Q) Write a function to traverse threaded binary tree in pre-order ?

{page 278 , question 4}

K.A.S.Sekhar ***160114733091*** ***CSE-2 BE(2/4)***

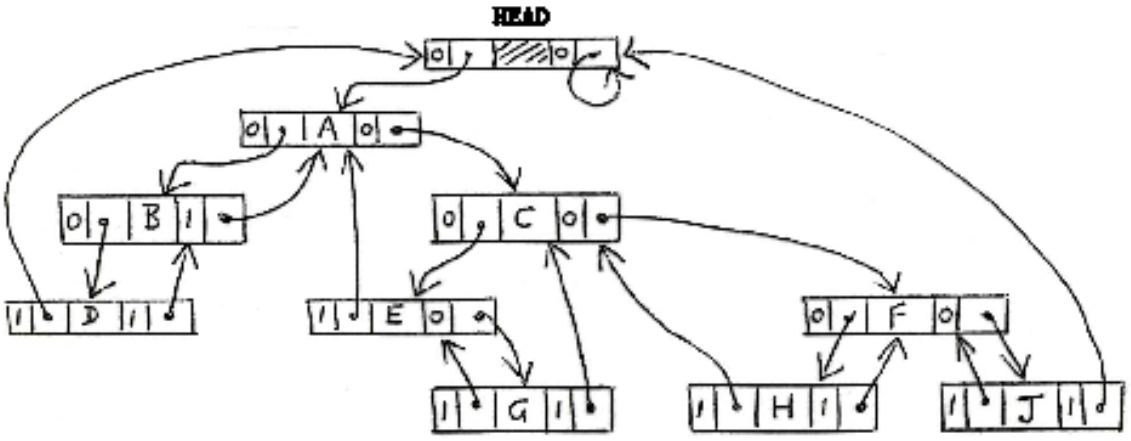
Let us consider the following Threaded Binary Tree. A node of threaded binary tree always contains 5 fields.

1) Data
2) Left child (lch)
3) Right child (rch)
4) Left thread (lth)
5) Right thread (rth)

Left and right child are pointers of type node itself.

Data can be anything, integer, char etc.

Left and right thread are Boolean data types.(0 is false , 1 is true)



Initially there are is a head node, whose data is field is garbage , both thread values are false. The right child(rch) pointer points towards itself, the left child (lch) pointer points towards the first child.

IF,

The value of lth is true ,then lch points towards the inorder predecessor.

The value of rth is true ,then rch points towards the inorder successor.

The value of lth is false, then lch points towards left child.

The value of rth is false , then rch points towards right child.

The question asked is to traverse the tree in pre-order format,

That is DATA,LEFT,RIGHT.

CODE:-

```
template<class T>
void TBTree<T>::preorder(TBTnode<T> *p)
{
     TBTnode<T> *temp=p;
     while(1)
     {
          temp=preSucc(temp);
          if(temp==p)
          break;
          cout<<"\t"<<temp->data;
     }
}
```

```
template<class T>
TBTnode<T> * TBTree<T>::preSucc(TBTnode<T> *p)
{
     if(p->lth==false)
     return p->lch;
     else
     {
          while(p->rth==true)
          p=p->rch;
          return p->rch;
     }
}
```

EXPLANATION:-

There are two functions , one named preorder ,which produces the output if the node address is not the node address of HEAD node (since head node has no data). Once the HEAD node address is obtained , the function uses a break statement and comes out of the loop.

The second function is preSucc (preorder successor) ,it generates the preorder successor of a given node and returns its address to the preorder function . This function uses the inorder linking (or threading) already present in the Threaded Binary Tree and gives us the preoder successor .

Here , TBTnode is a class which is nothing but the node of the threaded binary tree.

```
template<class T>

class TBTnode

{

     T data;

     TBTnode<T> *lc,*rc;

     bool lth,rth;

     public:

          TBTnode(T d,TBTnode<T> *l,TBTnode<T> *r);
          friend class TBTree<T>;

};
```

It is a template class since the data can be of any type.

TBTree is the class which contains the Head node and the functions and also the above two functions to manipulate the tree.

If we apply these functions to the above given tree , that is , if we call the preorder function using the address of head node :-

We get the following **RESULT:-**

A B D C E G F H J

COMPLEXITY:-

If we talk about the complexity of the functions, then it is nothing but the complexity of the preSucc function.

Its same as the complexity of the function that generates the inorder successor. That is O(n) for an n-node tree.