hw4b algorithm:

To implement the vertical sum algorithm two datastructures will be used: tree to store the numbers and a linked list to keep the sums.

To assign the positive/negative labels (henceforth reffered to as the x-index) we use a pre order traversal algorithm. The end goal of this is to add the vertical sums of the binary tree, so it should be noted that no real emphasis is placed on preserving the x-index of the node.

The algorithm will first add the root node into a linked list, it will then go to the left node and prepend the child node to a double linked list, and then append the right node to the linked list. Each linked list node contains the x-index value as well as a sum counter.

The algorithm will continue doing the same with the children, except before it appends or prepends it will check the appropriate neighbor to see if the x-index exist. If the x-index already exist get the value of the tree node and add it to the linked Node's sum counter and store the new result in the sum counter.

Repeat until all nodes have been traversed.

This algorithm takes advantage of the fact that the x-index can only change by one from a parent to a child, so it uses a double linked list to be able to keep track of the two nearest neighbors. It also takes advantage that the end goal is to sum up the vertical numbers and not keep the x-index tied to a specific node (however if needed, the x-index can be assigned to the tree node easily).

Tree traversal takes O(n), adding the elements to the double linked list can be done in constant time, so the final complexity is O(n).

The space complexity is O(n) for the tree elements, the space complexity for the linked list can range from O(logn) for a balanced tree, to O(n) for a skewed tree.