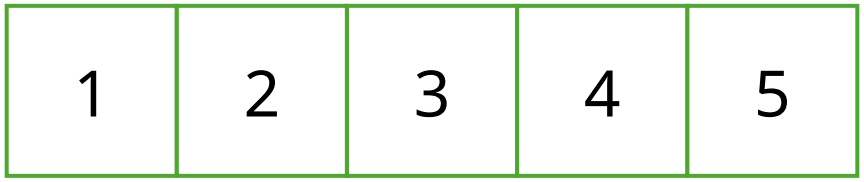
Week2

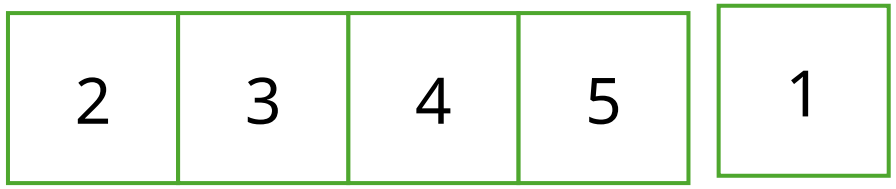
2024-07-12

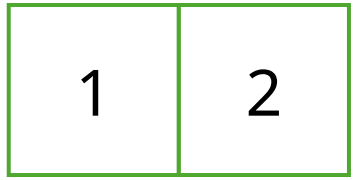
rotate

왼쪽으로 2 번 밀기

N = 5







[3, 4, 5, 1, 2]

rotate 의 특징

- 왼쪽으로 i 번 로테이션 하는 것은 오른쪽으로 n – i 로테이션 하는것과 같음

- 1 2 3 4 5 -> [2, 3, 4, 5, 1]

- 1 2 3 4 5 -> 5 1 2 3 4 -> 4 5 1 2 3 -> 3 4 5 1 2 -> [2, 3, 4, 5, 1]

오른쪽에 가까운지 , 왼쪽에 가까운지 찾고 정답에 더한 후 ,
한 방향으로 rotate 하면 코드가 간결해질 수 있다

(속도는 느림)

덱을 사용하지 않고 로테이션 하는 방법

[1, 2, 3, 4, 5] 배열을 왼쪽으로 두 칸 밀어서
[3, 4, 5, 1, 2] 로 만들려면 ??

추가적인 배열 사용 불가

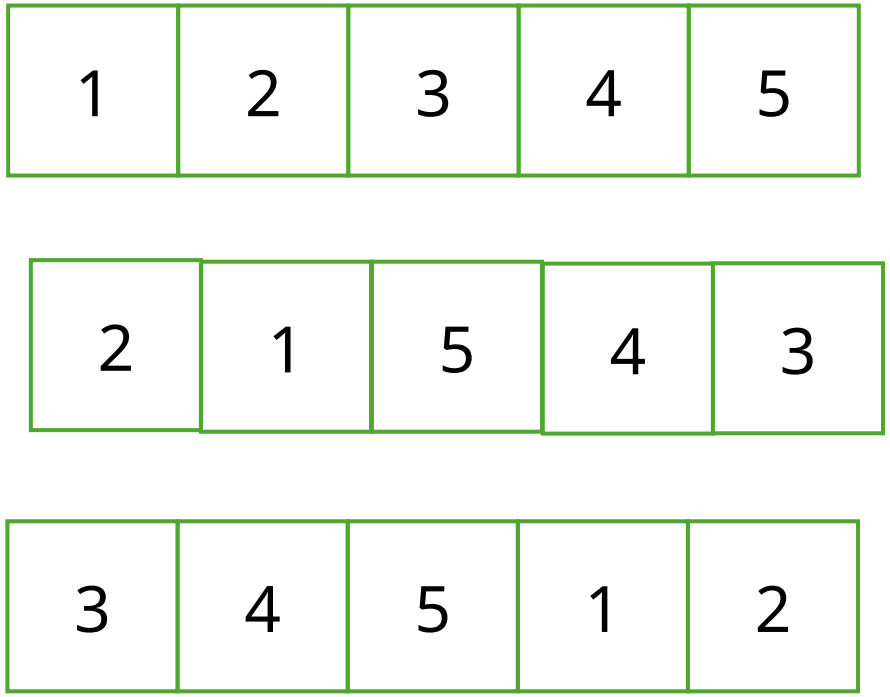
reverse 연산

왼쪽 2 개를 reverse, 오른쪽 3 개를 reverse, 전체를 reverse

[1, 2] [3, 4, 5]

[2, 1] [5, 4, 3]

[3, 4, 5, 1, 2]



reverse 연산

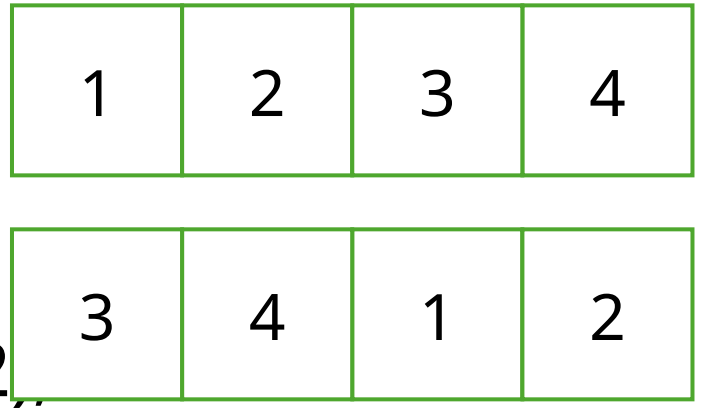
장점 :
다른 자료구조나 추가적인 배열이 필요하지 않음 (메모리 절약)
코드가 짧아짐

단점 :
매번 N 의 연산이 필요함 (비교적 많은 연산)
모르는 사람이 봤을 때 코드를 이해하기 어려움

reverse 연산

짝수인 배열의 절반을 나눠서 좌 우를 바꾸는 경우에 사용할 수 있음

재귀적으로 호출할 수 있음

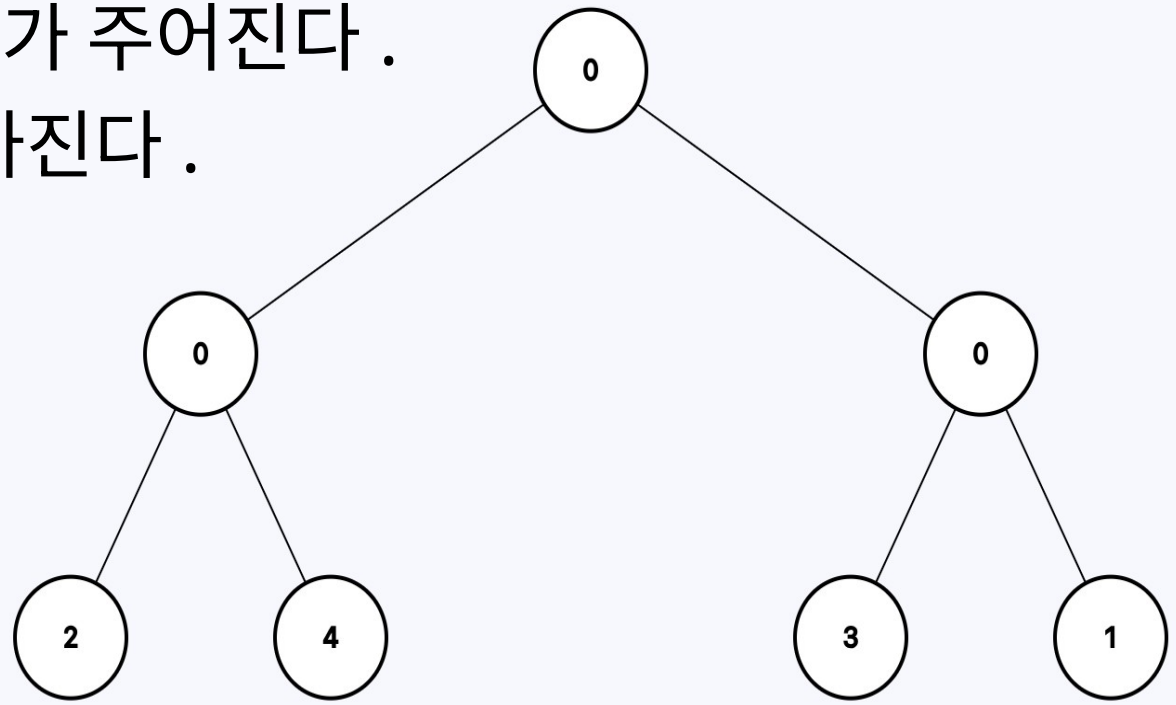



```
reverse(a.begin(), a.begin() + a.size() / 2);
reverse(a.begin() + a.size() / 2, a.end());
reverse(a.begin(), a.end());
```

심화 . 문제

문제 (소마 코테 , 정확하진 않음)

리프노드의 개수가 n 인 포화 이진트리가 주어진다 .
리프 노드는 [1, n] 의 서로 다른 값을 가진다 .



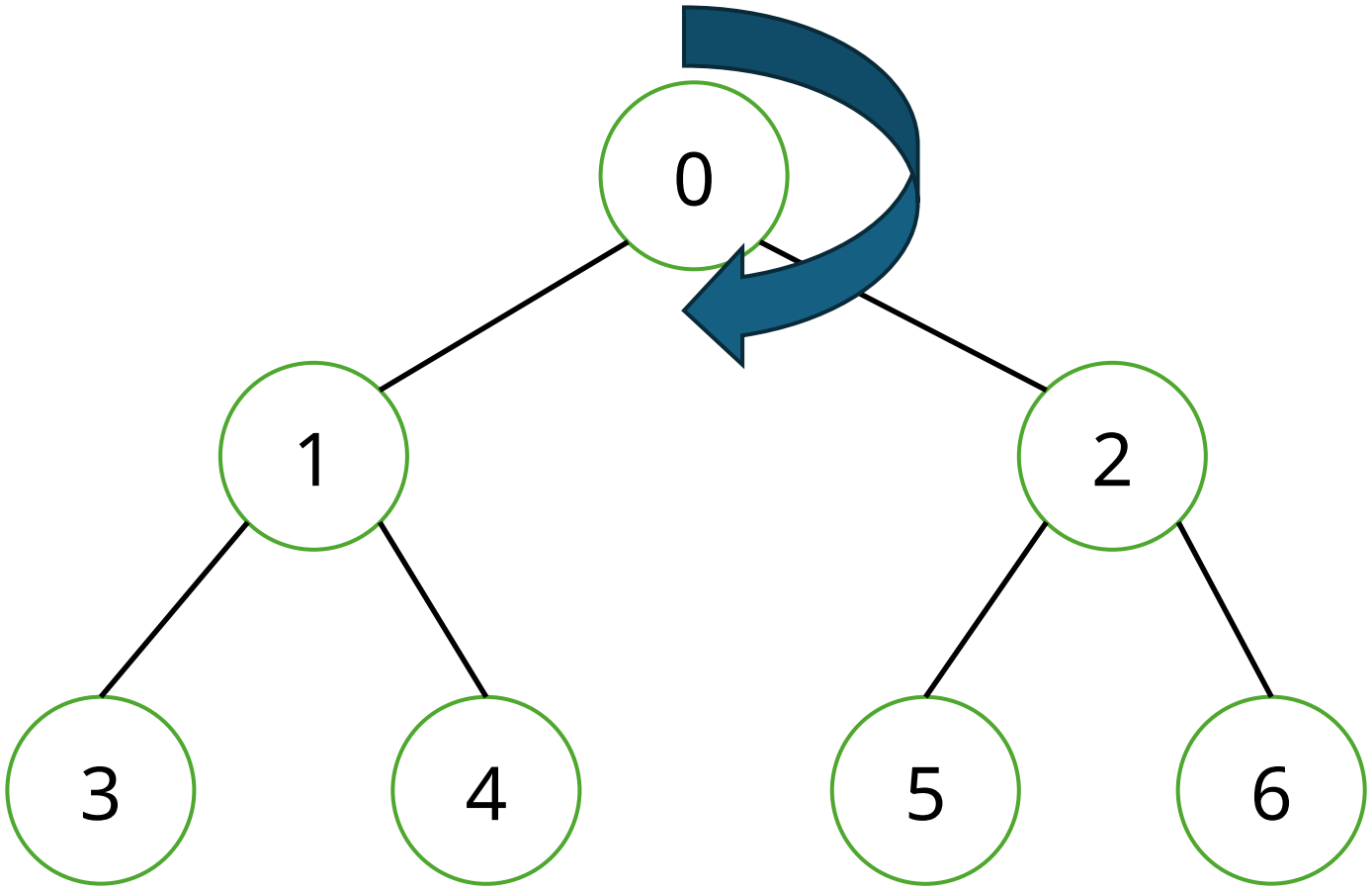
각 노드는 회전 연산을 할 수 있다 .

횟수와 위치 상관없이 연산을 진행하여 리프노드를 정렬할 수 있는지 ?

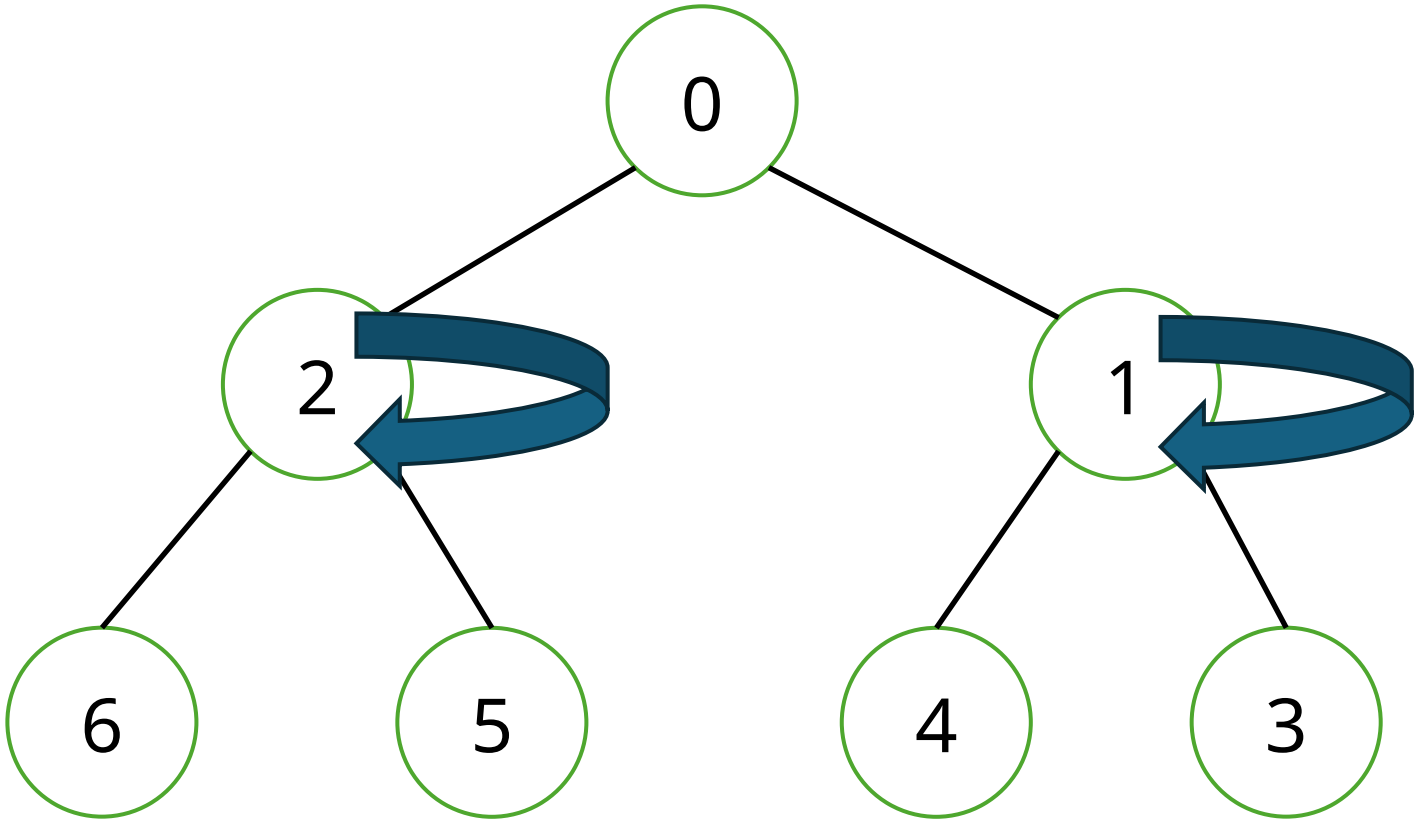
트리 회전 이해하기

- 특정 노드 아래의 모든 노드 좌우가 바뀜
- 샹들리에를 돌리는 것처럼 생각

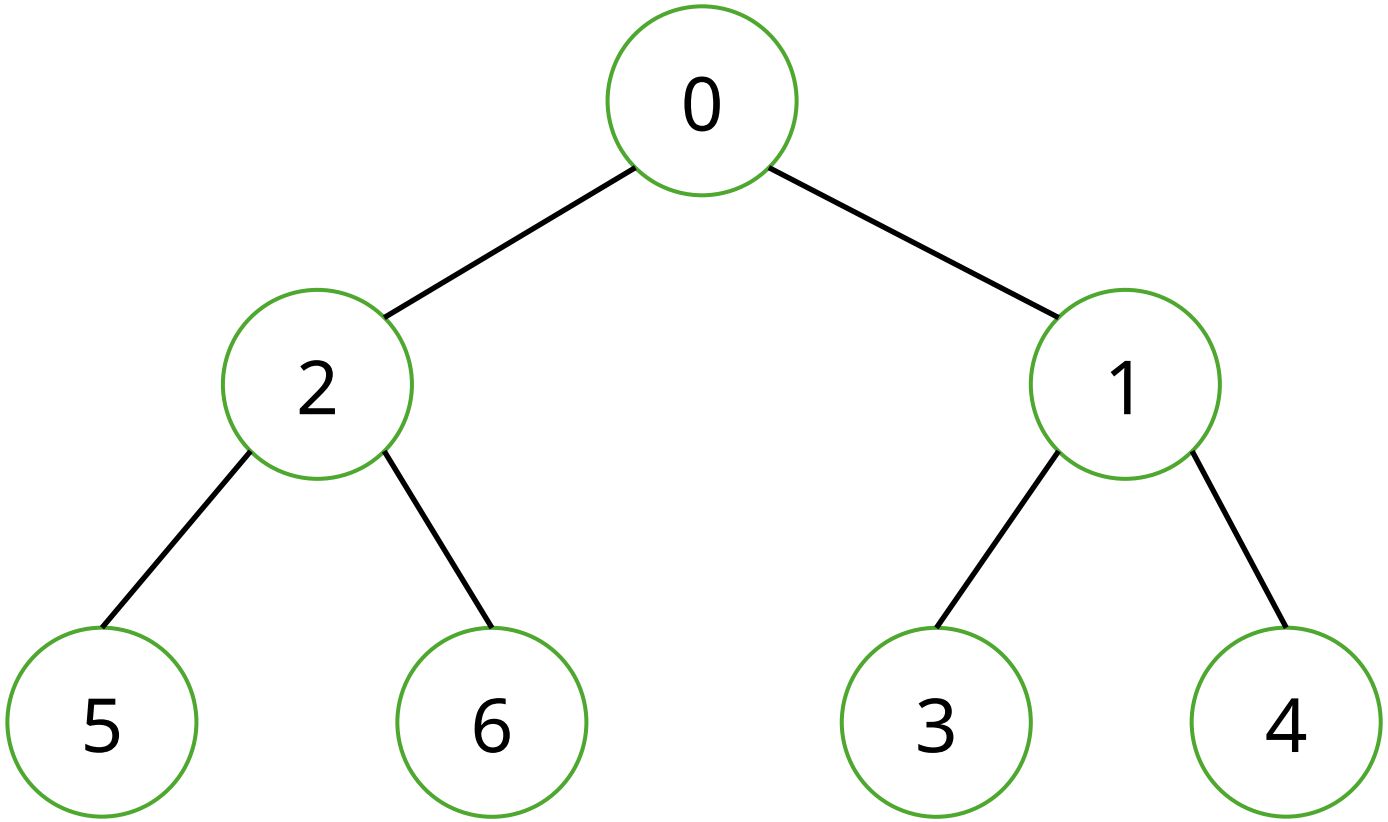
회전



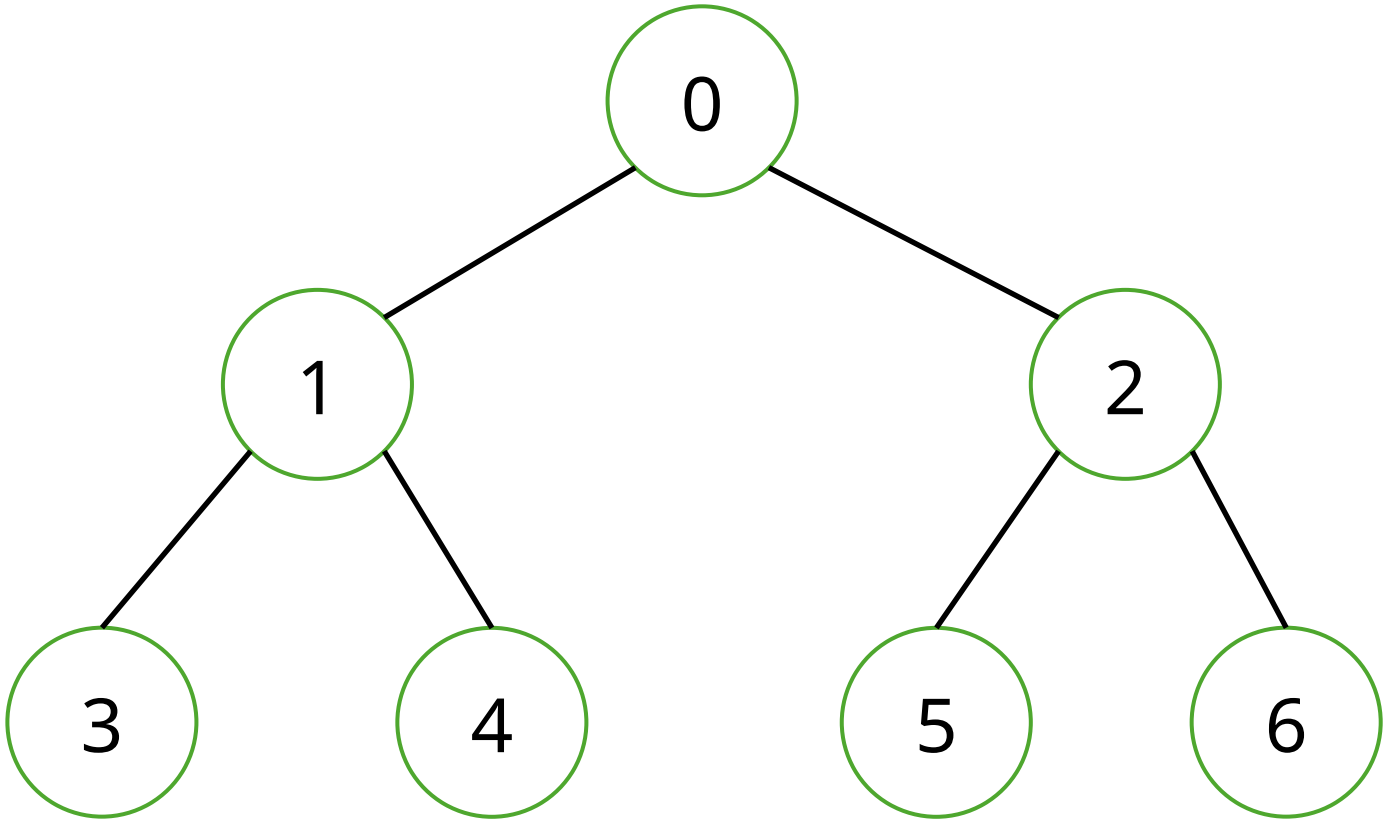
회전



회전

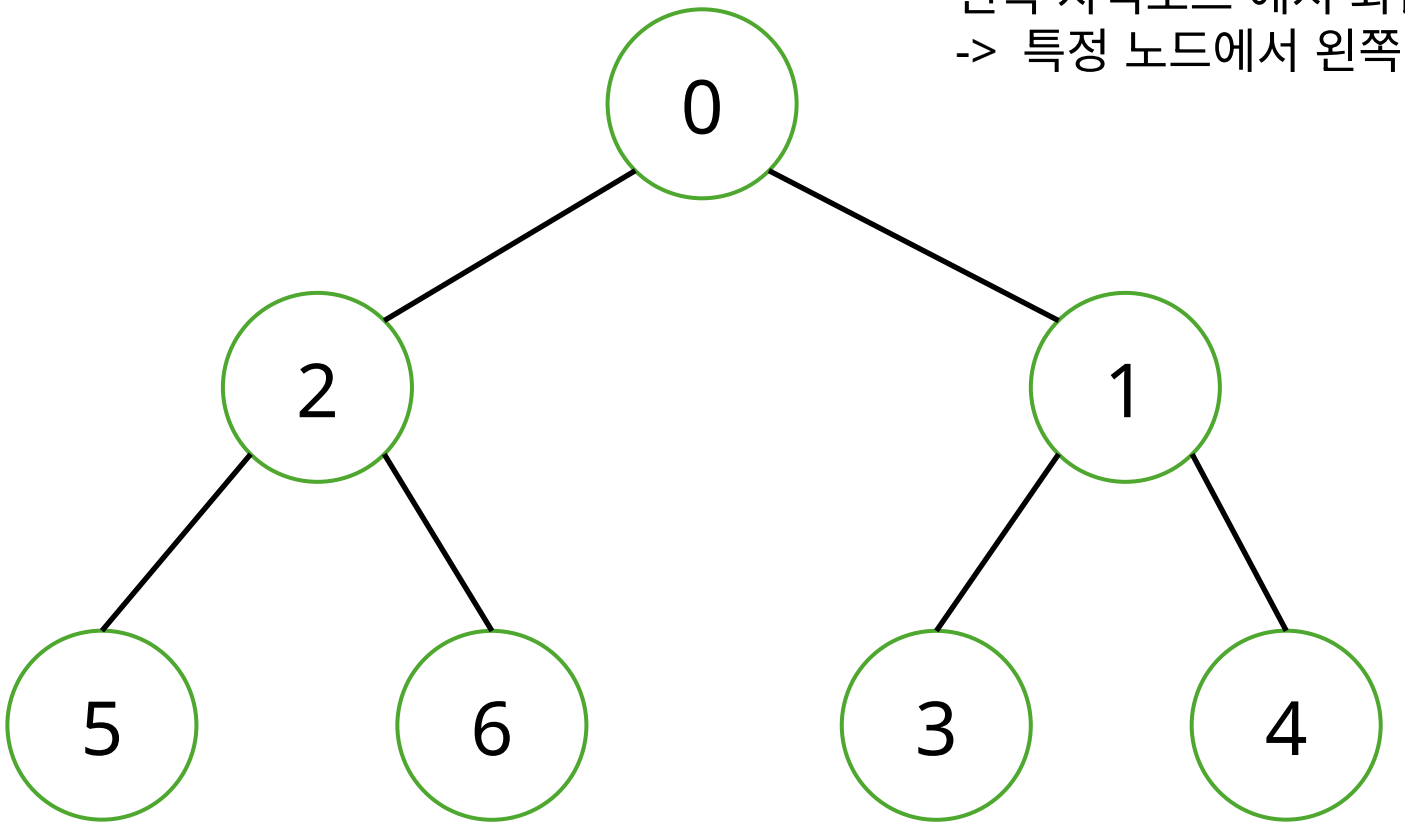


회전



회전

이진트리에서
특정 노드에서 회전하고 ,
왼쪽 자식노드 에서 회전 , 오른쪽 자식노드에서 회전
-> 특정 노드에서 왼쪽 자식과 오른쪽 자식을 바꿈

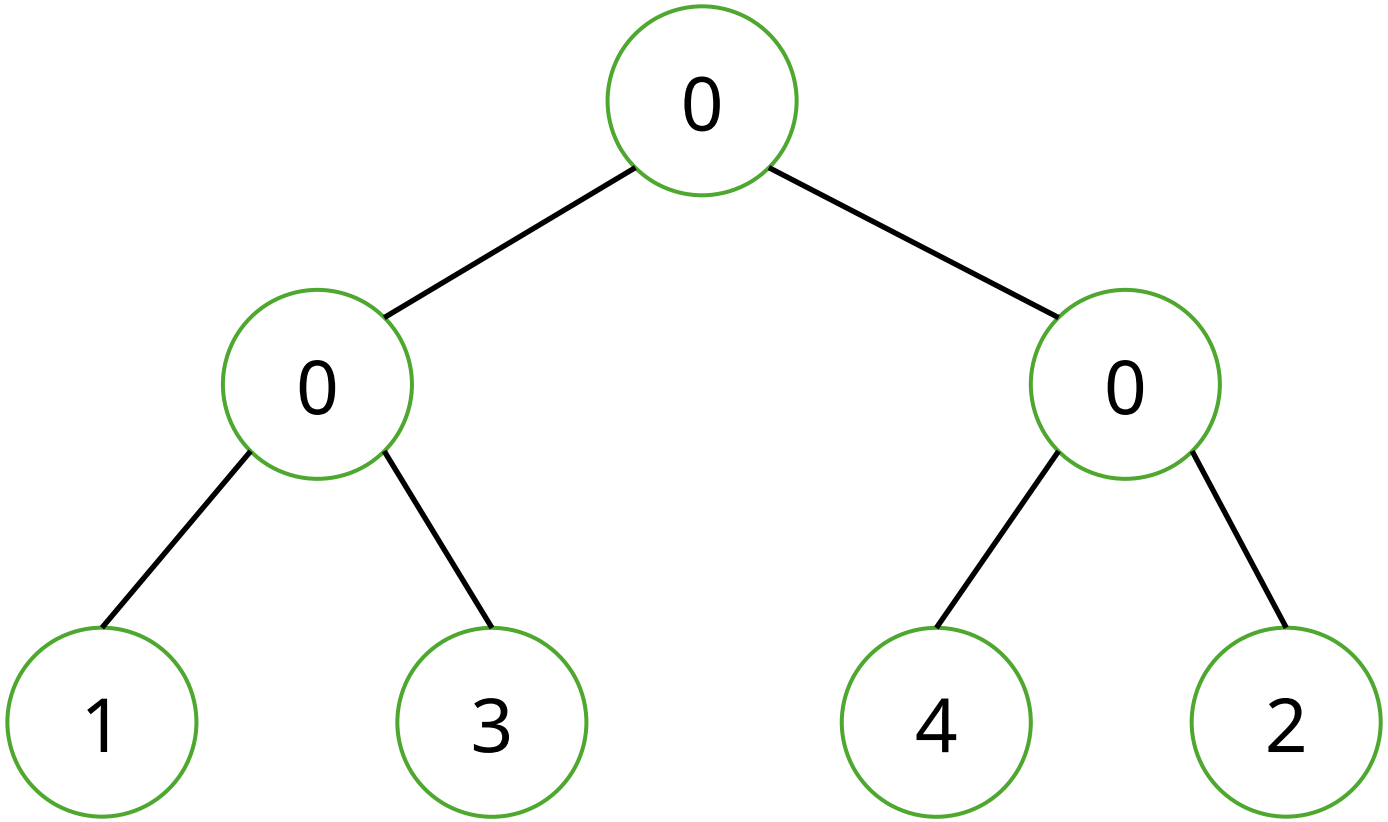


회전

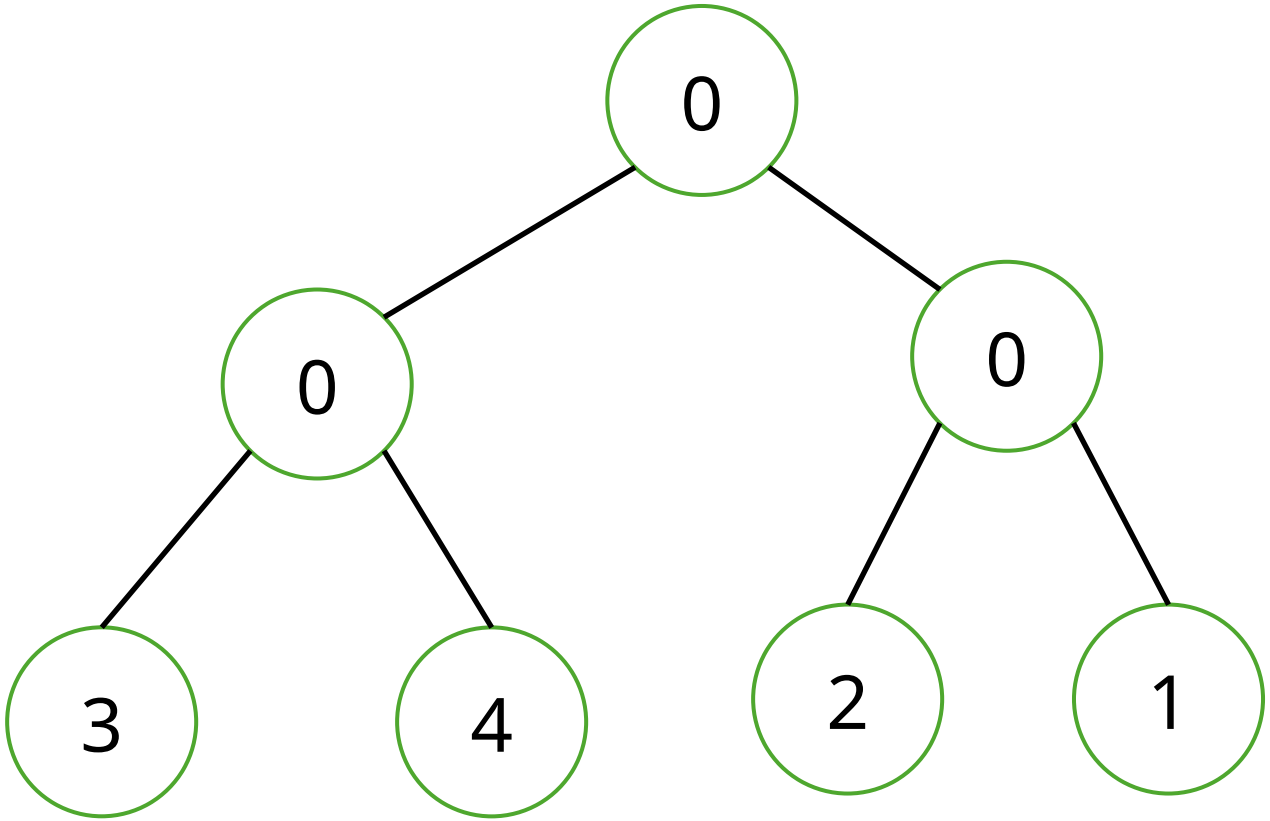
회전으로 정렬하는 것보단 좌 우를 옮겨서 정렬하는 게 훨씬 쉽다

불가능 한 경우

불가능 [1, 3, 4, 2]



가능 [3, 4, 1, 2]



[3, 4, 1, 2]

[1, 2, 3,4]

풀기

- 그리디 + 재귀

- 그리디하게 재귀적으로 정렬을 시도해 보고
- 마지막에 제대로 정렬 되었는지 확인함

```
for(int i = 0; i < n; i++) {
        if(a[i] != i + 1) return false;
}
```

풀이

- 분할정복
- 함수를 정의하고 해당 함수가 항상 특정 동작을 한다고 생각함
- 재귀를 탈출하기 위한 베이스 케이스가 필요함

rec(array, start, end) ->
array[start.. end) 를 정렬하는 동작을 함

순서
rec(array, start, end)
-> rec(array, start, mid), rec(array,mid, end) 호출
-> 왼쪽 자식 , 오른쪽 자식이 정렬 되었으므로
-> 왼쪽 자식의 왼쪽 리프노드 , 오른쪽 자식의 왼쪽 리프노드를 비교한다 .
-> 왼쪽 자식의 왼쪽 리프노드가 더 크다면 왼쪽 자식과 오른쪽 자식을 바꾼다 .

코드 ...

```
rec (a, 0, n);
void rec(int a[], s, e) {
        //  탈출 조건 .  리프노드까지 온 경우
        if(s + 2 == e) {
                if(a[s] > a[s + 1]) {
                        swap(a[s], a[s + 1]);
                }
                return;
        }
        int mid = (s + e) / 2;
        rec(a, s, mid)
        rec(a, mid, e);
        if(a[s] > a[mid]) {
                여기서 회전  3 번으로 좌우 바꿈
        }
        return ;
```

} 하고나서 for 문으로 순서대로인지 확인