To illustrate

Suppose we wish to search this BST for 60.

We begin at the root since 60 is greater than root 50 so we will go to the right side of the tree, left (skipping left subtree)

Because we know that the desired value if present in the binary tree then it must be located to the right of the root. that is it must be in the right subtree.

Now we will ~~traverse~~ search right subtree again compare 60 with root 80. We observe that 60 is less than the 80 we will go to left side to search in left subtree

Finally we finds the desired value at one node left subtree.

Now we will develope an algorithm to search (using binary search) in the binary search tree.

Suppose following is the declaration mode to define a tree in pascal.

```
TYPE
    ElementType = . . ;
    TreePointer = ^TreeNode
    TreeNode = RECORD
        Data : ElementType;
        Lchild,
        Rchild : TreePointer;
    END;
VAR
    Root : TreePointer;
```

We will take one pointer (say) RootPointer initially pointing to the root of BST.

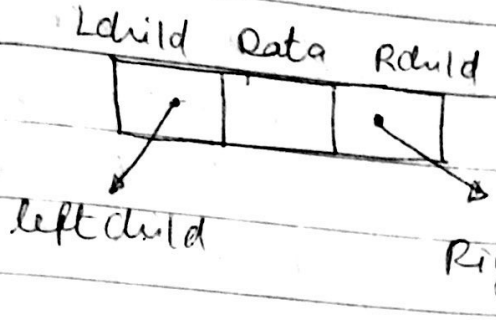
Then it will be replaced left or Right link of the current Node depending on the result of comparison of search value with the root value.

```
Procedure BSTSearch (Rootptr: Treepointer;
Item: ElementType; var Found: boolean;
var Curptr: Treepointer);

(* Procedure to search BST with
Root pointed to by Rootptr.
Found is returned as true
if Curptr points to a node
containing Item if the search is
successful; otherwise, Found is
returned as False *)
Begin
    Curptr := Rootptr;
    Found := False;

    While Not Found and (Curptr <> Nil)
                        do
    with Curptr^ do
    If Item < Data Then (* search left *)
        Curptr := LChild
    else
        If Item > Data Then
            Curptr := RChild (* search Right *)
        else
            Found := True;
                    (* Item Found *)
End;
```

Lchild and Rchild are pointers to nodes representing its left and right children, respectively.



B without WITH :-

```
Begin
    Curptr := Rootptr;
    Found := false;
While not Found and Curptr <> Nil) do
Begen
    If Item < Curptr^.Data Then
        Curptr := Curptr^.Lchild
    else
        If Item > Curptr^.Data Then
            Curptr := Curptr^.Rchild
        else
            Found := True;
End;
```

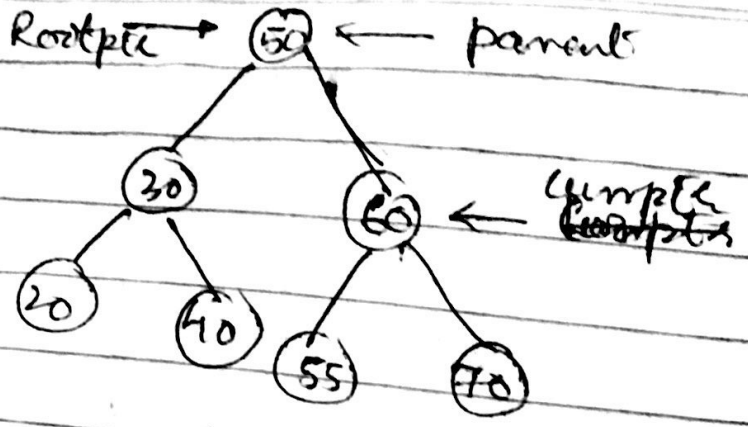


Fig. (2)

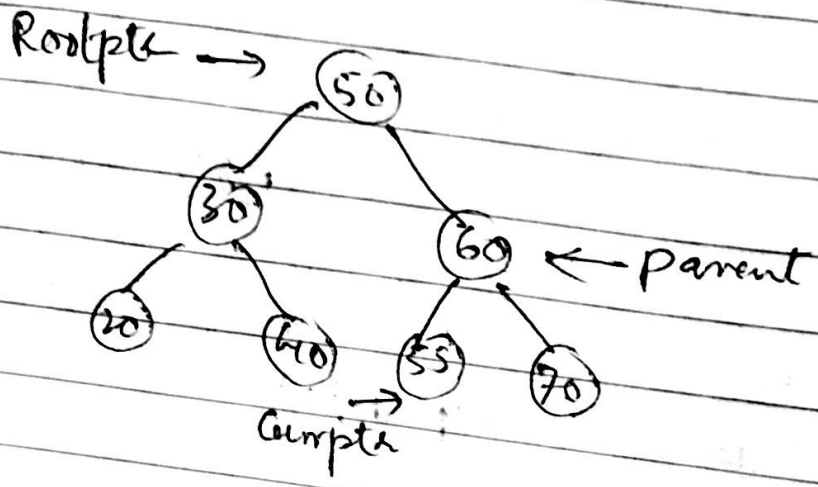


Fig (3)

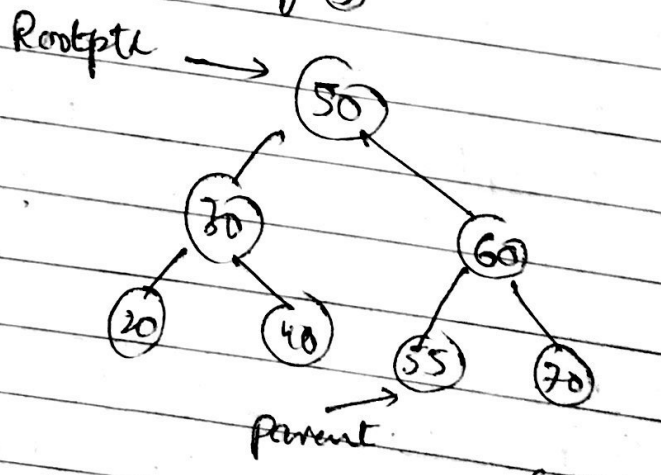
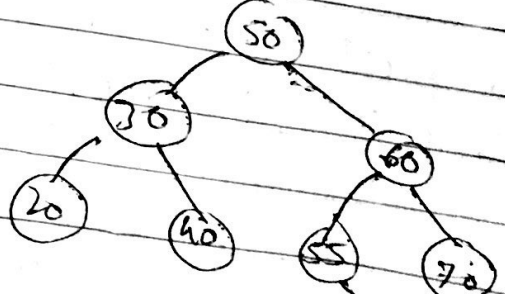


Fig. (4)



The procedure that we are going to develop now make use of three pointer variables.

⇒ Rootptr → points to the root

⇒ currptr → Initially points to the root and will go left. or right depending on the result of comparison of element to be inserted with roots value

⇒ parent ⇒ this will be nil initially and these will chase (trail) the currptr (current pointer) to keep track of the parent node so that the new node can be attached to the BST in the proper place.