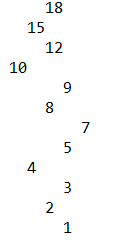
Lab3: The program bst.py in the class webpage contains implementations of basic binary search tree operations, including insertion, deletion, search and display.

1. Display the binary search tree as a figure, as shown below

This part is to use the Insert function to create the BST. To insert the numbers should read the Tree the type in by order

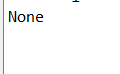
left to right, bot to top

1. Iterative version of the search operation.

This function is just like the normal search function, but not the recursion.

The idea of this function is use a loop to check, if the item is bigger then go to left, other wise will go to right

Ex:find 2 

Find: 20

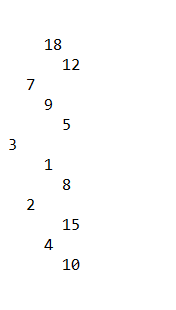
The only problem of is function is need to add .item to the return statement, other wise will print the address. If with the .item, the will be error for the number not in the tree.

1. Building a balanced binary search tree given a sorted list as input.

For this part, my idea is to find the middle number of the list, and set it to the root.

Then break from middle, as left and right part.

For those two part, break form middle and add to left/right(repeat the last two steps)

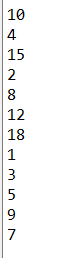


1. Extracting the elements in a binary search tree into a sorted list.

For this function, I decide to create a empty linked list and use loop to insert the numbers of the tree.

First to add the root, then add left and right item in to the list, and keep going.

Then return the list and print out.

but the order is follow by tree not the origin list

1. Printing the elements in a binary tree ordered by depth.

The main idea is to locate the depth and print

So I use two inputs, the T(tree) and k(the number of depth)

The code will keep go to next level, and k will decrease

When the k is 0, mean arrive the depth, then to print all number in that depth.

Ex:1

Ex 3: 

This lab is to let us learn more about BST. By doing this lab, let me have more understanding of the BST and will do better in the future.