LeetCode 235

Abhinav Ganesh

January 28, 2024

235. Lowest Common Ancestor of a Binary Search Tree

Prompt: Given a binary search tree (BST), find the lowest common ancestor (LCA) node of two given nodes in the BST.
According to the definition of LCA on Wikipedia: "The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow a node to be a descendant of itself)."

Constraints:

1. The number of nodes in the tree is in the range [2, 105].
2. -109 $\leq$ Node.val $\leq$ 109
3. All Node.val are unique.
4. p != q
5. p and q will exist in the BST.

Solution:

Let the inputs be BST B, and $p, q \in V$ s.t. $p \neq q$.
Recall that the definition of a BST is that:

1. all elements in the left subtree of a node must be less than that element and all elements in the right subtree of a node must be less than that element.
2. all subtrees of the BST are also BST's
3. all elements in a BST are unique (can be derived from the other 2 properties).

We want the LCA l such that l is the lowest node in B that has both p and q as descendants where a node can be a descendant of itself.

Let us define this more strictly.

Given a node in $n \in B$, we can recursively define its <u>descendants</u> as $DESC(n)$:

1. $\{l\} \in DESC(l)$
2. if $n \in DESC(n)$ then both its left child and its right child must also be in $DESC(n)$.

Note that this means $DESC(l) = \{l\} \cup DESC(\text{left child of l}) \cup DESC(\text{right child of l})$. Note that $DESC(\text{left child of l})$ is the <u>left subtree</u> of l and $DESC(\text{right child of l})$ is the <u>right subtree</u> of l.
Observe that n is a <u>common ancestor</u> of p, q if $p, q \in DESC(n)$.

We also need to define what it means to be the lowest common ancestor of p, q in B. l is the <u>lowest common ancestor</u> of p, q in B if:

1. l is a common ancestor of p, q
2. there exists no common ancestor l' of p, q where $l' \in DESC(l)$ and $l' \neq l$;

$$\nexists\ l' \in DESC(\text{left child of l}) \cup DESC(\text{right child of l}) \text{ s.t. } p, q \in DESC(l')$$

Realize that all nodes by definition must have the root of the tree as a common ancestor, so p, q must at least have the common ancestor as the root. So $\exists$ at least one common ancestor of p, q. Let us call it a.
There are 2 cases:

1. **Case 1:** Either $p = a$ or $q = a$. WLOG say $p = a$. If this is the case then there cannot exist any $l' \neq a \in DESC(a)$ s.t. $p \in DESC(l')$. Else the value p must appear twice in the tree, which contradicts the definition of BST. So a must be the lowest common ancestor in this case.

2. **Case 2:** $p \neq a$ and $q \neq a$. If this is the case then $p \in DESC(a) \setminus \{a\}$ and $q \in DESC(a) \setminus \{a\}$. Note that

$$DESC(a) \setminus \{a\} = DESC(\text{left child of a}) \cup DESC(\text{right child of a})$$

So $p, q \in DESC(\text{left child of a}) \cup DESC(\text{right child of a})$. Then consider the following subcases:

(a) **Subcase 2.a:** p,q in the same subtree of a; $p, q \in DESC(\text{left child of a})$ or $p, q \in DESC(\text{right child of a})$.
WLOG say $p, q \in DESC(\text{left child of a})$. Let this left child of a be called a'. By definition of common ancestor, a' is a common ancestor of p, q as $p, q \in DESC(a')$.

Further, as a' is the left child of a then $a' \in DESC(a)$ and $a' \neq a$ which means a cannot be the lowest common ancestor of p, q. Rather, a' is a lower common ancestor of p, q.
We must investigate $DESC(a')$ to see if there is an even lower common ancestor of p, q. That is, the lowest common ancestor of p, q must be in the left subtree of a.

Note that if we had taken $p, q \in DESC(\text{right child of a})$ then similar logic would conclude that p, q must be in the right subtree of a.

(b) **Subcase 2.b:** p,q in different subtrees of a; WLOG say $p \in DESC(\text{left child of a})$ but $q \in DESC(\text{right child of a})$.
By definition of BST, all elements in the left subtree of a must be $< a$ and all elements in the right subtree of a must be $> a$. Then $p < a$ and $q > a$.

Pick an arbitrary element $a' \in DESC(\text{left child of a}) \cup DESC(\text{right child of a})$. By definition of $\cup$ either $a' \in$ (left subtree of a) or $a' \in$ right subtree of a. Then consider the cases:

i. $a' \in$ left subtree of a. Then all nodes in $DESC(a')$ are also in the left subtree of a by the structure of a BST. But $q \in$ right subtree of a so $q \notin$ left subtree of a (as $q > a$). Then $q \notin DESC(a')$ so a' is not an ancestor of q and thus cannot be a lowest common ancestor of p, q.

ii. $a' \in$ right subtree of a. Then all nodes in $DESC(a')$ are also in the right subtree of a by the structure of a BST. But $p \in$ left subtree of a so $p \notin$ right subtree of a (as $p < a$). Then $p \notin DESC(a')$ so a' is not an ancestor of p and thus cannot be a lowest common ancestor of p, q.

Then it cannot be the case that a' is a common ancestor of p, q.

So $\nexists\ a' \in DESC(\text{left child of a}) \cup DESC(\text{right child of a})$ s.t. $p, q \in DESC(a')$. Then by definition of lowest common ancesetor, a is the lowest common ancestor of p,q.

Then we can take an algorithm that starts at the root (which must be a common ancestor of p,q) and follows the case logic described and we should eventually reach either case 1 or subcase 2.b which will stop and correctly return the lowest common ancestor of p, q.

Note that the algorithm will reach one of these subcases. If the algorithm does not stop then $\nexists$ a subtree in B where p, q are on opposite sides of the subtree's root. But this would imply that $p = q$ which contradicts our input constraints that $p \neq q$. Therefore the algorithm must terminate.