Calculate the Maximum Sum in the Valid Binary Search Tree Subtree [LeetCode](https://leetcode.com/problems/maximum-sum-bst-in-binary-tree/description/)

Given a **binary tree** root, return *the maximum sum of all keys of****any****sub-tree which is also a Binary Search Tree (BST)*.

Assume a BST is defined as follows:

* The left subtree of a node contains only nodes with keys **less than** the node's key.
* The right subtree of a node contains only nodes with keys **greater than** the node's key.
* Both the left and right subtrees must also be binary search trees.

Example:

1

/ \

4 3

/ \ / \

2 4 2 5

/ \

4 6

Output: The Maximum of Valid BST Subtree: 20

**Approach 1: Find the maximum sum of values in a Binary Search Tree (BST)**

* **Function Purpose**:
  + The **maxSumBST** function finds the maximum sum of values in a BST.
* **Explanation**:
  + It checks if the entire tree is a BST within the full range of **INT\_MIN** to **INT\_MAX**.
  + If the tree is a valid BST, it calculates the sum of values.
  + Negative sums are set to 0.
  + It recursively finds the maximum sum in the left and right subtrees.
  + Returns the maximum of these three values.
* **Time Complexity**:
  + **O(N^2), where N is the number of nodes in the BST due to repeated isBST and getSum calls.**
* **Space Complexity:**
  + **O(H), where H is the height of the BST, for the call stack space.**

**Approach 2: Find the maximum sum in a BST using an optimized approach**

* **Function Purpose**:
  + The **maxSumBSTOptimized** function finds the maximum sum of values in a BST using an optimized approach.
* **Explanation**:
  + It uses a helper function **maxSumBSTHelper** to calculate information about a node and its subtrees.
  + The helper function checks if the node and its subtrees form a BST and calculates the sum.
  + The maximum sum is updated while calculating this information.
* **Time Complexity**:
  + **O(N), where N is the number of nodes in the BST, as it traverses all nodes once.**
* **Space Complexity:**
  + **O(H), where H is the height of the BST, for the call stack space.**

**Conclusion**:

* **Approach 2 is more efficient with a lower time complexity, making it the preferred approach.**