DATA STRUCTURES AND ITS APPLICATIONS

Shylaja S S & Kusuma K V
Department of Computer Science
& Engineering

DATA STRUCTURES AND ITS APPLICATIONS

Threaded BST and its Implementation

Shylaja S S
Department of Computer Science & Engineering

Threaded Binary Search Tree

Motivation

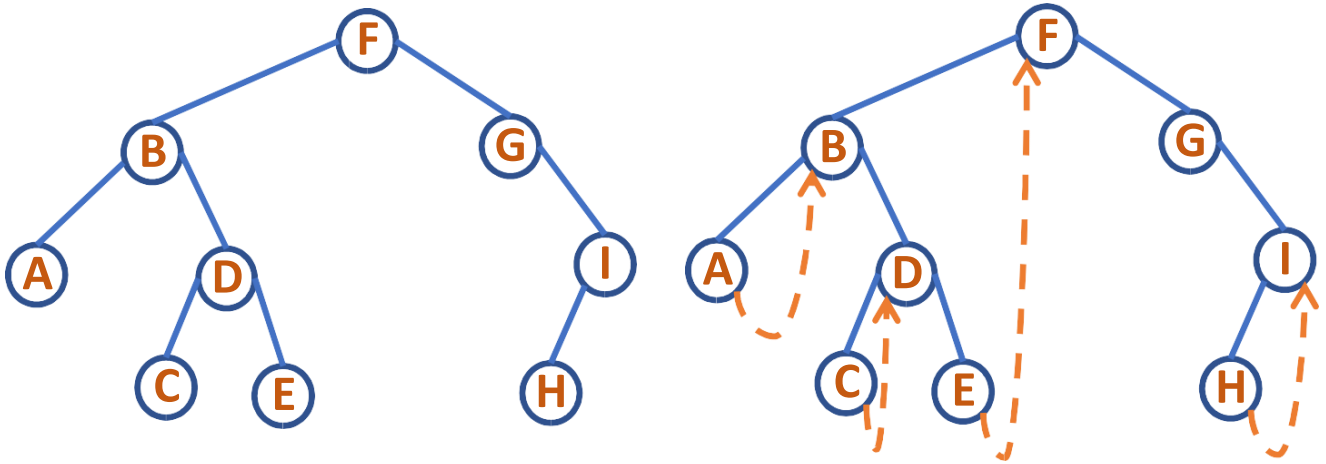
- Iterative Inorder Traversal requires Explicit stack
- Costly
- Since we loose track of address as and when we navigate, Node addresses were stacked
- If this can be achieved through some other less expensive mechanism, we can eliminate the use of explicit stack
- Small structural modification carried on Binary tree will solve the above problem

Threaded Binary Search Tree

- We can use the right pointer of a node to point to the inorder successor if in case it is not pointing to the child. Such a tree is called **Right-In Threaded** Binary Tree
- If we use the left pointer to store the inorder predecessor, the tree is called **Left-In Threaded** Binary Tree
- If we use both the pointers, the tree is called **In Threaded** Binary Tree

Threaded Binary Search Tree

Right-In Threaded Binary Tree



Binary Tree

Right-In Threaded Binary Tree

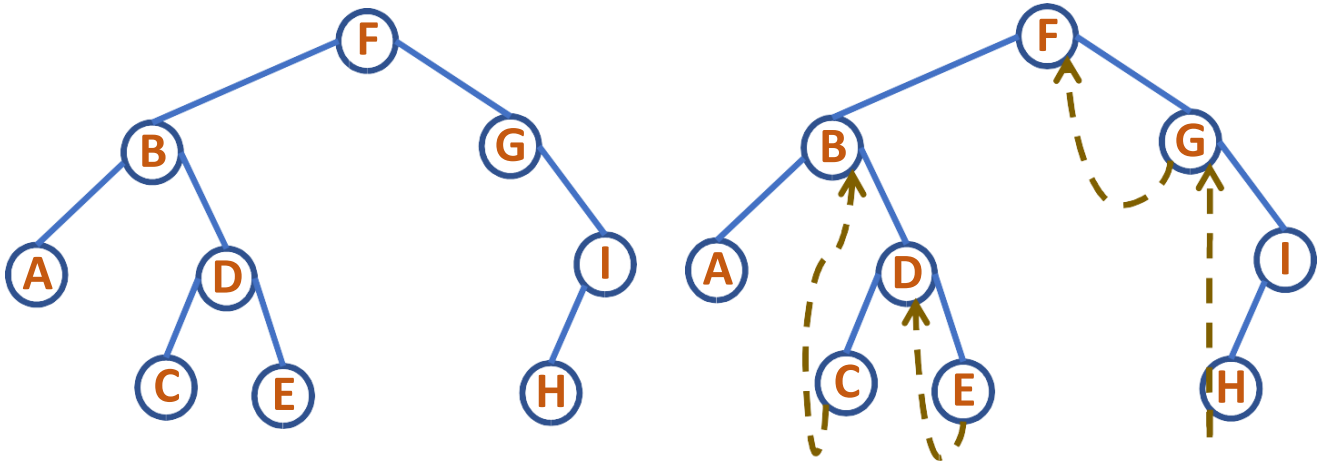
Inorder Traversal:

A B C D E F G H I

Nodes with Right Pointer NULL	A	C	E	H	I
Inorder Successor	B	D	F	I	-

Threaded Binary Search Tree

Left-In Threaded Binary Tree



Binary Tree

Left-In Threaded Binary Tree

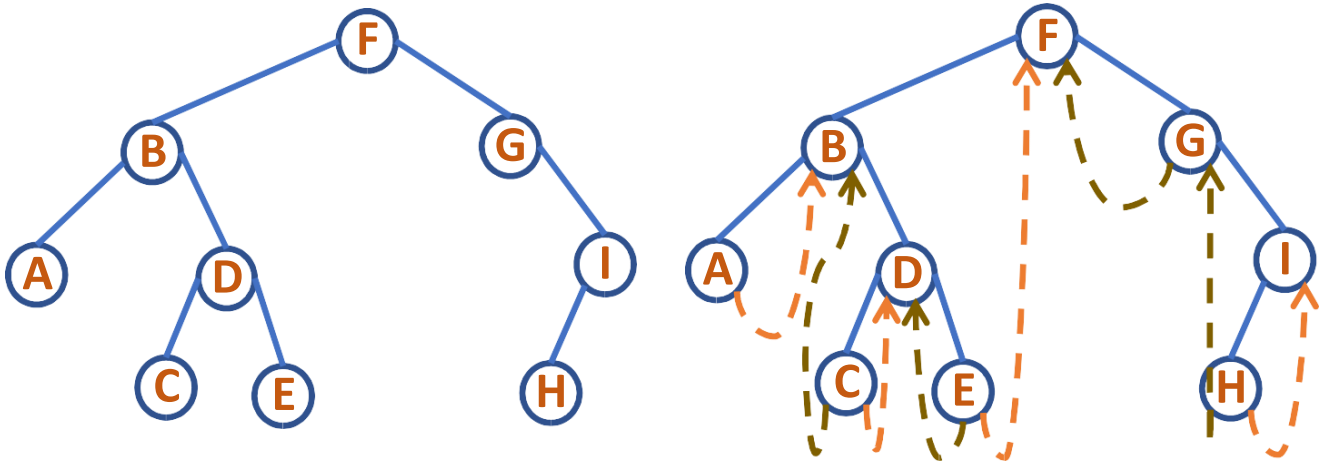
Inorder Traversal:

A B C D E F G H I

Nodes with Left Pointer NULL	A	C	E	G	H
Inorder Predecessor	-	B	D	F	G

Threaded Binary Search Tree

In Threaded Binary Tree



Binary Tree

In Threaded Binary Tree

Threaded Binary Search Tree: Implementation

Right In Threaded Binary Tree

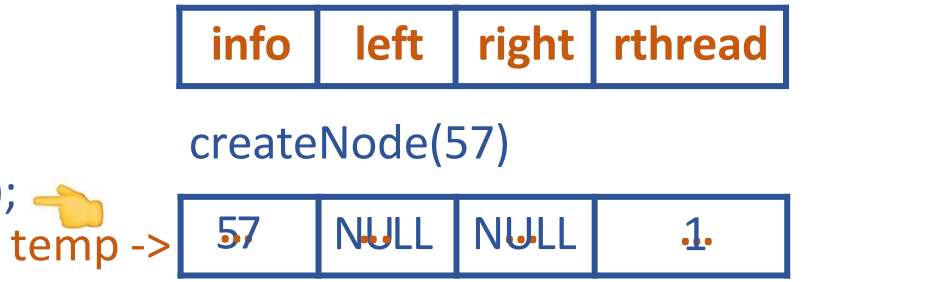
```
typedef struct node
{
        int info;
        struct node *left;      // pointer to left child
        struct node *right;     // pointer to right child
        int rthread;            // rthread is TRUE if right is NULL
                                // or a non-NULL thread
}NODE;
```

Node Structure

DATA STRUCTURES AND ITS APPLICATIONS

Threaded Binary Search Tree: Implementation

```
NODE* createNode(int e)
{
        NODE* temp=malloc(sizeof(NODE));
        temp->info=e;
        temp->left=NULL;
        temp->right=NULL;
        temp->rthread=1;
        return temp;          // Returns: 2000
}
```



Threaded Binary Search Tree: Implementation

Right In Threaded Binary Tree: 57, 25, 28

- A node is created with rthread set to TRUE
- **insert 57**

Address: 800

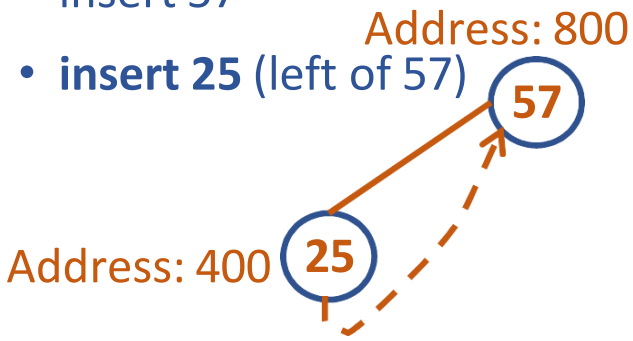
Node Structure

info	left	right	rthread
57	NULL	NULL	1

Threaded Binary Search Tree: Implementation

Right In Threaded Binary Tree: 57, 25, 28

- A node is created with rthread set to TRUE
- insert 57
- **insert 25** (left of 57)



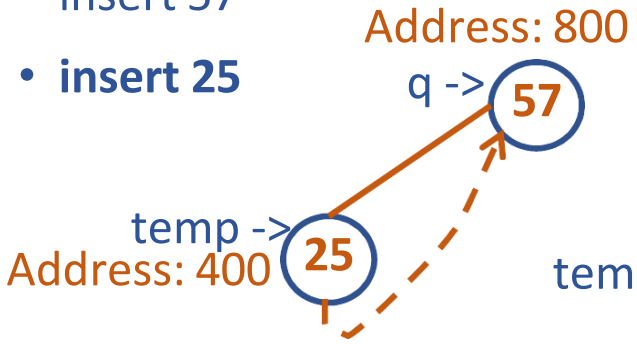
Node Structure

info	left	right	rthread
57	400	NULL	1
25	NULL	800	1

Threaded Binary Search Tree: Implementation

Right In Threaded Binary Tree: 57, 25, 28

- A node is created with rthread set to TRUE
- insert 57
- **insert 25**



Node Structure

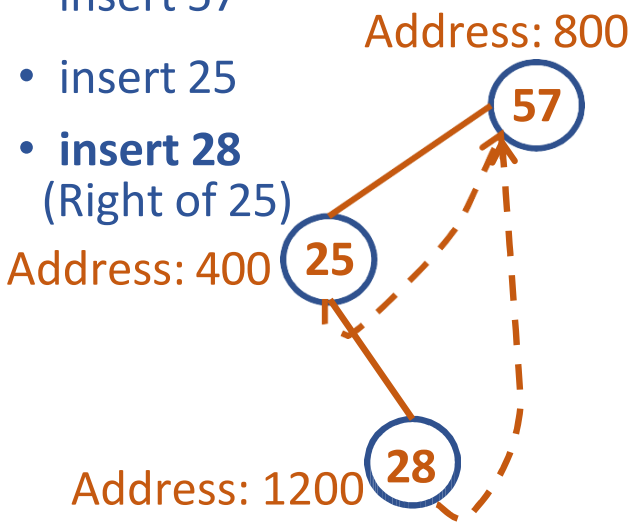
	info	left	right	rthread
q ->	57	400	NULL	1
temp ->	25	NULL	800	1

```
void setLeft(NODE* q, int e) {      //set node with info e to left of q
        NODE* temp=createNode(e);           //e=25
        q->left=temp;
        temp->right=q;
}
```

Threaded Binary Search Tree: Implementation

Right In Threaded Binary Tree: 57, 25, 28

- A node is created with rthread set to TRUE
- insert 57
- insert 25
- **insert 28**
  (Right of 25)



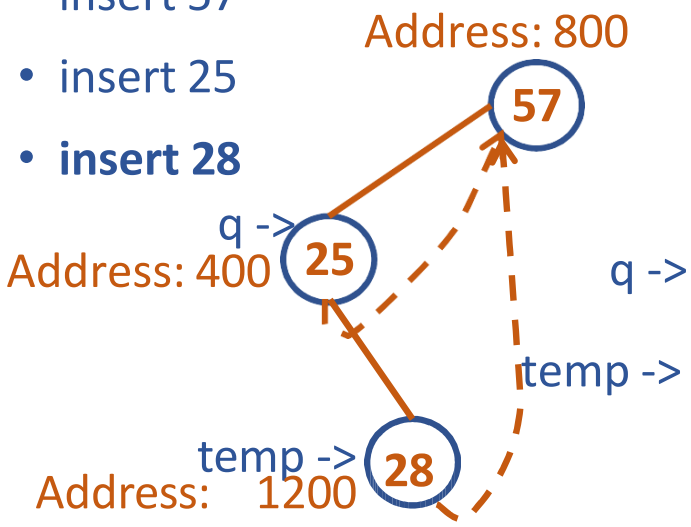
Node Structure

info	left	right	rthread
57	400	NULL	1
25	NULL	1200	0
28	NULL	800	1

Threaded Binary Search Tree: Implementation

Right In Threaded Binary Tree: 57, 25, 28

- A node is created with rthread set to TRUE
- insert 57
- insert 25
- **insert 28**



Node Structure

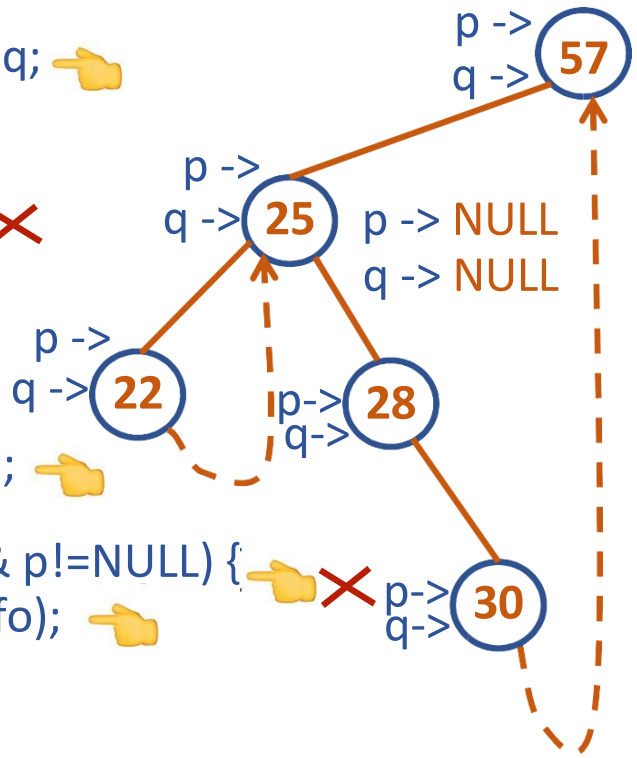
info	left	right	rthread
57	400	NULL	1
25	NULL	1200	0
28	NULL	800	1

q ->

```
void setRight(NODE* q,int e) {
   NODE* temp=createNode(e);
temp->right=q->right;
q->right=temp;
   >rthread=0;
```

Threaded Binary Search Tree: Inorder Traversal

```
void inOrder(NODE *root) {
   NODE *p=root;   NODE *q;
   do{
        q=NULL;
        while(p!=NULL) {
           q=p;
           p=p->left;
        }
        if(q!=NULL) {
           printf("%d ",q->info);
           p=q->right;
           while(q->rthread && p!=NULL) {
              printf("%d ",p->info);
              q=p;
              p=p->right;
           }
        }
   }while(q!=NULL);
}
```



q ->NULL

rthread is TRUE for nodes with info: 22, 30, 57

Inorder Traversal:

22 25 28 30 57

1. What is the main advantage of a threaded BST?

A) Faster insertion than a normal BST

B) Efficient inorder traversal without recursion or stack

C) Requires less memory than normal BST

D) Allows multiple keys per node

1. What is the main advantage of a threaded BST?

A) Faster insertion than a normal BST

B) Efficient inorder traversal without recursion or stack

C) Requires less memory than normal BST

D) Allows multiple keys per node

2. In a right-threaded BST, which pointers are replaced by threads?

A) Only left child null pointers
B) Only right child null pointers
C) Both left and right child null pointers
D) Parent pointers

2. In a right-threaded BST, which pointers are replaced by threads?

A) Only left child null pointers

B) Only right child null pointers

C) Both left and right child null pointers

D) Parent pointers

3. Which of the following is a limitation of a threaded BST?

A) Cannot perform preorder traversal
B) Insertion is more complex than normal BST
C) It cannot store duplicate values
D) Traversal requires recursion

3. Which of the following is a limitation of a threaded BST?

A) Cannot perform preorder traversal

B) Insertion is more complex than normal BST

C) It cannot store duplicate values

D) Traversal requires recursion

4. In a threaded BST, if a node has a thread instead of a right child, following that pointer leads to:

A) Its left child
B) Its right child
C) Its inorder successor
D) Its parent

4. In a threaded BST, if a node has a thread instead of a right child, following that pointer leads to:

A) Its left child

B) Its right child

C) Its inorder successor

D) Its parent

5. Which of the following statements is true about a threaded BST?

A) It eliminates recursion completely for all types of traversals

B) It eliminates the need for a stack during inorder traversal

C) It has no null pointers at all

D) It is always a balanced BST

5. Which of the following statements is true about a threaded BST?

A) It eliminates recursion completely for all types of traversals

B) It eliminates the need for a stack during inorder traversal

C) It has no null pointers at all

D) It is always a balanced BST

THANK YOU

Shylaja S S

Department of Computer Science
& Engineering

shylaja.sharath@pes.edu