Binary Search Tree to Sorted Doubly Linked List [CodeStudio](https://www.codingninjas.com/studio/problems/bst-to-sorted-dll_1263694?leftPanelTab=0)

You are provided with a BST you need to convert the BST to Sorted Doubly Linked List.

Example:

10

/ \

7 17

/ / \

5 15 19

/ /

1. 13

Output: Sorted DLL: [1 ⇌ 5 ⇌ 7 ⇌ 10 ⇌ 13 ⇌ 15 ⇌ 17 ⇌ 19]

**Approach 1: Convert a Binary Search Tree (BST) to a sorted doubly linked list using in-order traversal**

* **Function Purpose:** Converts a BST to a sorted doubly linked list using in-order traversal.
* **Explanation:**
  + The **bstToSortedDLL** function performs an in-order traversal of the BST.
  + It stores the nodes in a vector.
  + Then, it creates a doubly linked list from the stored nodes, connecting them in sorted order.
* **Time Complexity:** **O(N) - N is the number of nodes in the BST.**
* **Space Complexity: O(N) - For the vector that stores the nodes.**

**Approach 2: Convert a Binary Search Tree (BST) to a sorted doubly linked list using an optimized approach**

* **Function Purpose:** Converts a BST to a sorted doubly linked list using an optimized approach.
* **Explanation:**
  + The **covertBSTToSortedDoublyLinkedList** function traverses the BST in reverse in-order (right-root-left).
  + It maintains a **head** pointer for the doubly linked list and updates it as nodes are added, creating the list in sorted order.
* **Time Complexity:** **O(N).**
* **Space Complexity: O(H) - H is the height of the tree.**

**Conclusion:**

* Approach 1 is memory-intensive as it stores all nodes in a vector.
* **Approach 2 (optimized) uses less memory (O(H)) and is more efficient.**
* The optimized approach is better for converting a BST to a sorted doubly linked list.