 public:
8     char name;
9     AncestralTree *ancestor;
10
11     AncestralTree(char name) {
12         this->name = name;
13         this->ancestor = NULL;
14     }
15
16     void addAsAncestor(vector<AncestralTree *> descendants);
17 };
18
19 int getDescendantDepth(AncestralTree *descendant, AncestralTree *topAncestor);
20 AncestralTree *backtrackAncestralTree(AncestralTree *lowerDescendant,
21                                     AncestralTree *higherDescendant,
22                                     int diff);
23
24 // O(d) time | O(1) space - where d is the depth (height) of the ancestral tree
25 AncestralTree *getYoungestCommonAncestor(AncestralTree *topAncestor,
26                                     AncestralTree *descendantOne,
27                                     AncestralTree *descendantTwo) {
28     int depthOne = getDescendantDepth(descendantOne, topAncestor);
29     int depthTwo = getDescendantDepth(descendantTwo, topAncestor);
30     if (depthOne > depthTwo) {
31         return backtrackAncestralTree(descendantOne, descendantTwo,
32                                     depthOne - depthTwo);
33     } else {
34         return backtrackAncestralTree(descendantTwo, descendantOne,
35                                     depthTwo - depthOne);
36     }
37 }
38
39 int getDescendantDepth(AncestralTree *descendant, AncestralTree *topAncestor) {
40     int depth = 0;
41     while (descendant != topAncestor) {
42         depth++;
43         descendant = descendant->ancestor;
44     }
45     return depth;
46 }
47
48 AncestralTree *backtrackAncestralTree(AncestralTree *lowerDescendant,
49                                     AncestralTree *higherDescendant,
50                                     int diff) {
51     while (diff > 0) {
52         lowerDescendant = lowerDescendant->ancestor;
53         diff--;
54     }
55     while (lowerDescendant != higherDescendant) {
56         lowerDescendant = lowerDescendant->ancestor;
57         higherDescendant = higherDescendant->ancestor;
58     }
59     return lowerDescendant;
60 }
```

Solution 1

Solution 2

Solution 3

```
1 #include <vector>
2 using namespace std;
3
4 class AncestralTree {
5 public:
6     char name;
7     AncestralTree *ancestor;
8
9     AncestralTree(char name) {
10         this->name = name;
11         this->ancestor = NULL;
12     }
13
14     void addAsAncestor(vector<AncestralTree *> descendants);
15 };
16
17 AncestralTree *getYoungestCommonAncestor(AncestralTree *topAncestor,
18                                     AncestralTree *descendantOne,
19                                     AncestralTree *descendantTwo) {
20     // Write your code here.
21     return NULL;
22 }
23
```

Custom Output

Raw Output

Submit Code

Run or submit code when you're ready.