알고리즘 멘토링

- Segment Tree –

- 주민찬 -

Segment Tree

index	0	1	2	3	4	5
arr	5	1	-4	8	2	7

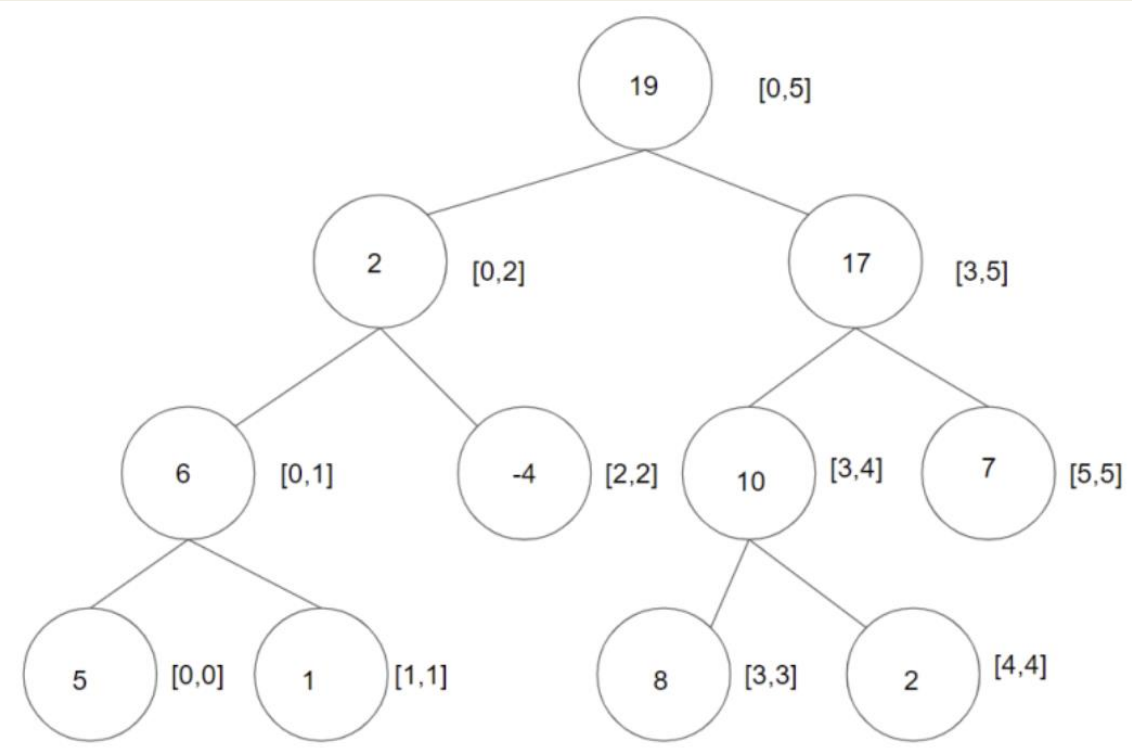
이 리스트에서 [L,R]의 합을 구하고 싶을 때 어떻게 해야 할까
➔ 누적 합 이용 $O(1)$ (누적 합 만드는 것 제외)

그런데 리스트의 3번째 요소가 2로 바뀌었을 경우 누적 합을 이용하면 다시 만들어야 한다.
➔ $O(n)$

합을 구하는 과정을 n번 진행하고 그 과정에서 원소가 계속 바뀔 때 더 빨리 할 수 없을까?
➔ Segment Tree

Segment Tree

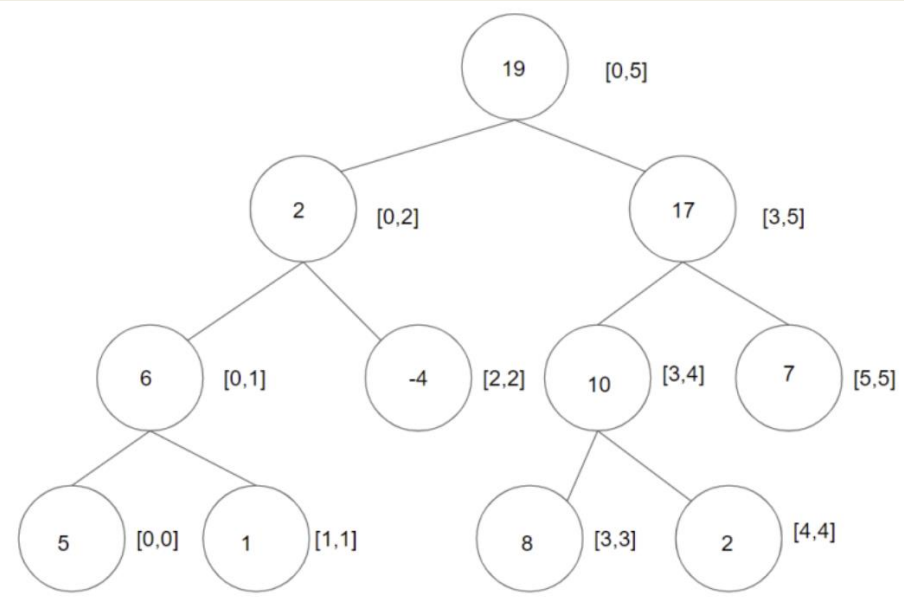
index	0	1	2	3	4	5
arr	5	1	-4	8	2	7



이진 트리를 이용
반으로 나누어 가면서 저장

이진 트리의 장점
1. 리스트로 표현 가능
2. 자식 노드 바로 탐색 가능
3. 시간복잡도 $O(logn)$

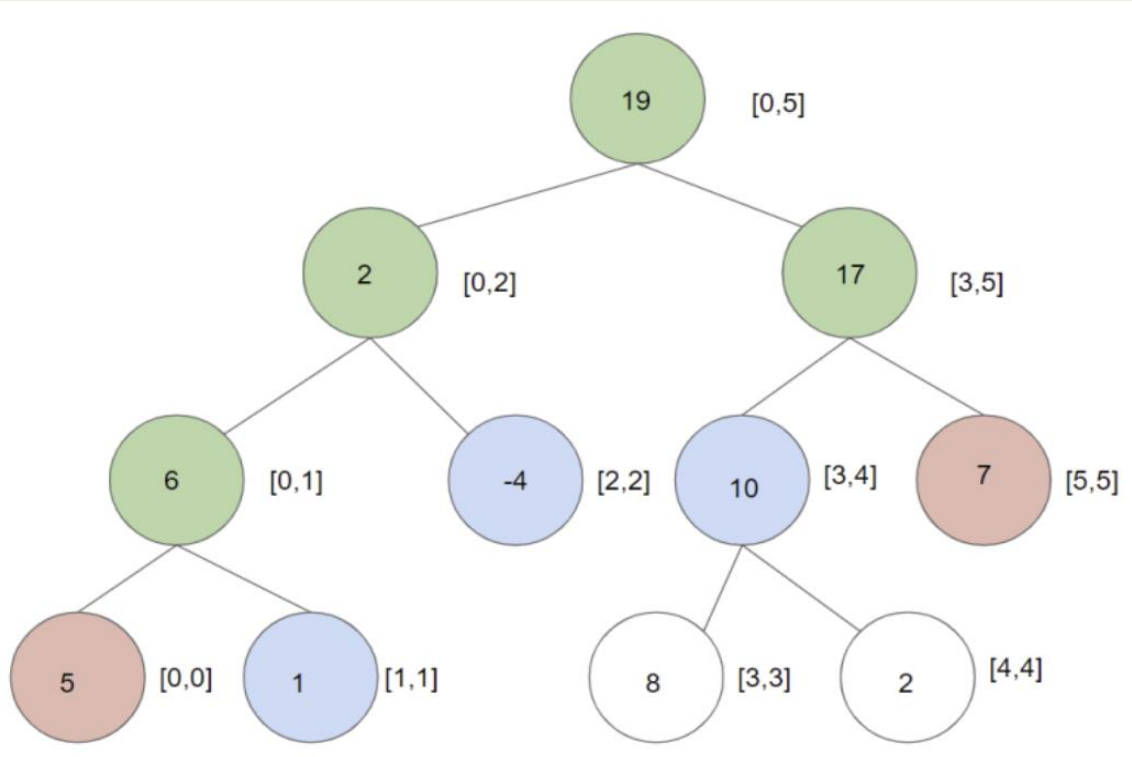
Segment Tree




```
def make_segment_sum(tree,ls,s,e,node):
    if s == e:
        tree[node] = ls[s]
        return tree[node]
    mid = (s+e)//2
    tree[node] = make_segment_sum(tree,ls,s,mid,node*2) + make_segment_sum(tree,ls,mid+1,e,node*2+1)
    return tree[node]
```

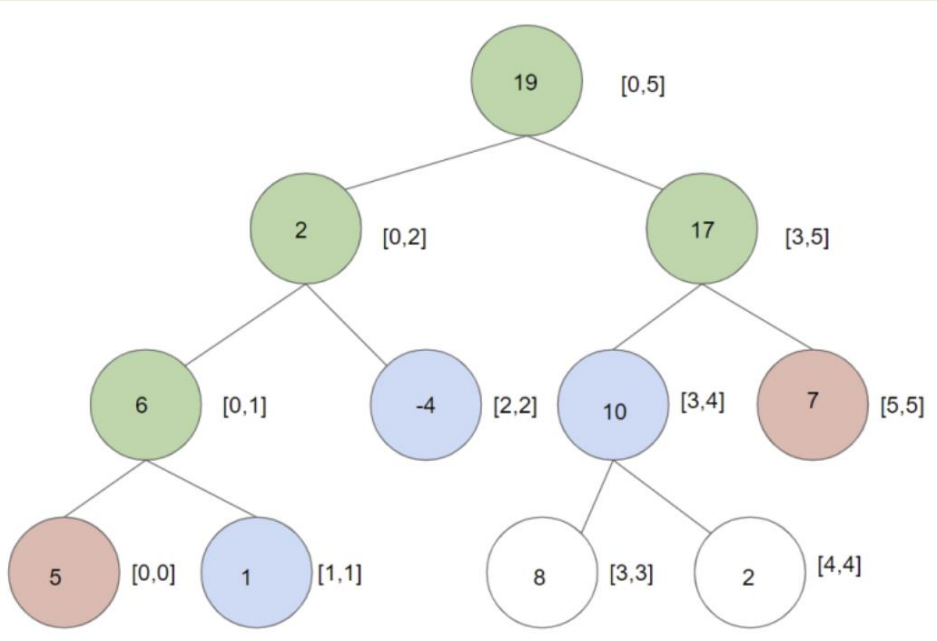
Segment Tree

index	0	1	2	3	4	5
arr	5	1	-4	8	2	7



[1,4]의 합을 구하고 싶을 때는?

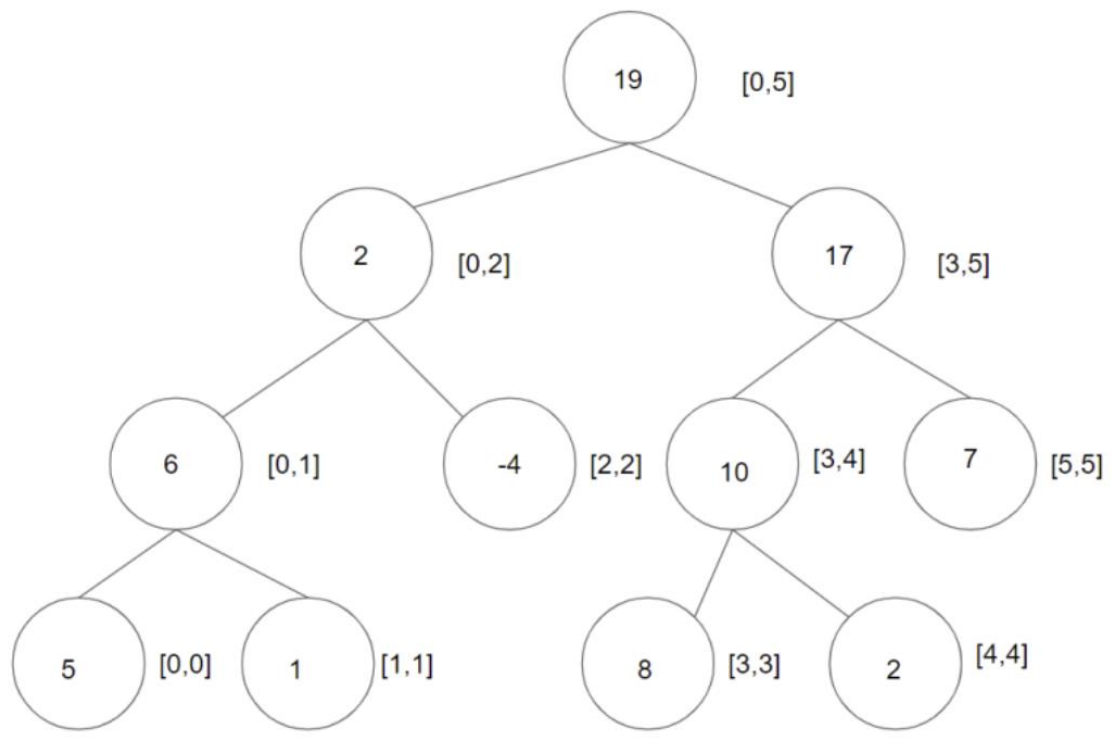
Segment Tree




```
def segment_sum(tree,s,e,node,i,j):
    if j < s or e < i: return 0
    if i <= s and e <= j: return tree[node]
    mid = (s+e)//2
    return segment_sum(tree,s,mid,node*2,i,j) + segment_sum(tree,mid +1,e,node*2+1,i,j)
```

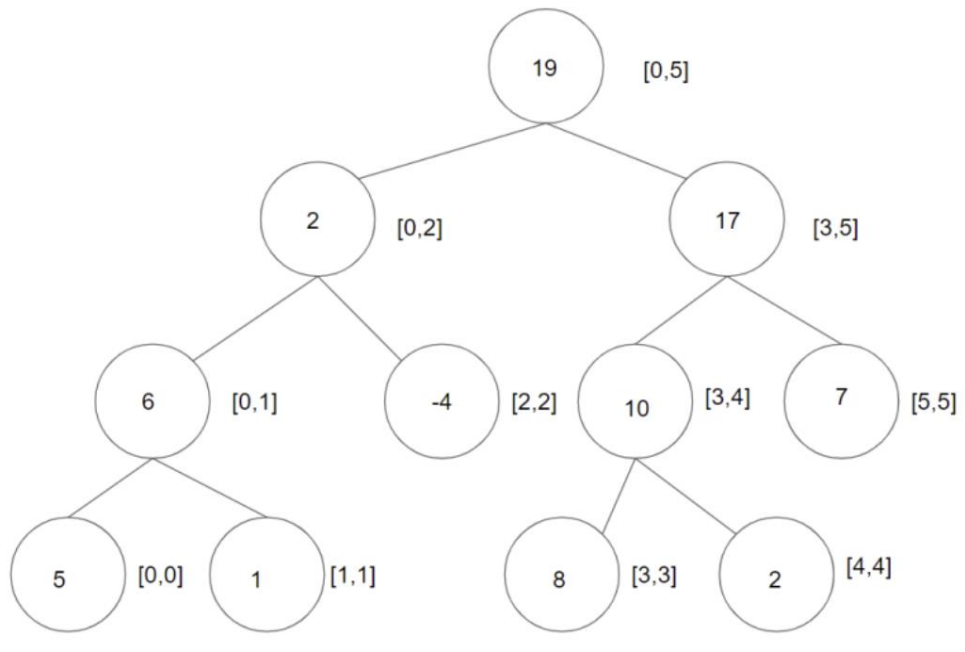
Segment Tree

index	0	1	2	3	4	5
arr	5	1	-4	8	2	7



리스트의 2번째 요소를 3으로 변경

Segment Tree




```python
def segment_change_sum(tree,s,e,node,i,j):
    if i < s or e < i: return
    if s == e:
        tree[node] = j
        return
    mid = (s+e)//2
    segment_change_sum(tree,s,mid,node*2,i,j)
    segment_change_sum(tree,mid+1,e,node*2+1,i,j)
    tree[node] = tree[node*2] + tree[node*2+1]
    return
```


실습!!