자료구조 Data Structure | 조행래

트리

이진트리의 추가 연산

학습 목표

- 이진 트리의 추가적인 연산들을 이해
  - 이진 트리의 복사 알고리즘
  - 이진 트리의 동일성 검사 알고리즘
  - 이진 트리의 노드 수를 구하는 알고리즘
  - 이진 트리의 단말 노드 수를 구하는 알고리즘

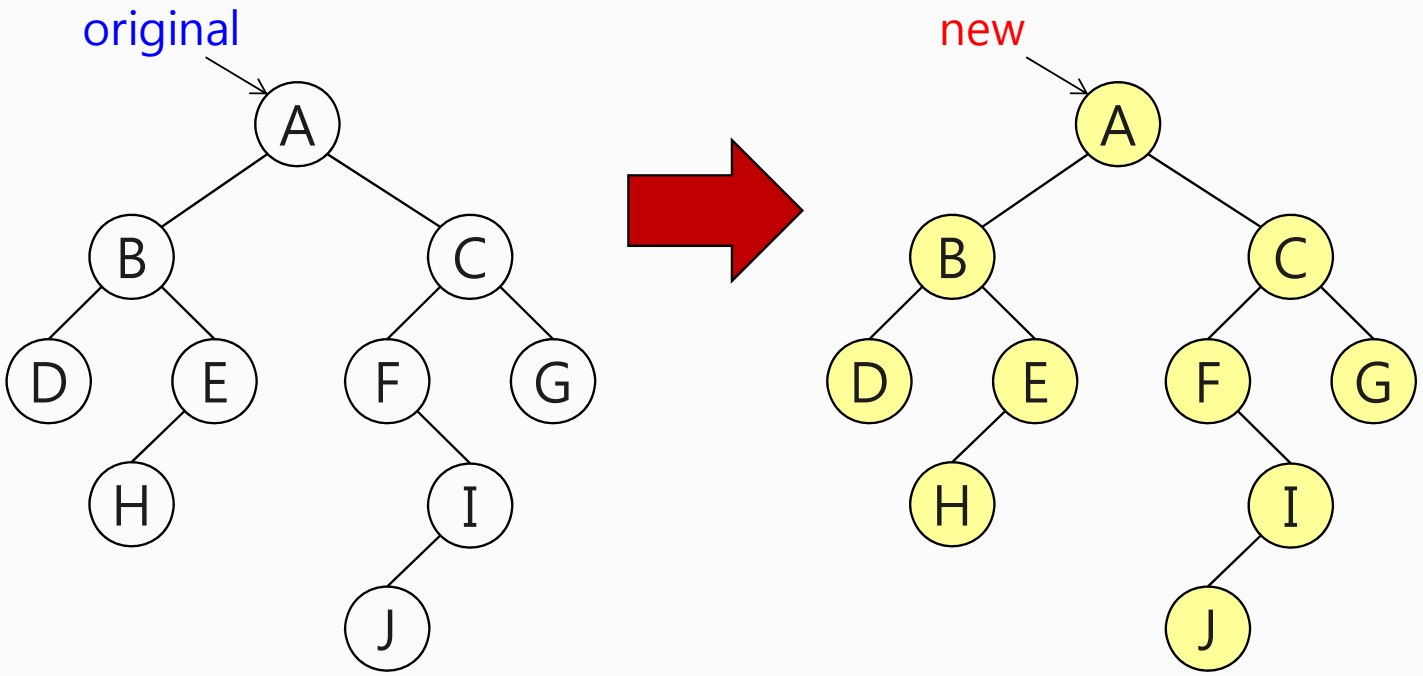
1. 이진 트리의 추가 연산

- **추가 연산의 종류**
  - 이진 트리의 복사
  - 이진 트리가 동일한지 검사
  - 이진 트리의 노드 수 계산
  - 이진 트리의 단말 노드 수 계산

- **트리의 모든 노드들을 방문할 필요성**
  - 이진 트리의 순회 알고리즘들을 응용

2. 이진 트리의 복사

- **문제 설명**
  - 입력 이진 트리의 노드 구조가 동일한 새로운 이진 트리를 생성하여 루트 노드의 주소를 반환
  - 후위 순회 알고리즘을 응용

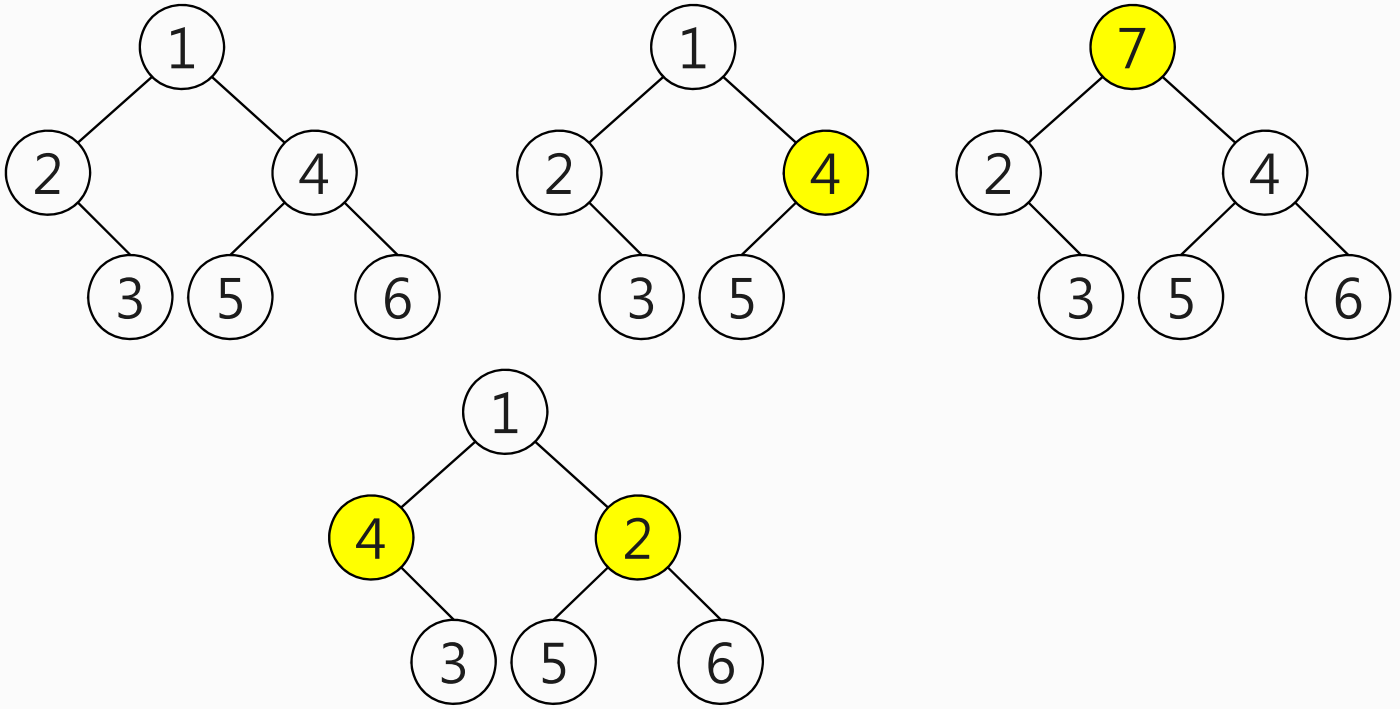


이진 트리의 복사 알고리즘

```
struct node *copy(struct node *original)
{  // original 트리를 복사한 새로운 이진 트리를 반환
    struct node *temp;
    if (original != NULL) {
        temp = (struct node *) malloc(sizeof(struct node));
        temp→left_child = copy(original→left_child);
        temp→right_child = copy(original→right_child);
        temp→data = original→data;
        return temp;
    }
    return NULL;
}
```

3. 이진 트리의 동일성 검사

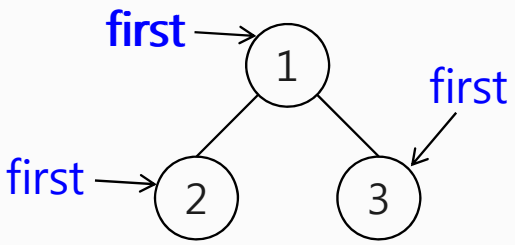
- **문제 설명**
  - 두 개의 이진 트리가 동일한 데이터와 동일한 구조(부모-자식, 형제 등)를 갖는지를 검사
  - 전위 순회 알고리즘을 응용



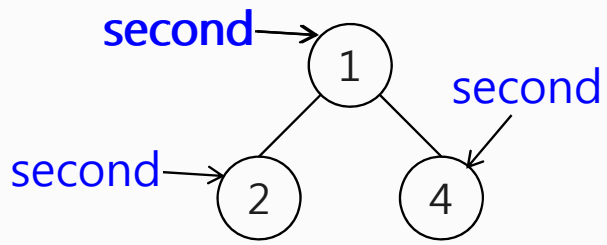
동일성 검사 알고리즘

```
int equal(struct node *first, struct node *second)
{
    /* first와 second 트리가 다를 경우 FALSE를 반환.
       트리가 동일할 경우, TRUE를 반환 */

    return ((!first && !second) || (first && second &&
        (first→data == second→data) &&
        equal(first→left_child, second→left_child) &&
        equal(first→right_child, second→right_child)));
}
```



first = NULL



second = NULL

4. 이진 트리의 노드 수 계산

- **접근 방법**
  - 루트 노드가 NULL이면 0을 반환
  - NULL이 아니면, "1+ 왼쪽 서브트리의 노드 수 + 오른쪽 서브트리의 노드 수" 를 반환
    - 서브트리의 노드 수? → 순환 알고리즘

```
int get_node_count(struct node *ptr)
{
    int count = 0
    if (ptr != NULL)
        count = 1 + get_node_count(ptr→left_child) +
                    get_node_count(ptr→right_child);
    return count;
}
```

응용 문제: 트리의 높이 계산

5. 이진 트리의 단말 노드 수 계산

- **접근 방법**
  - 루트 노드가 NULL이면, 0을 반환
  - 단말 노드이면, 1을 반환
  - 자식이 있을 경우, "왼쪽 서브트리의 단말 노드 수 + 오른쪽 서브트리의 단말 노드 수" 를 반환
    - 서브트리의 단말 노드 수? → 순환 알고리즘

단말 노드 수 계산 알고리즘

```
int get_leaf_count(struct node *ptr)
{
    int count = 0;

    if (ptr != NULL) {
        if (ptr→left_child == NULL &&
            ptr→right_child == NULL)        // 단말 노드
                return 1;
        else  count = get_leaf_count(ptr→left_child) +
                      get_leaf_count(ptr→right_child);
    }
    return count;
}
```

요약 정리

- 이진 트리의 추가 연산에 대해서 설명
  - 이진 트리의 복사/동일성 검사/노드 수 계산 등
  - 이진 트리의 순회 알고리즘을 응용한 순환 함수로 구현