Tarjan's least common ancestor

April 6, 2016

This took me a little while to figure out. Let's start out with a related problem: range-minimum query[1]. Consider the tree

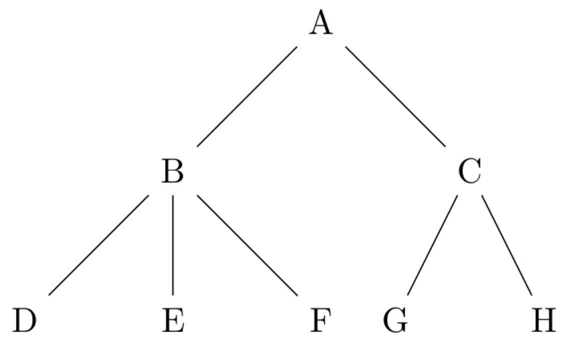


Figure 1: Example tree

You can build an array with property that the least common ancestor of any two nodes in the tree can be determind by finding the minimum of some related range in the array. For example suppose the corresponding array to the example tree is

$$T = [0, 1, 2, 1, 2, 1, 2, 1, 0, 1, 2, 1, 2, 1, 0]$$

where the element-wise correspondence is

$$T' = [A, B, D, B, E, B, F, B, A, C, G, C, H, C, A]$$

Then the least common ancestor (LCA) of any two nodes u, v in T' is the node corresponding to the minimum of the range in T' corresponding to$[u, v]$. For example $LCA(\{D, F\}) = \min([2, 1, 2]) = 1 = B$. This looks purely coincidental and hinky but it's true. Try it for any pair of nodes. It's not coincidental because of how T was constructed[2] from the example tree: it's an Euler circuit of the corresponding digraph[3] to the tree. Constructing it was easy: pre-order walk but print when you come back up to the root too:

[1]I stole most of this from http://wcipeg.com/wiki/Lowest_common_ancestor#Properties

[2]Also notice that T' is the depth of the nodes in the tree.

[3]A traversal of the tree that starts and ends at the same node, and visits every edge exactly once (the digraph corresponding to the tree has two edges for every edge in the undirected tree).

Algorithm 1 Euler circuit

```
Euler-Circuit($u$)
    print $u$
    for $v$ of $u$
        Euler-Circuit($v$)
        print $u$

Euler-Circuit($A$)
```

Here's an example of how it looks on another tree

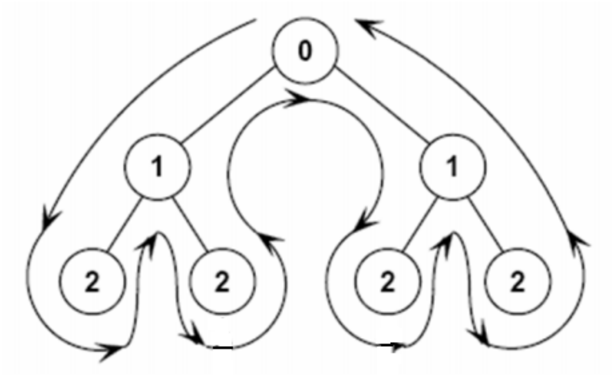


Figure 2: Example Euler circuit

Every time the traversal passes by a node it prints. The output is 0121210121210.

The Euler circuit has the property that between visiting u and v the $LCA(\{u,v\})$ is guaranteed to be visited at least once, and no other node with depth less than or equal to that of $LCA(\{u,v\})$ will appear at all. For example, consider the nodes D and F in T': just like $1 \in T$, node B appears at least once and no shallower node appears. Think about it: you get "stuck" under the least common ancestor of u, v. Fact: since a tree with n nodes has $n-1$ edges and the Euler circuit walks each one in the digraph corresponding to the tree, it has length $2n-2$ and visits $2n-1$ nodes.

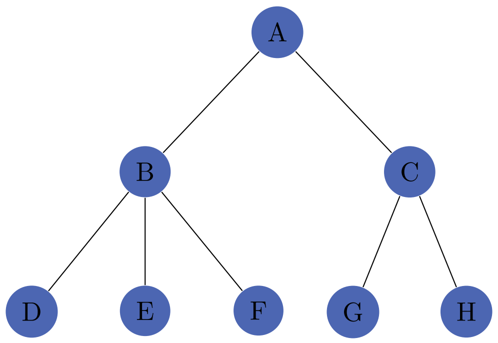
Okay how do we use this to compute least common ancestor? Here is Tarjan's least common ancestor algorithm that uses Union-Find data structures:

Algorithm 2 Tarjan's algorithm

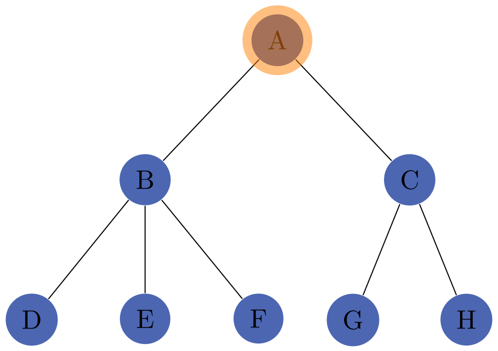
```
global P // set of pairs you want to find common ancestors for

for $u$ in Tree:
    $u.color$ := BLUE

LCA($u$)
    Make-Set($u$)
    Find-Set($u$).ancestor := $u$
    for each child of $v$ of $u$
        LCA($v$)
        Union($u,v$)
        Find-Set($u$).ancestor := $u$
    $u.color$ := RED
    for each $v$ such that $\{u,v\} \in P$
        if $v.color$ == RED
            print "LCA($\{u,v\}$) is: " Find-Set($v$).ancestor
```

Notice the similarity to `Euler-Circuit.` The algorithm proceeds growing "bottom up" sets corresponding to subtrees whose roots are the least common ancestor of any pair of nodes in the tree **which have been completely traversed by the Euler circuit**, i.e. colord `RED`. A picture is worth a thousand words so here are bunch: I'll walk through the operation of the algorithm if P is every pair of nodes in the Example tree.

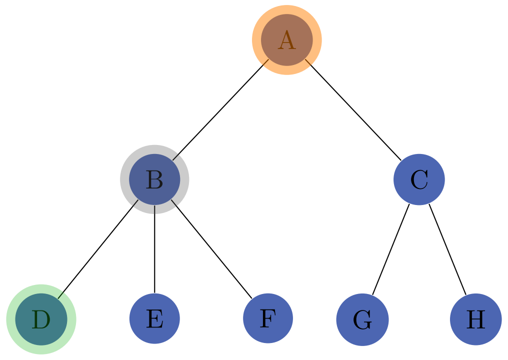
1.



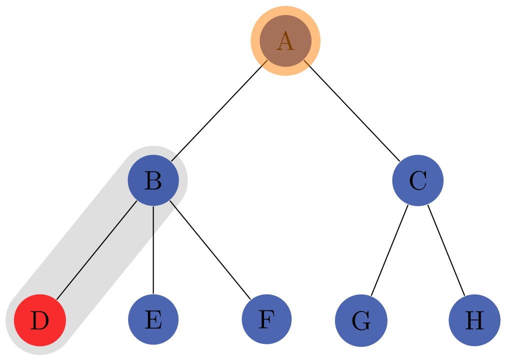
2.



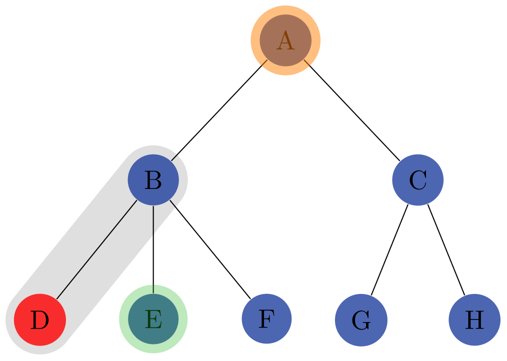
3.



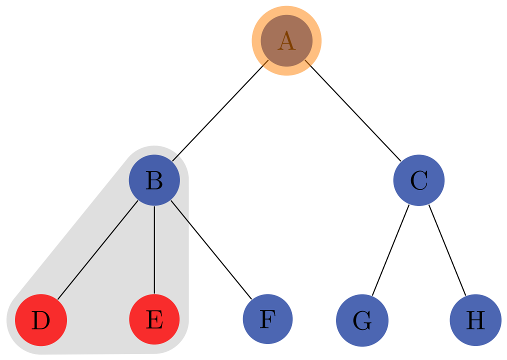
4.



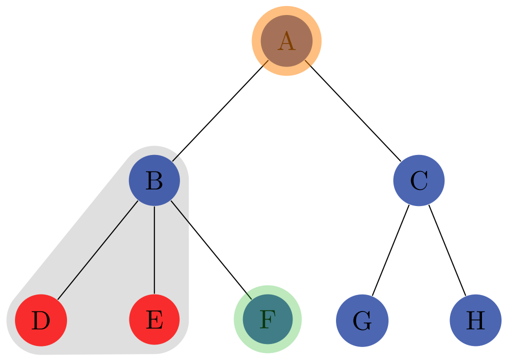
5.



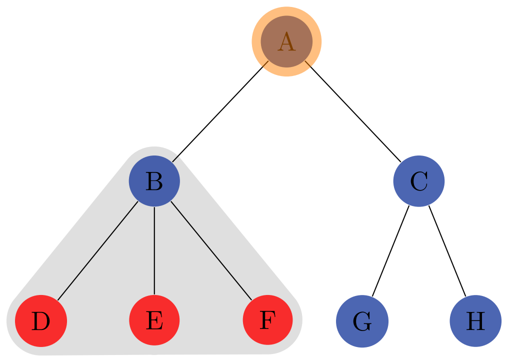
6.



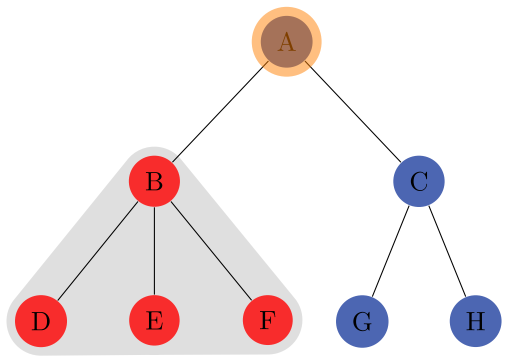
7.



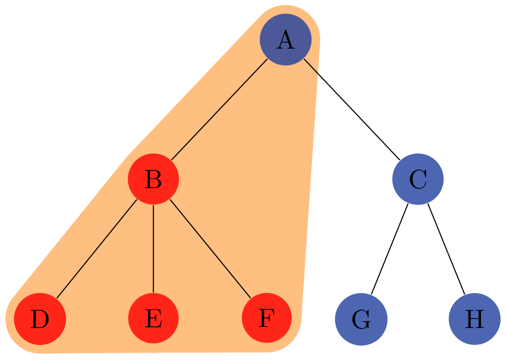
8.



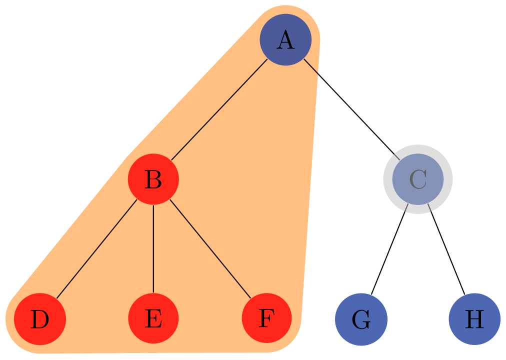
9.



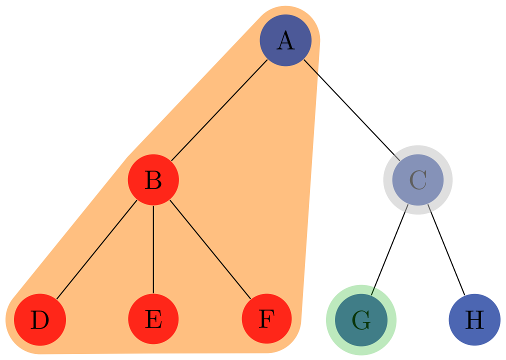
10.



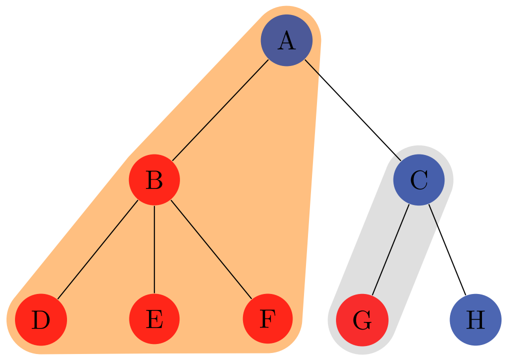
11.



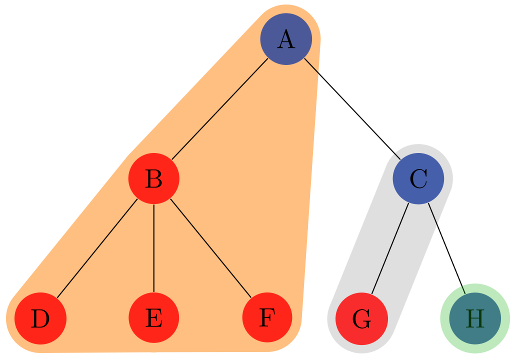
12.



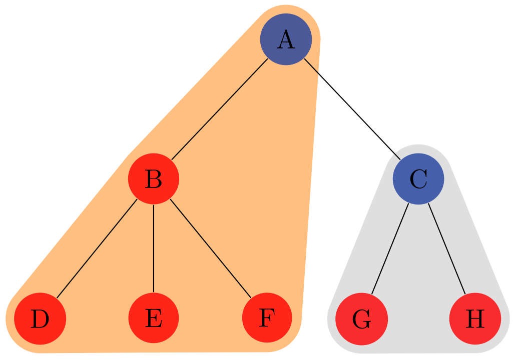
13.



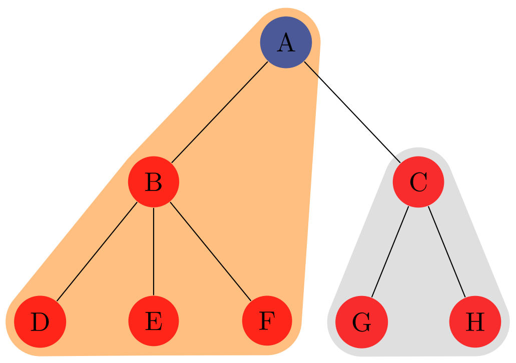
14.



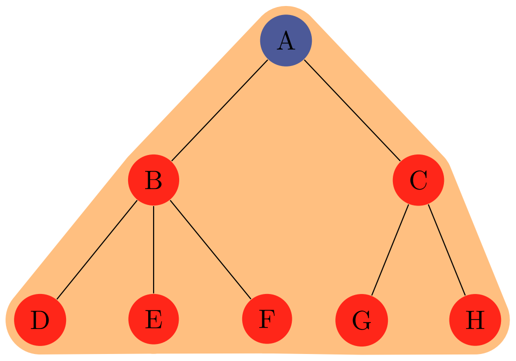
15.



16.



17.



18.