**The binary search tree traversal**

There are various strategies which can be employed to traverse the items in a tree; the choice of strategy depends on which node visitation order you require. In this section we will touch on the traversals that DSA provides on all data structures that derive from *BinarySearchT ree*.

**Preorder**

When using the preorder algorithm, you visit the root first, then traverse the left subtree and finally traverse the right subtree.

1) algorithm Preorder(*root*)

2) Pre: *root* is the root node of the BST

3) Post: the nodes in the BST have been visited in preorder

4) if $root\ne ∅$

5) yield *root*.Value

6) Preorder(*root*.Left)

7) Preorder(*root*.Right)

8) end if

9) end Preorder

**Postorder**

The value of the node is yielded after traversing both subtrees. An example of postorder

traversal:

1) algorithm Postorder(*root*)

2) Pre: *root* is the root node of the BST

3) Post: the nodes in the BST have been visited in postorder

4) if$root \ne ∅$*;*

5) Postorder(*root*.Left)

6) Postorder(*root*.Right)

7) yield *root*.Value

8) end if

9) end Postorder

**Inorder**

Another variation of the algorithms that of inorder traversal where the value of the current node is yielded in between traversing the left subtree and the right subtree.

1) algorithm Inorder(*root*)

2) Pre: *root* is the root node of the BST

3) Post: the nodes in the BST have been visited in inorder

4) if *root 6*= *;*

5) Inorder(*root*.Left)

6) yield *root*.Value

7) Inorder(*root*.Right)

8) end if

9) end Inorder





