Find kth smallest and largest element in BST.

Posted on September 29, 2020 | by Prashant Yadav

Posted in Algorithms, Tree | Tagged Easy

Given a binary search tree (BST), find the kth largest and smallest element in it.

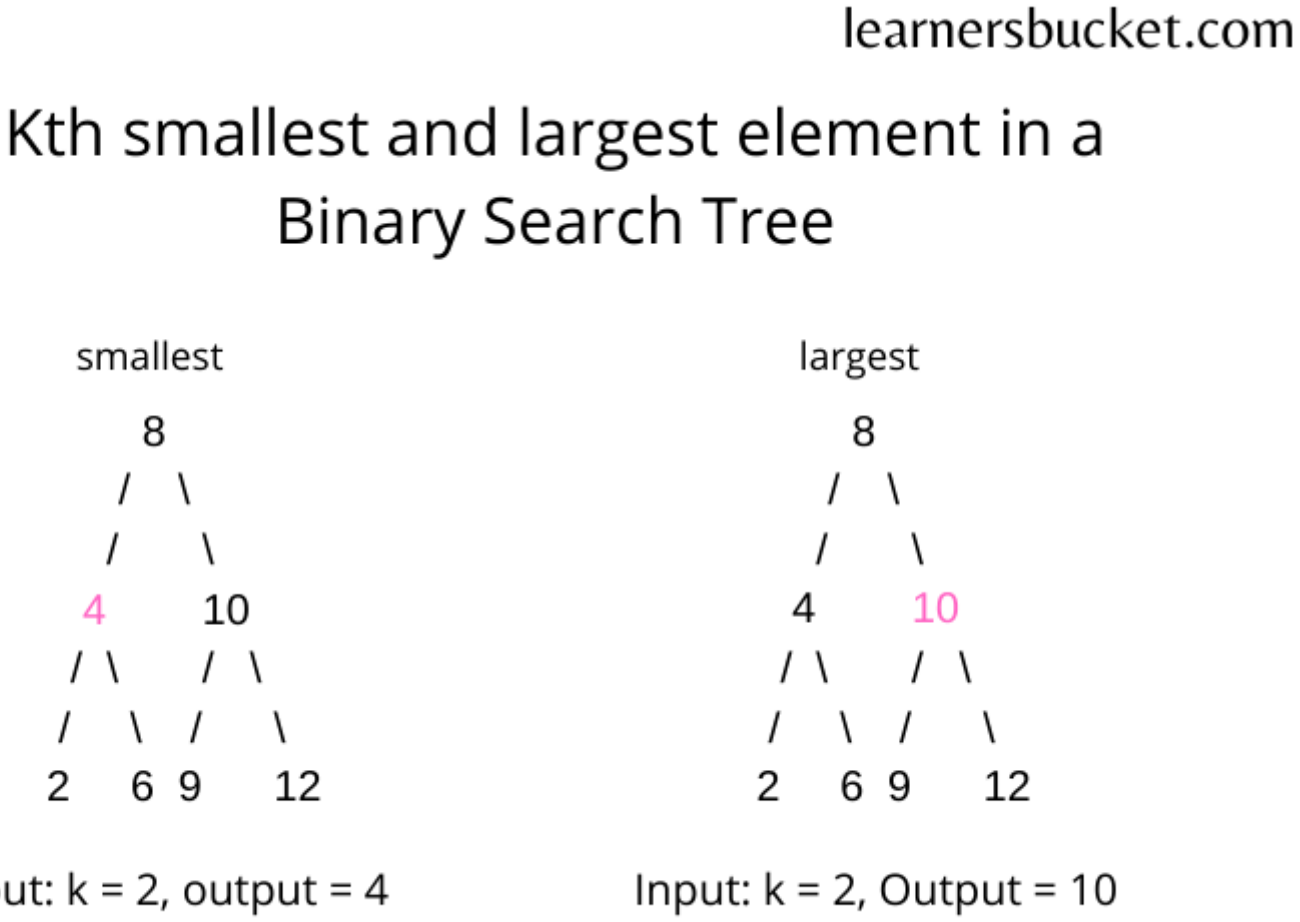
Example

Copy

```
Input:
        10
       /  \
      /    \
     6      11
    / \    / \
   /   \  /   \
  5    7 8    12

Smallest: k = 2;
Largest: k = 2;

Output:
6
11
```



Reading the problem statements give us a hint that to find the to find the k'th largest and smallest element, we should access the data in sorted order.

For which we can perform the in-order traversal on binary search tree as it returns the node in ascending order and store the nodes in an array and then return the kth smallest and largest element from the array.

The only problem with this approach is we need to store the elements before accessing it. But it can be optimized further to solve it in constant space.

By performing the in-order traversal on the tree, but instead of storing the nodes, we can use variables to track the kth node.

Finding kth smallest element in BST.

For the smallest element we start from lowest to highest.

Conceptually this is how it works

- Use a counter to track the kth position.
- For the smallest element first, check in the left subtree, (where smallest elements are there) and else if the current is element is the kth then return or check in the right subtree.

```
const kthSmallest = (root, count, k) => {
  // base case
  if(root === null){
    return null;
  }

  // find in left subtree
  let left = kthSmallest(root.left, count, k);

  // if found then return it.
  if(left !== null){
    return left;
  }

  // increment the count
  count.i++;

  // if current element is the k'th smallest then return it
  if(count.i === k){
    return root.val;
  }

  // find in right subtree
  return kthSmallest(root.right, count, k);
}
```

```
Input:
function Node(val) {
  this.val = val;
  this.left = null;
  this.right = null;
}

const tree = new Node(10);
tree.left = new Node(6);
tree.right = new Node(11);
tree.left.left = new Node(5);
tree.left.right = new Node(7);
tree.right.left = new Node(8);
tree.right.right = new Node(12);

console.log(kthSmallest(tree, {i:0}, 3));

Output:
7
```

Finding the kth largest element in BST.

For the largest element we will move from highest to lowest.

Same as how we searched the smallest, the same procedure can be done for the largest one with the only difference that we will look in the right subtree first (largest elements) and then check if the current element is kth then return it or else look in the left subtree.

```
const kthLargest = (root, count, k) => {
  //base case
  if(root === null){
    return null;
  }

  //find in the right subtree
  let right = kthLargest(root.right, count, k);

  //if found in right subtree then return it.
  if(right !== null){
    return right;
  }

  //increment the count
  count.i++;

  //if current element is the k'th largest then return it
  if(count.i === k){
    return root.val;
  }

  //find in left subtree
  return kthLargest(root.left, count, k);
}
```

```
Input:
function Node(val) {
  this.val = val;
  this.left = null;
  this.right = null;
}

const tree = new Node(10);
tree.left = new Node(6);
tree.right = new Node(11);
tree.left.left = new Node(5);
tree.left.right = new Node(7);
tree.right.left = new Node(8);
tree.right.right = new Node(12);

console.log(kthLargest(tree, {i:0}, 2));

Output:
11
```

Time complexity for both the solution is O(h) where h is the height of the tree and O(h) space (if call stack is considered).

Recommended Posts:

- Find the intersection point of two linked list
- Program to print the Collatz sequence in javascript.
- Serialize and Deserialize Binary Tree
- Alternatively merge two different arrays

Find the correct position to insert an element in the array

Merge sort a linked list

Sum and Product of all the nodes in the linked list which are less than k

Find non duplicate number in an array

LRU cache in Javascript

Print matrix in L pattern

Prev Next

Keep Learning

Email

Subscribe