285. Inorder Successor in BST

Aug. 13, 2019 | 25.4K views

Given a binary search tree and a node in it, find the in-order successor of that node in the BST.

The successor of a node `p` is the node with the smallest key greater than `p.val`.

Example 1:

```
Input: root = [2,1,3], p = 1
Output: 2
Explanation: 1's in-order successor node is 2. Note that both p and the return value i
```

Example 2:

```
Input: root = [5,3,6,2,4,null,null,1], p = 6
Output: null
Explanation: There is no in-order successor of the current node, so the answer is n
ull.
```

Note:

1. If the given node has no in-order successor in the tree, return `null`.
2. It's guaranteed that the values of the tree are unique.

Solution

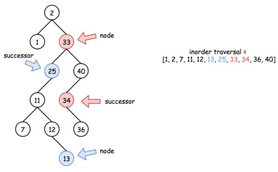
Approach 1: Iterative Inorder Traversal

Inorder traversal of BST is an array sorted in the ascending order.

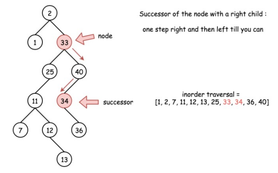
> Successor is the smallest node in the inorder traversal *after* the current one.

There could be two situations :

1. If the node has a right child, the successor is somewhere lower in the tree, see red nodes on the Fig. below.
2. Otherwise, the successor is somewhere upper in the tree, see blue nodes on the Fig.



Let's first check the simple case 1. To find a successor, go one step right and then left till you can.



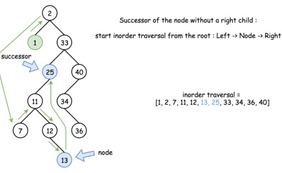
This approach has $O(H_p)$ time complexity, where H_p is a height of the node p.

Now let's consider the case 2. There is no access to the parent nodes here, and hence one has to traverse the tree starting from the root and *not* from the node.

Inorder traversal could be implemented in three different ways: recursive, iterative and Morris. Here it's better to choose the iterative traversal in order to optimise the performance and minimize the space at the same time.

> Iterative inorder traversal is simple: go left till you can, and then one step right. Repeat till the end of nodes in the tree.

The idea is to keep just one previous node during the inorder traversal. If that previous node is equal to `p`, then the current node is a successor of `p`.



This approach has $O(H)$ time complexity, where H is a tree height. Basically, this approach is universal and could be used for the case 1 as well. We simply don't do that because the approach 1 is faster.

Algorithm

- If the node has a right child, go one step right and then left till you can. Return the successor.
- Otherwise, implement iterative inorder traversal. While there are still nodes in the tree or in the stack:
  - Go left till you can, adding nodes in stack.
  - Pop out the last node. If its predecessor is equal to `p`, return that last node. Otherwise, save that node to be the predecessor in the next turn of the loop.
  - Go one step right.
- If we're here that means the successor doesn't exit. Return null.

Implementation

Java | Python

```
class Solution {
  public TreeNode inorderSuccessor(TreeNode root, TreeNode p) {
    // the successor is somewhere lower in the right subtree
    // successor: one step right and then left till you can
    if (p.right != null) {
      p = p.right;
      while (p.left != null) p = p.left;
      return p;
    }

    // the successor is somewhere upper in the tree
    ArrayDeque<TreeNode> stack = new ArrayDeque<>();
    int inorder = Integer.MIN_VALUE;

    // inorder traversal : left -> node -> right
    while (!stack.isEmpty() || root != null) {
      // 1. go left till you can
      while (root != null) {
        stack.push(root);
        root = root.left;
      }

      // 2. all logic around the node
      root = stack.pop();
      // if the previous node was equal to p
      // then the current node is its successor
      if (inorder == p.val) return root;
```

Complexity Analysis

- Time complexity : $O(H_p)$ in the best case, when node `p` has a right child. Here H_p is a height of node `p`. $O(H)$ in the worst case of no right child. Here H is a tree height.
- Space complexity : $O(1)$ in the best case, when node `p` has a right child. Otherwise, up to $O(H)$ to keep the stack.

Analysis written by @liaison and @andvary

Rate this article:

Previous Next

Comments: 9

Sort By

Type comment here... (Markdown is supported)

Preview Post

dyckia ★ 128 November 23, 2019 12:29 AM

Here is a much simpler solution to the problem. The idea is pretty straight forward.
We start from the root, utilizing the property of BST:

- If current node's value is less than or equal to p's value, we know our answer must be in the right subtree.

Read More

74 Share Reply

SHOW 4 REPLIES

jianwu ★ 590 August 14, 2019 6:18 PM

For case 2, complexity should be O(n) instead

31 Share Reply

SHOW 4 REPLIES

haryyu1994 ★ 31 December 21, 2019 1:34 AM

This is not the optimal solution for this problem though. It doesn't take advantage of the properties of Binary Search Tree.

7 Share Reply

lcpremium101 ★ 1 September 24, 2019 4:04 PM

An O(1) space complexity solution:

```
def inorderSuccessor(self, root, p):
        # If p has a right child, successor is the min node in the right child
        if p.right:
```

Read More

5 Share Reply

nickconnolder ★ 103 August 31, 2019 1:46 AM

I rarely take the time to write things on the comments but this time I have to stop and give you a big thank you. This problem and your solution made me finally understand inorder iterative using a stack. Also finally understood that p is a pointer inside the same tree just like root is the pointer to the start of the tree. That must have been what threw me off initially.

5 Share Reply

SHOW 1 REPLY

nmam ★ 168 August 19, 2019 6:42 AM

Nice Article

3 Share Reply

tkg37 ★ 12 August 15, 2019 1:08 AM

Assume we had parent pointer available for the given node, then in the case 2, successor would be just the go uptill root from the current node, and the last left turn we took while doing this, is my successor.

2 Share Reply

SHOW 4 REPLIES

eefaafra ★ 445 November 23, 2019 10:03 AM

FYI: @andvary / @liaison

"If p has a right child, successor is somewhere lower, so step right and go left as much as you can"
This can be more formally written as:
"The successor of a node is the left-most node in its right sub-tree (if it exists). Otherwise it is the first

Read More

1 Share Reply

lei0108 ★ 121 February 26, 2020 7:03 AM

Simpler:

```
public TreeNode inorderSuccessor(TreeNode root, TreeNode p) {
        TreeNode succ = null;
        while (root != null) {
```

Read More

0 Share Reply

SHOW 1 REPLY