Thread Binary Tree

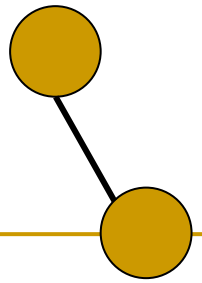
What is a Threaded Binary Tree ?

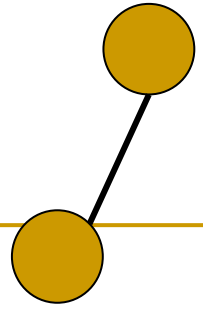
- 이진 트리를 연결리스트로 표현할 경우 실제 포인터 수보다 자식 노드가 없는 null 링크의 수가 더 많아진다
  - 총 노드의 개수를 n 이라고 할 경우, null 링크를 포함하는 총 링크의 수는 2n이며 그 중 null 링크의 수는 n+1 이 된다
  - 이진 트리 순회의 경우 재귀함수를 호출하여 실행하는 것은 이해에 도움을 주나 실행 성능에는 비효율적이다

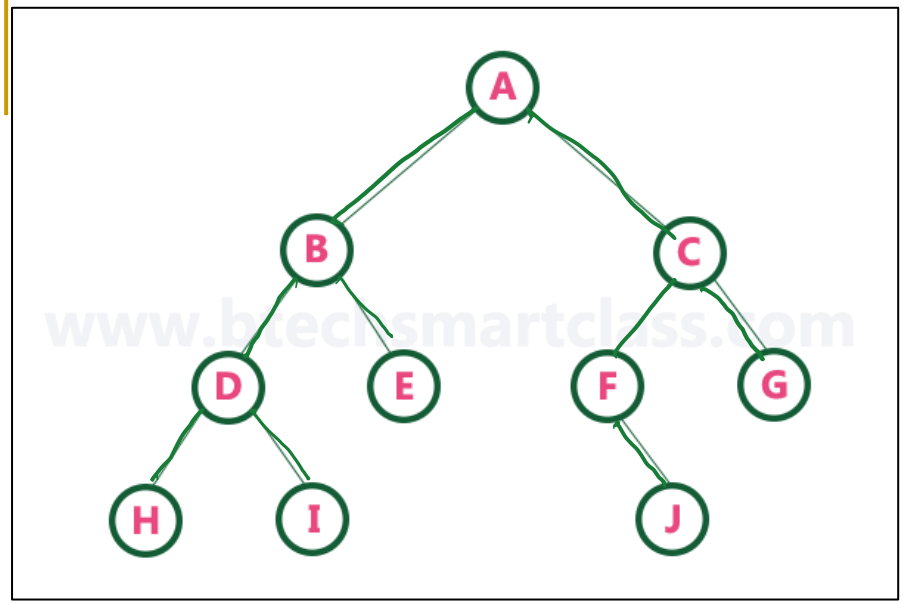
- 이진 트리의 NULL 링크를 이용하여 재귀호출 없이 반복문을 통해 (즉, 스택을 사용하지 않고)트리의 노드를 순회하는 방식을 취하는 아이디어 제안 (A. J. Perlis & C. Thornton)
  - null 링크를 다른 노드를 가리키도록 대체 링크를 지정하는데, 이러한 대체 링크를 스레드(thread)라고 하며, 스레드를 사용하여 변형된 이진 트리를 스레드 이진 트리(threaded binary tree)라고 한다

Thread를 구성하는 규칙

- **node -> left_child가 null**인 경우
  - 이를, inorder traversal에 있어서, node 직전에 방문한 노드를 가리키는 링크로 대체 (이를 node의 inorder predecessor, 중위순회 선행자라고 한다)
  - 직전에 방문한 노드가 존재하지 않는 경우 head 노드(root 노드를 가리키는 추가 노드)를 링크
- **node -> right_child가 null**인 경우
  - 이를, inorder traversal에 있어서, node 직후에 방문한 노드를 가리키는 링크로 대체 (이를 node의 inorder successor, 중위순회후속자라고 한다)
  - 직후에 방문한 노드가 존재하지 않는 경우 head 노드를 링크

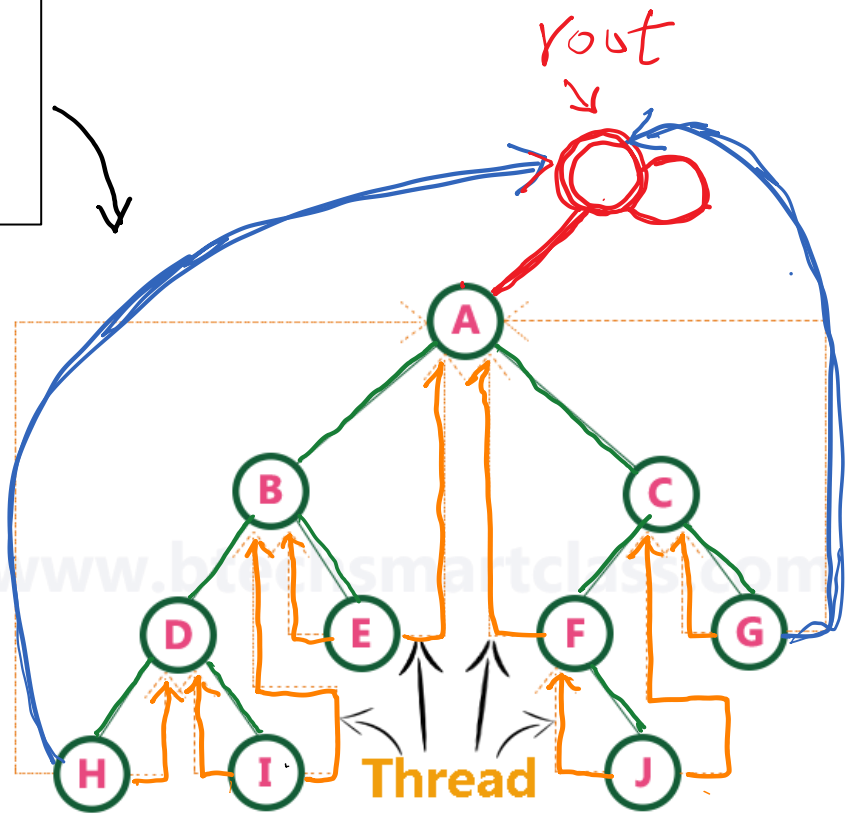






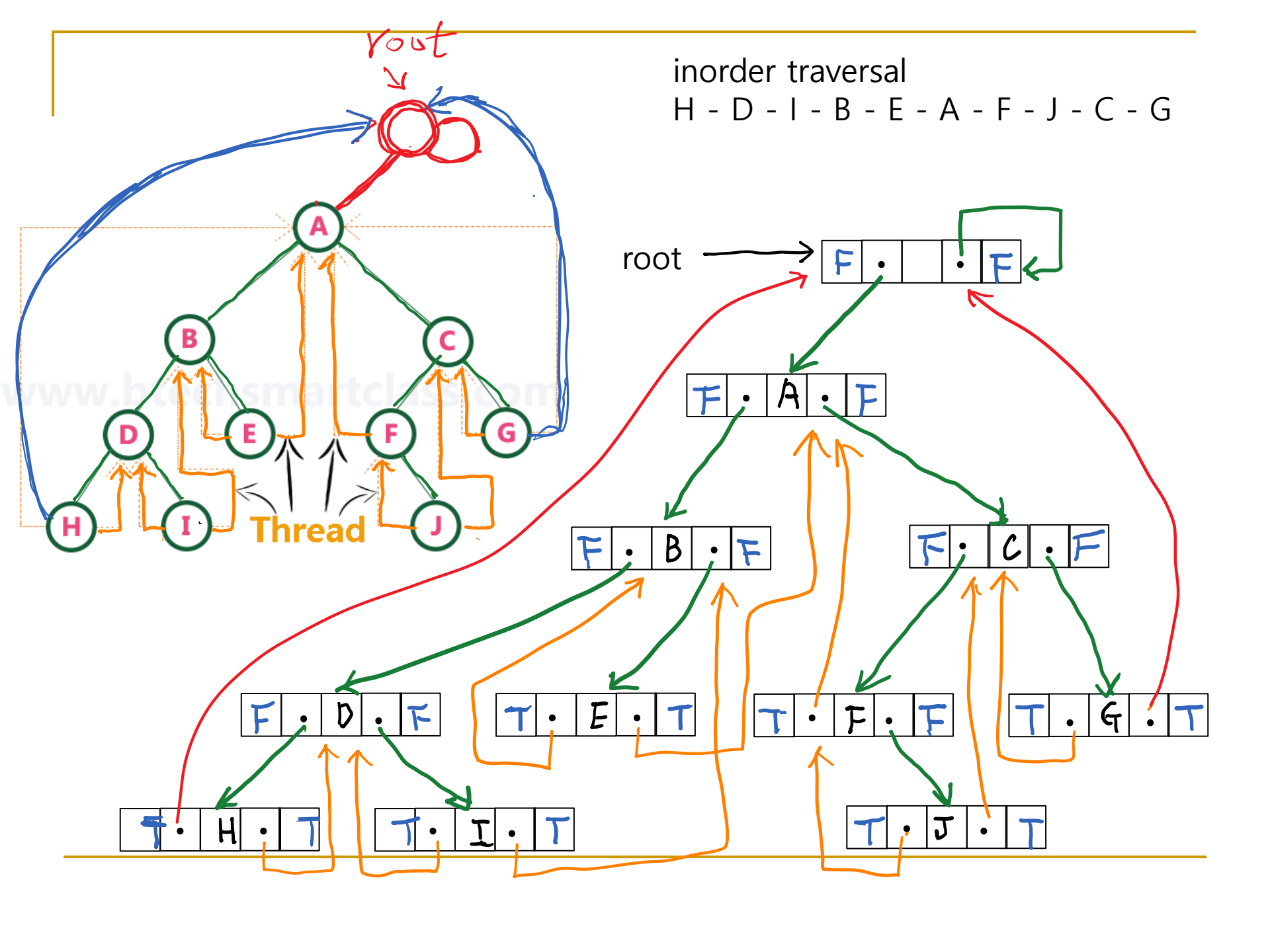
binary tree

inorder traversal
H - D - I - B - E - A - F - J - C - G



- **TBT 노드를 위한 데이터 구성**
  - **2개의 thread pointer : leftChild, rightChild**
  - **2개의 tags :**
    - **thread 링크인지 실제 링크인지 판단**
    - **leftThread, rightThread --- Boolean**

```
typedef struct threadedTree {
        short       leftThread;     // True or False
        threadPtr   leftChild;      // point to the left child
        char        data;
        threadPtr   rightChild;     // point to the right child
        short       rightThread;    // True or False
        // other members if necessary
} ;
typedef struct threadedTree     *threadPtr;
```


root
A
B
C
D
E
F
G
H
I
J
Thread
inorder traversal
H - D - I - B - E - A - F - J - C - G

inorder successor 노드를 찾는게 중요

```
//find the inorder successor of threaded binary tree
threadedPtr  inorderSuc(threadedPtr ttree) {
        threadedPtr         temp;
        temp = ttree->rightChild;
        if(!ttree->rightThread)
                while(!temp->leftThread)
                        temp = temp->leftChild;
        return temp
}
void traverseInorder(threadedPtr  ttree) {
        threadedPtr  temp = ttree;
        for( ; ; ) {
                temp = inorderSuc(temp);
                if(temp == ttree) break;
                printf("%c", temp->data);
        }
}
```