Recursive Algorithm for Tree Operation

1. Searching

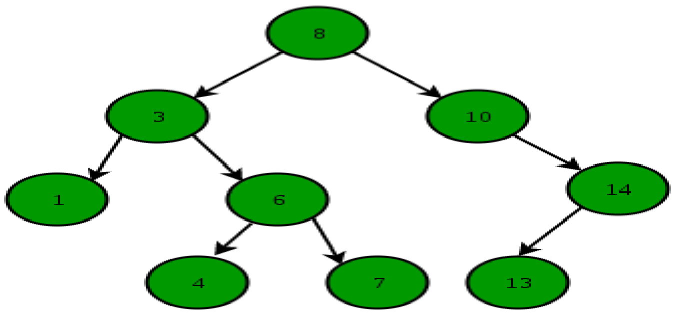
To search a given key in Binary Search Tree, we first compare it with root, if the key is present at root, we return root. If key is greater than root's key, we recur for right subtree of root node. If key is lesser than root's key, we recur for left subtree of root node.

C function to search a given key in a given BST

```
struct node* search(struct node* root, int key)
{

   if (root == NULL || root->key == key)
      return root;

  else if (key> root->key)
      return search(root->right, key);

    else
   return search(root->left, key);
}
```

Illustration to search 6 in below tree:

1. Start from root.
2. Compare the inserting element with root, if less than root, then recurse for left, else recurse for right.
3. If element to search is found anywhere, return true, else return false.



2. Insertion:

If the tree is empty, we return a new node containing the key and value; if the search key is less than the key at the root, we set the left link to the result of inserting the key into the left subtree; otherwise, we set the right link to the result of inserting the key into the right subtree.

A new key is always inserted at leaf. We start searching a key from root till we hit a leaf node. Once a leaf node is found, the new node is added as a child of the leaf node.

C function to insert a new node with given key in Binary search tree

```
struct node* insert (struct node* node, int key)
{

   if (node == NULL)
    return newNode(key);



   if (key < node->key)
      node->left = insert(node->left, key);

   else if (key > node->key)
      node->right = insert(node->right, key);

   return node;
}
```