

The seal of Hanyang University is a circular emblem. It features a central shield-like shape with the Korean characters '한양' (Hanyang) inside. The words 'HANYANG UNIVERSITY' are written in a circular path around the center. At the bottom of the seal, the year '1939' is inscribed. The seal is rendered in a light blue, semi-transparent style.

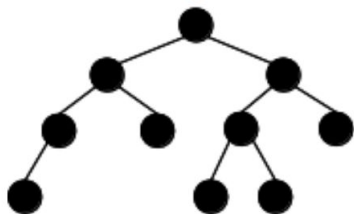
Lab 04: Tree Traversal

Data Structure 2023

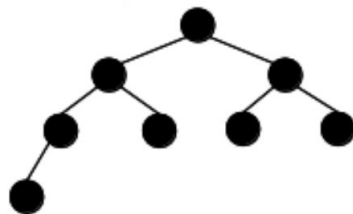
Binary Tree

- **Full binary tree** is a binary tree in which every node has 0 or 2 children
- **Complete binary tree** is a binary tree in which every level, except the last, is completely filled and the last level has all its nodes to the left side

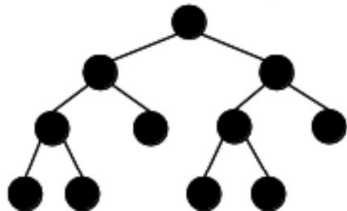
Neither complete nor full



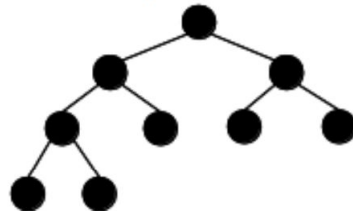
Complete but not full



Full but not complete

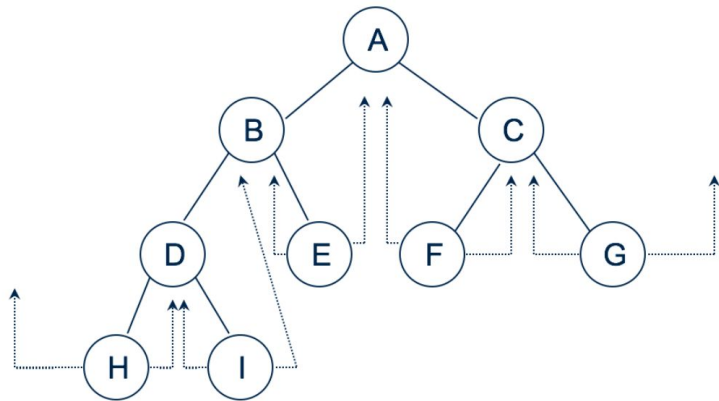


Complete and full



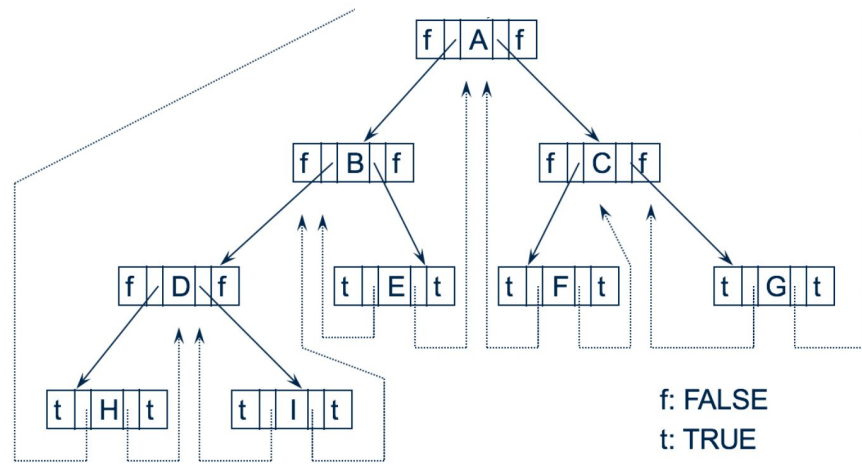
Threaded Binary Tree

- There are **$n+1$ null links** out of **$2n$ total links**
- Replace the null links by pointers, called **threads** to other nodes in the tree
 - if **ptr -> leftChild** is null, replace the null with a pointer to the node that would be visited **before ptr in an in-order traversal**
 - if **ptr -> rightChild** is null, replace the null with a pointer to the node that would be visited **after ptr in an in-order traversal**



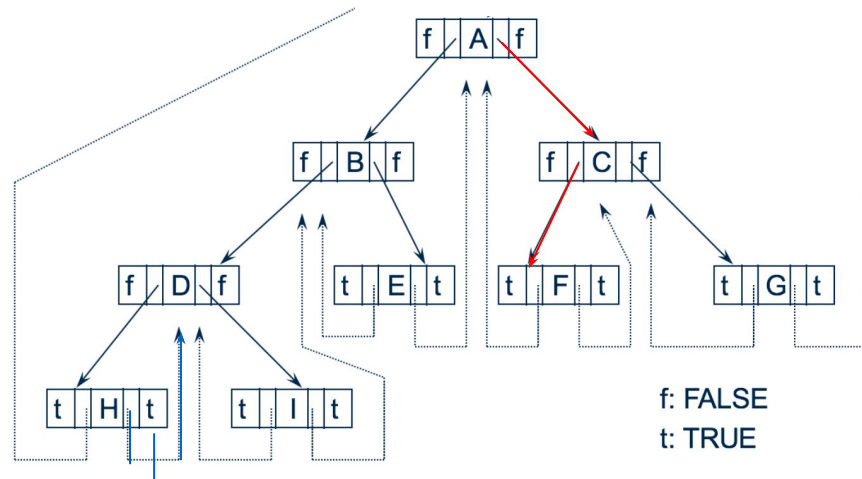
Threaded Binary Tree

- if **ptr -> leftThread** is **True**, ptr -> leftChild contains a thread
- if **ptr -> leftThread** is **False**, ptr -> leftChild contains a pointer to the left child



Threaded Binary Tree

- Find the in-order successor of ptr **without using stack**
 - if **ptr -> rightThread = TRUE**, ptr = ptr-> rightChild
 - otherwise follow a path of **leftChild links from the rightChild of ptr** until we reach a node with leftThread = TRUE



Threaded Binary Tree ADT

- **CreateTree** create a root of new threaded binary tree.
- **Insert** insert a new node in tree. If dynamic allocation is failed, just print error message.
Do not use recursive functions of stack.
- **printInorder** print the tree by inorder traversal. **Do not use recursive functions of stack.**
- **DeleteTree** free all memory of the threaded binary tree.

Threaded Binary Tree ADT

Structure

```
struct ThreadedTree {  
    int left_thread; // flag if ptr is thread  
    ThreadedPtr left_child;  
    ElementType data;  
    ThreadedPtr right_child;  
    int right_thread; // flag if ptr is thread  
}ThreadedTree;
```

Function

```
ThreadedPtr CreateTree();  
int Insert(ThreadedPtr root, int root_idx,  
           ElementType data, int idx);  
void printInorder(ThreadedPtr root);  
void DeleteTree(ThreadedPtr root);
```

Input & Output Example

input1.txt /data/data_bac...	output1.txt /data/data_bac...
8 1 2 3 4 5 6 7 8	inorder traversal : 8 4 2 5 1 6 3 7
일반 텍스트 ▼ 탭 너비: 4 ▼ 1행, 1열 ▼ 삽입	일반 텍스트 ▼ 탭 너비: 4 ▼ 1행, 1열 ▼ 삽입

Assignment

- Due
 - ~ **2023.04.05(Wed) 23:59**
 - Last Commit 기준
- 자세한 내용은 과제 명세 PDF 파일 참고