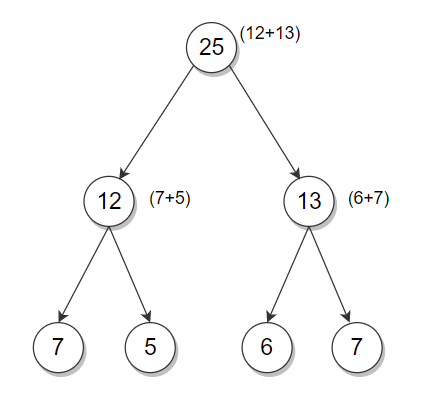
**Problems**

1. **Check the children-sum property in a binary tree**

Given the root of a binary tree, determine if the binary tree holds the children-sum property. For a tree to satisfy the children-sum property, each node’s value should be equal to the sum of values at its left and right nodes. The value of an empty node is considered 0.

For example, the following binary tree holds the children-sum property:



Hints

The idea is to traverse the binary tree in a postorder fashion. For each non-leaf node, check if the node’s value is equal to the sum of values at its left and right subtree. If this relation does not hold for any node, then the binary tree does not hold the children-sum property.

1. **Calculate the sum of root to leaf digits in a binary tree**

Given a binary tree, where each node stores a value between 0 and 9, calculate the sum of the numbers created by the paths from the root to leaf. For example,

Une image contenant texte

Description générée automatiquement

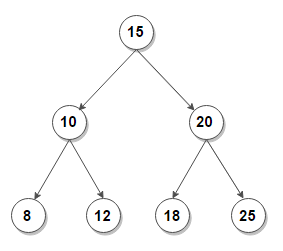
Hints

We can use recursion to solve the problem. The idea is to traverse the tree in a preorder fashion and construct the sum for each root-to-leaf path, as we go from top to bottom.

1. **Print a complete Binary Search Tree (BST) in increasing order**

Given a level order representation of a complete binary search tree, print its elements in increasing order.

For example, the level order representation of the complete BST below is [15, 10, 20, 8, 12, 18, 25]. The solution should print [8, 10, 12, 15, 18, 20, 25].



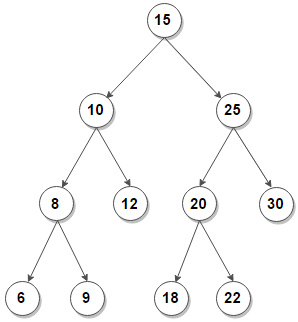
Hints

In the array representation of the binary tree, the left child for a node at index i occupies index 2i+1, and the right child occupies index 2i+2. For a complete binary tree, there will be no vacant positions in the array.

The idea is to process the array similarly as an inorder traversal of the binary tree using the above property since our binary tree is a BST – the inorder traversal prints the elements in increasing order.

1. **Count subtrees in a BST whose nodes lie within a given range**

Given a BST, count subtrees in it whose nodes lie within a given range. For example, consider the following BST. The total number of subtrees with nodes in the range [5, 20] is 6.



Une image contenant regarder

Description générée automatiquement

Hints

A simple solution would be to traverse the tree and, for each encountered node, check if all nodes under the subtree rooted under the node are within the given range or not. The time complexity of this solution is O(n2) for a binary search tree with n nodes. We can improve time complexity to linear by traversing the tree in a bottom-up manner and transferring some information from children to the parent node.

The idea is to perform a postorder traversal on the given BST. Then for any node, if both its left and right subtrees are within the range along with the node itself, we can say that the subtree rooted with this node is also within the range.

1. **Connect `n` ropes with minimal cost**

Given n ropes of different lengths, connect them into a single rope with minimum cost. Assume that the cost to connect two ropes is the same as the sum of their lengths. For example,

Une image contenant texte, lettre

Description générée automatiquement

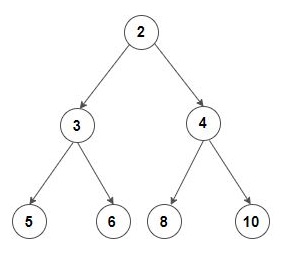
Hints

The idea is to connect the two lowest-cost ropes first. The resultant rope has a cost equal to the sum of the connected ropes. Repeat the process (with the resultant rope included) until we are left with a single rope.

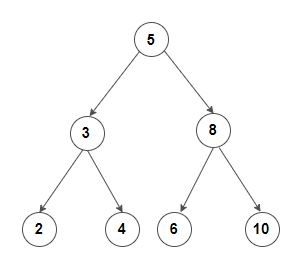
At each iteration of the loop, we will be left with one less rope, and the optimal cost is added to the total cost. The final cost for connecting n ropes will be minimal among all possible combinations. A priority queue implemented using min-heap is best suited for this problem.

1. **Check if a binary tree is a min-heap or not**

Given a binary tree, check if it is a min-heap or not. In order words, the binary tree must be a complete binary tree where each node has a higher value than its parent’s value. For example, the following binary tree is a min-heap:



On the other hand, the following binary tree is not a min-heap:



Hints

Recursive Solution. The idea is to traverse the tree in a preorder fashion. The value of each encountered node should be less than its left or right child. If that is not the case for every internal node, the binary tree is not a min-heap.

To check for a complete binary tree, the left and right child’s index for any node is less than the total number of nodes for every node. We can pass the index as a recursion parameter and check for every node that their left and right child’s index is within the correct range.