This is the second lecture on the topic of binary tree.

This lecture focuses on creating binary search tree. How can we create a binary search tree like this? How is the insert function written?

First, let us go through the concept then we will read the code.

If we start from nothing, the first inserted number will be the root of the tree.

The next number is 9. It is the left child of 27 because 9 is smaller than 27.

The next number is 38. It is greater than 27 so 38 is the right child of 27.

The next number is 15. 15 is smaller than 27. Thus, it should be at the left side of the root. Then, we compare 9 with 15. 15 is larger and it becomes the right child of 9.

Next, we insert 3. This number is smaller than 27 and it is inserted to the left side of the root. 3 is also smaller than 9. 3 becomes the left child of 9.

The next number is 46. It is greater than 27 and 38. It becomes the right child of 38.

The next number is 32. It goes to the right side of 27 and the left side of 38.

The next number is 6. It goes to the left side of 27, the left side of 9, and the right side of 3.

The next number is 29. It goes to the right side of 27, the left side of 38, and the left side of 32.

The last number is 36. It goes to the right side of 27, the left side of 38, and the right side of 32.

The shape of a binary search tree depends on the order of the inserted numbers. If the inserted numbers are 2, 3, 1. The root is 2. 3 is on the right side. 1 is on the left side.

If the order is 2, 1, 3, the tree is the same.

If the inserted numbers are 3, 2, 1, then 3 is the root. 2 is the left child of 3. 1 is the left child of 2.

If the order is 1, 2, 3, the root is 1. 2 is the right child of 1. 3 is the right child of 2.

Now, let’s study how to write the insert function. We need to study two functions. The first is the node construction function. It has one argument of integer. The function allocates memory for a tree node. Both left and right children are initialized to NULL. This is important because we test whether a pointer is NULL to decide whether it points to a valid child. The node’s value is assigned to the value from the argument.

The function returns the address of the allocated memory.

This is the insert function. The insert function takes two arguments. The first is a tree node and the second is an integer.

At the top of this function, it checks whether the tree is empty. If the tree is empty, it is NULL. The function calls the construct function and return the memory returned by the construction function.

If the tree is not empty, the insert function compares the value to be inserted with this node’s value. If the values are the same, the function returns this node because we do not want to store the same value in the tree twice.

If the values are different, then we consider whether the inserted value is smaller or larger. If the value is smaller, the value is inserted to the left side.

If the value is greater, the value is inserted to the right side.

How do we call this insert function?

First, we create a tree node pointer and initialize it to NULL. It is absolutely critical that this pointer is initialized to NULL. If it is not initialized to NULL, the program’s behavior is undefined.

To insert a node, put the root as the first argument. The value to be inserted is the second argument. The function’s return value is stored in the root.

How does this work? Let’s see what happens to the stack and heap memory.

Let’s consider the beginning. We create a pointer called root and it is initialized to NULL. This is shown in the stack memory.

The program calls the insert function. The value address is 100 because it is the address of root in the main function.

The insert function has two arguments. The first is a pointer and the second is an integer value.

When inserting the first node, the first argument’s value is NULL because it is copied from the root’s value.

The second argument is 27.

Inside the insert function, it first checks whether Tee N is NULL or not. If it is NULL, it calls the construct function and return the address from the construct function.

The construct function allocates memory, sets the left and the right pointers to NULL, and assigns the value.

The construct function returns the address of the allocated memory.

This address is written to the value at address 100 because 100 is the value address when calling the insert function.

After inserting the first value, the value of root is the address returned by the construct function.

Next, we want to insert another number. Please notice that the first argument is root. The second argument is 9.

The value of root is address 70000. This is the address returned by calling the construct function earlier.

The address 70000 is copied to the value of the firs argument of the insert function.

The second argument of the insert function is 9.

Because tee N is not NULL, the first if condition is not satisfied.

The program moves to the next if condition. It compares the second argument, Vee ei L, with Tee N arrow value. The value of Tee N is the address 70000. Tee N arrow value is 27. This was the first value inserted into the tree.

Vee ei L is the second argument and it is 9. 9 and 27 are different. Thus, the second if condition is not satisfied.

The function moves to the third if condition. Vee ei L is 9. Tee N arrow value is 27. This condition is satisfied.

The program enters this line calling the insert function again.

Let’s analyze this line carefully. The first argument is Tee N arrow left. This reads the value at the address 70000. The value is NULL. Thus, the first argument is NULL.

The left side of this assignment is Tee N arrow left. What is the address of this? It is at the address 70000. Thus, the value address is 70000. Please notice that this is Tee N arrow left. It is not Tee N. .

If we had put Tee N at the left side of this assignment, the value address would be 300.

The second call of the insert function calls the construct function. The first argument is NULL. Thus, the insert function calls the construct function. The construct function allocates memory, assigns 9 to the value attribute, and assigns NULL to the left and the right pointers.

Suppose the construct function returns memory address 80000. The address 80000 is written to the value at address 70000.

The insert function returns Tee N. .

Tee N is the address of the first node of the tree. In other words, Tee N returns 70000.

This value is written back to the root in the main function.

After the insert function returns, this is the stack memory and the heap memory.

Root’s value is 70000. It points to the memory address of 70000. The memory at 70000 has three attributes: left, right, and value.

The left attribute stores the memory address of 80000. The right attribute stores NULL. The value attribute stores integer 27.

The memory at address 80000 also has three attributes. The left attribute and the right attribute stores NULL. The value attribute stores integer 9.

This slide illustrates the tree in the middle.

Next, we insert number 38 into the tree.

For the insert function, the first argument is the value of Tee N. It is address 70000. The second argument stores value 38.

The first if condition tests whether Tee N is NULL. Tee N stores the address 70000. Thus, Tee N is not NULL.

This if condition is not satisfied.

The second if condition compares Vee ei L and Tee N arrow value. Vee ei L is 38. Tee N arrow value is 27. They are different. Thus, this if condition is not satisfied.

The next if condition compares the two values again. This is still not satisfied.

Thus, the program enters the else part calling the insert function again.

The first argument is Tee N arrow right. It is the right attribute of the node. This attribute is NULL. Thus, when the insert function is called the second time, the first argument is NULL.

The second value is vee ei L and its value is 38.

The left side of the assignment is Tee N arrow right. The address is 70008. Thus, the returned value will be written to the address 70008.

The insert function calls the construct function and allocates memory. Suppose the address of the newly allocated memory starts at address 90000. The left and right attributes are both NULL. The value is 38.

After the insert function returns, this is the stack memory and the heap memory.

As you can see the root points to memory address 70000. The left attribute points to address 80000. The right attribute points to address 90000.

We have successfully inserted three nodes into the binary search tree.