**Threaded Binary Trees Algorithm**

struct Node

{

struct Node \*left, \*right;

int info;

// True if left pointer points to predecessor

// in Inorder Traversal

boolean lthread;

// True if right pointer points to successor

// in Inorder Traversal

boolean rthread;

};

**Case 1: Insertion in empty tree**  
Both left and right pointers of tmp will be set to NULL and new node becomes the root.

root = tmp;

tmp -> left = NULL;

tmp -> right = NULL;

**Case 2: When new node inserted as the left child**  
After inserting the node at its proper place we have to make its left and right threads points to inorder predecessor and successor respectively. The node which was inorder successor. So the left and right threads of the new node will be:

tmp -> left = par ->left;

tmp -> right = par;

Before insertion, the left pointer of parent was a thread, but after insertion it will be a link pointing to the new node.

par -> lthread = false;

par -> left = tmp;

**Case 3: When new node is inserted as the right child**  
The parent of tmp is its inorder predecessor. The node which was inorder successor of the parent is now the inorder successor of this node tmp. So the left and right threads of the new node will be:

tmp -> left = par;

tmp -> right = par -> right;

Before insertion, the right pointer of parent was a thread, but after insertion it will be a link pointing to the new node.

par -> rthread = false;

par -> right = tmp;

**InOrder Traversal**

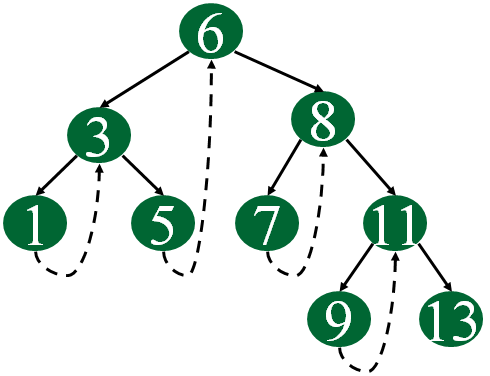
Inorder traversal of a binary tree can either be done using recursion or with the use of an auxiliary stack. The idea of threaded binary trees is to make inorder traversal faster and do it without stack and without recursion. A binary tree is made threaded by making all right child pointers that would normally be NULL point to the inorder successor of the node (if it exists).

There are two types of threaded binary trees.  
***Single Threaded:*** Where a NULL right pointers is made to point to the inorder successor (if successor exists)

***Double Threaded:*** Where both left and right NULL pointers are made to point to inorder predecessor and inorder successor respectively. The predecessor threads are useful for reverse inorder traversal and postorder traversal.

The threads are also useful for fast accessing ancestors of a node.

Following diagram shows an example Single Threaded Binary Tree. The dotted lines represent threads.



The following diagram demonstrates inorder order traversal using threads.

