Threaded Binary Tree

Dec. 11,2021, L-20

Threaded Binary Tree

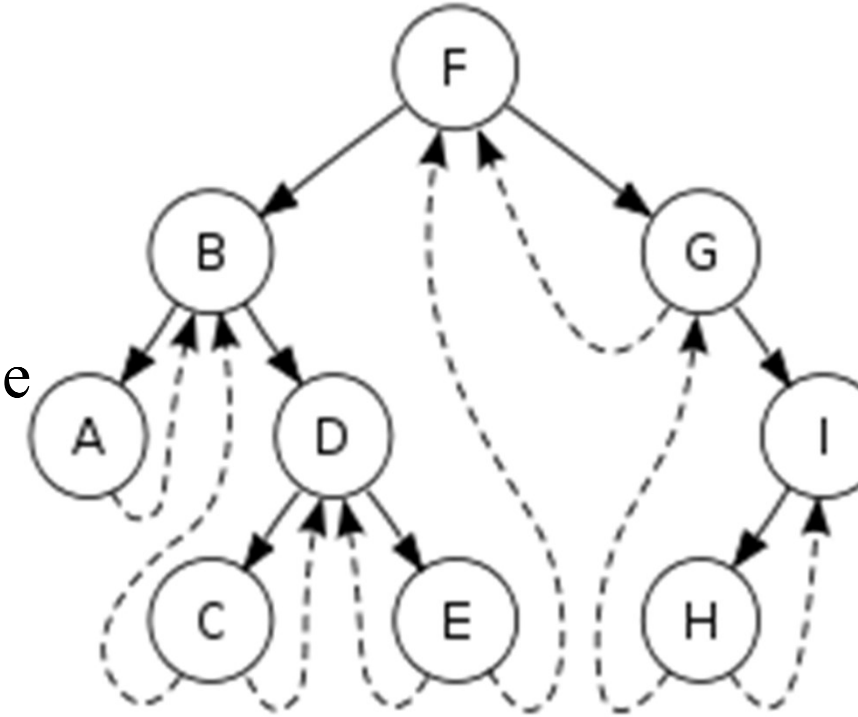
- To efficiently use the Null links
- Replace all the null links by pointers to other nodes of the tree and are referred as threads
- Rules to construct the Threaded binary tree

✓

- all right child pointers (ptr->rchild) that would normally be null point to the inorder successor of the node (if it exists)
- all left child pointers (ptr->lchild) that would normally be null point to the inorder predecessor of the node(if it exists)

Threaded Tree Example

inorder traversal : ABCDE FGHI



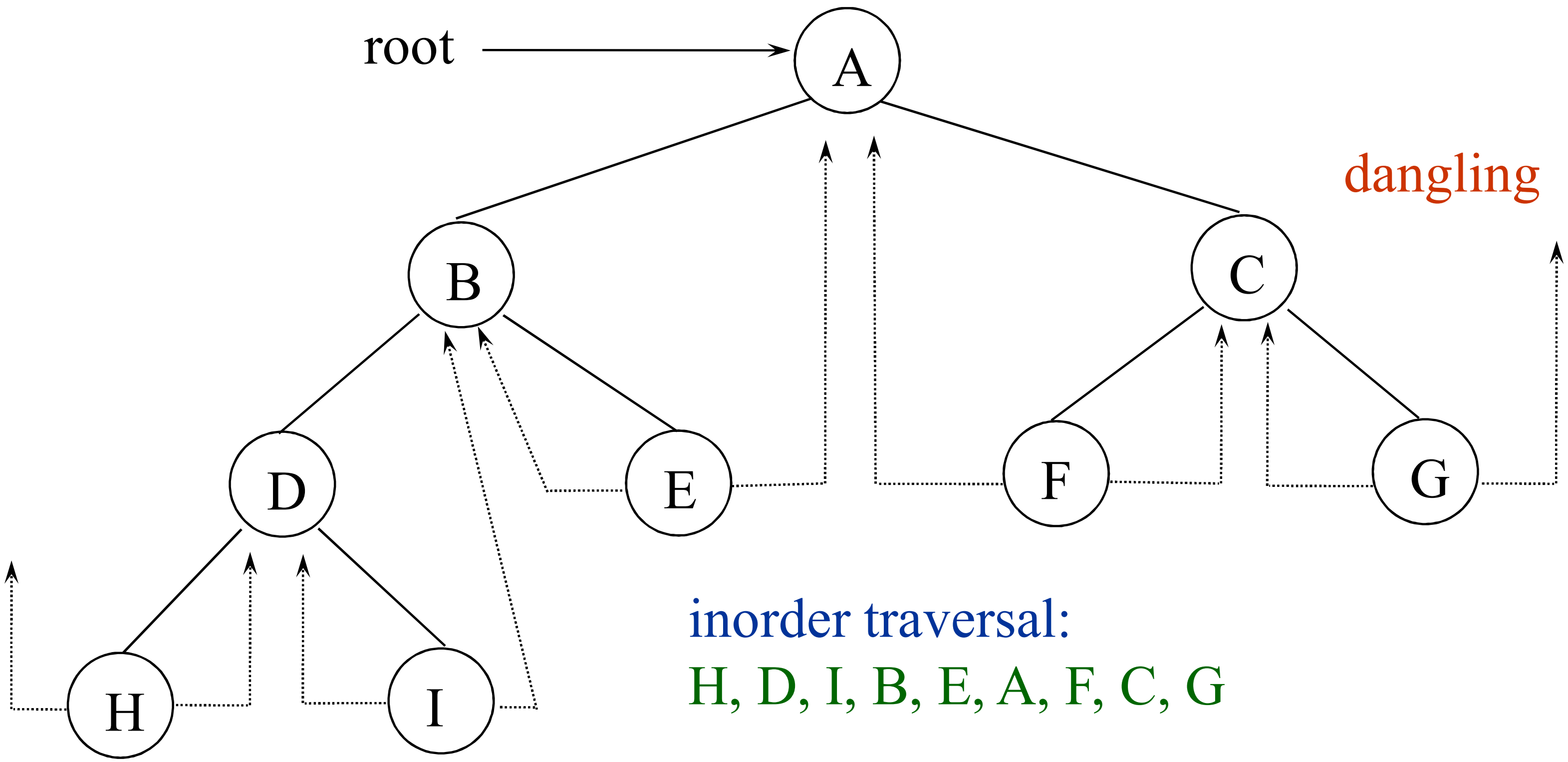
If `ptr->left_child` is null,

replace it with a pointer to the node that would be visited *before* `ptr` in an *inorder traversal (onorder predecessor)*

If `ptr->right_child` is null,

replace it with a pointer to the node that would be visited *after* `ptr` in an *inorder traversal (inorder successor)*

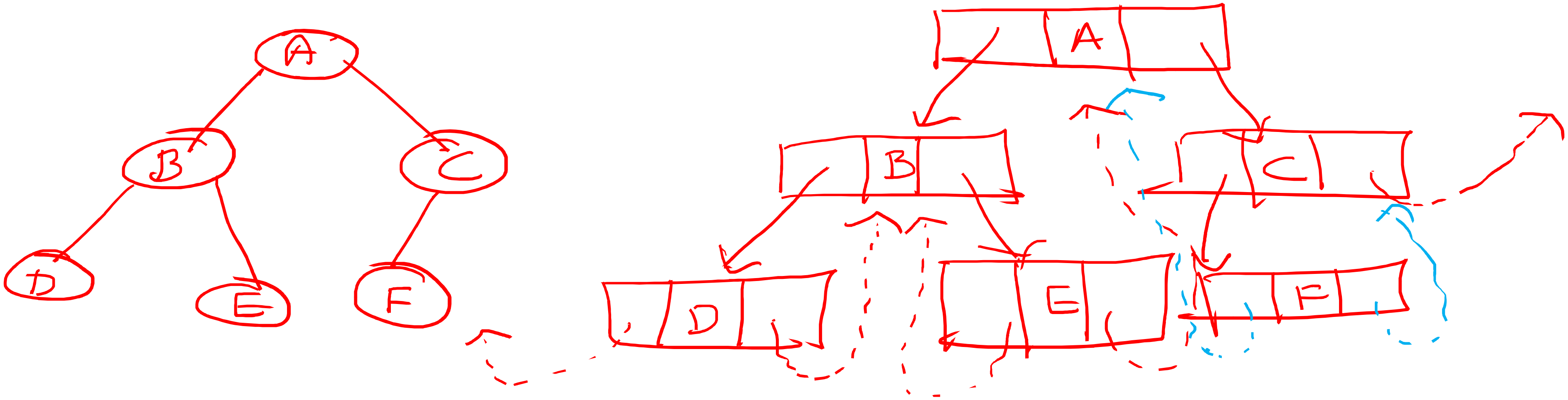
Example:



inorder traversal:

H, D, I, B, E, A, F, C, G

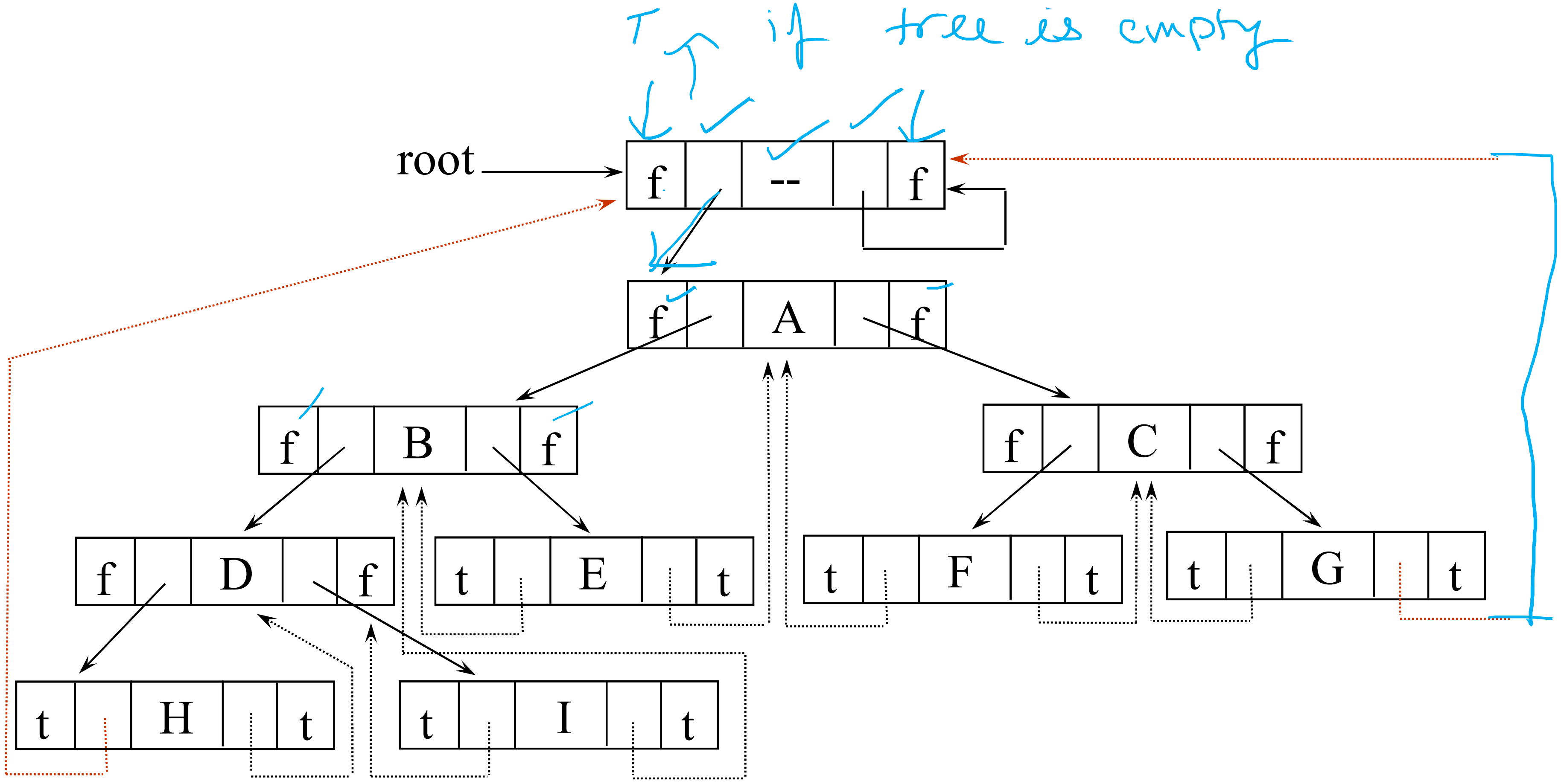
Threaded binary tree ~~with header node~~



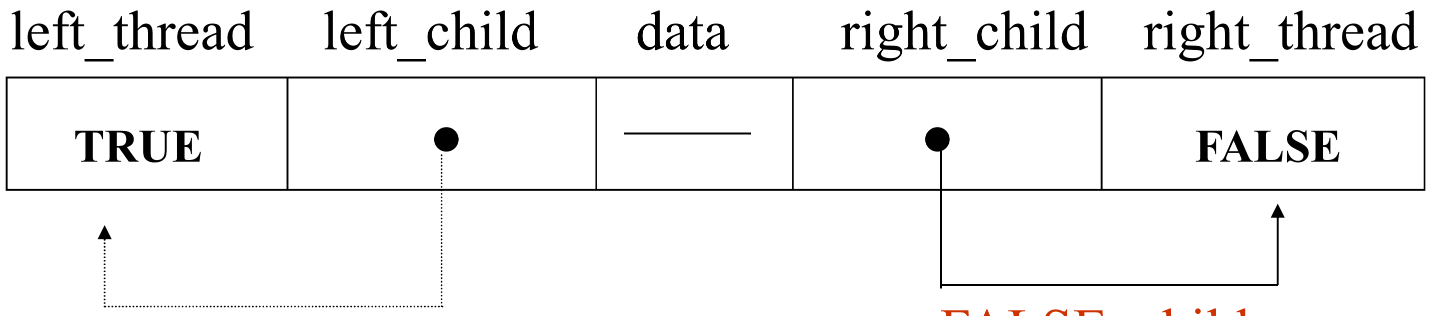
D B E A F C

Memory Representation of A Threaded BT

T if tree is empty



Data Structures for Threaded BT



TRUE: thread

FALSE: child

↳ ~~Tree~~ True is empty

```
class Thtree {
    short int left_thread;
    Thtree left_child;
    char data;
 Thtree right_child;
    short int right_thread; };
```

Comparison of Threaded BT

Threaded Binary Trees

- In threaded binary trees, The null pointers are used as thread.
- We can use the null pointers which is a efficient way to use computers memory.
- Traversal is easy. Completed without using stack or reccursive function.
- Structure is complex.
- Insertion and deletion takes more time.

Normal Binary Trees

- In a normal binary trees, the null pointers remains null.
- We can't use null pointers so it is a wastage of memory.
- Traverse is not easy and not memory efficient.
- Less complex than Threaded binary tree.
- Less Time consuming than Threaded Binary tree.