Segment Tree with Lazy Propagation

간단한 문제

- 길이 N인 배열 arr이 주어진다
- l, r $(1 \leq l, r \leq N)$이 주어졌을 때,
- arr[l] + arr[l+1] + … + arr[r-1] + arr[r] 의 값을 구하자

- Naïve (Bruteforce)

```
for(int i = l; i <= r; i++) ans += arr[i];
```

조금 생각해볼 만한 문제

- 길이 N인 배열 arr이 주어진다
- l, r $(1 \leq l, r \leq N)$이 주어졌을 때,
- arr[l] + arr[l+1] + … + arr[r-1] + arr[r] 의 값을 Q**번** 구하자

- $N, Q \leq 100\,000$
- Naïve (Bruteforce) : $O(NQ)$

조금 생각해볼 만한 문제

1	4	2	6	7	9	2	6	8	4

1	5	7	13	20	29	31	37	45	49

- 누적 합 (Prefix Sum): $O(N+Q)$

```
for(int i = 1; i <= N; i++) p[i] = p[i-1] + a[i];
for(int i = 1; i <= Q; i++) ans = p[r] - p[l-1];
```

어려운 문제 (BOJ 2042)

- 길이 N인 배열 arr이 주어진다
- l, r $(1 \leq l, r \leq N)$이 주어졌을 때, 다음 두 가지 연산 중 하나를 Q번 수행한다.
- `arr[l] + arr[l+1] + … + arr[r-1] + arr[r]` 의 값을 구하기
- `arr[i]`의 값을 k로 변경하기

- Prefix sum with update? $O(NQ)$
- 최악의 경우는, $i = 1$인 변경 쿼리가 계속해서 들어올 때

시간 초과

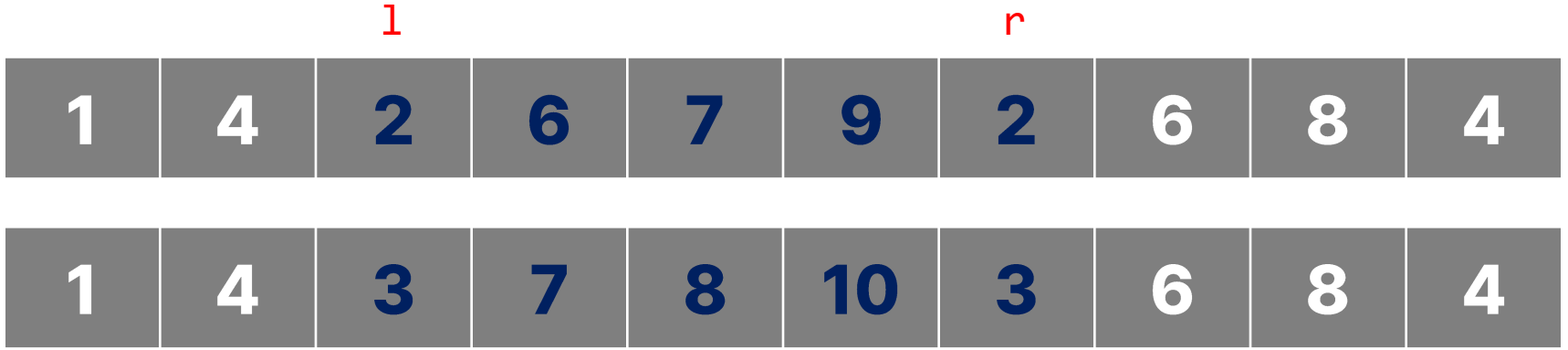
어려운 문제 (BOJ 2042)

- 길이 N인 배열 arr이 주어진다
- l, r $(1 \leq l, r \leq N)$이 주어졌을 때, 다음 두 가지 연산 중 하나를 Q번 수행한다.
- `arr[l] + arr[l+1] + … + arr[r-1] + arr[r]` 의 값을 구하기
- `arr[i]`의 값을 k로 변경하기

- Segment Tree를 활용해 쿼리당 $O(logN)$
- 총 시간복잡도 $O(QlogN)$

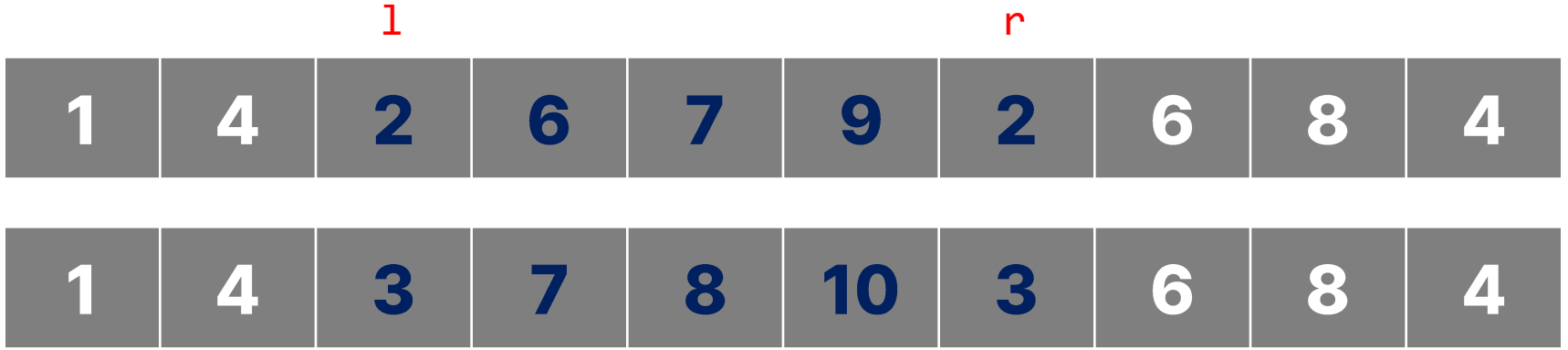
더 어려운 문제 (BOJ 10999)

- 길이 N인 배열 arr이 주어진다
- l, r $(1 \leq l, r \leq N)$이 주어졌을 때, 다음 두 가지 연산 중 하나를 Q번 수행한다.
- arr[l] + arr[l+1] + … + arr[r-1] + arr[r] 의 값을 구하기
- arr[l..r]의 값에 정수 k를 더하기



더 어려운 문제 (BOJ 10999)

- 길이 N인 배열 arr이 주어진다
- l, r $(1 \leq l, r \leq N)$이 주어졌을 때, 다음 두 가지 연산 중 하나를 Q번 수행한다.
- `arr[l] + arr[l+1] + … + arr[r-1] + arr[r]` 의 값을 구하기
- `arr[l..r]`의 값에 정수 k를 더하기



- Segment Tree를 활용한다고 하더라도 쿼리 범위에 따라 최악의 경우 $O(QNlogN)$

Segment Tree

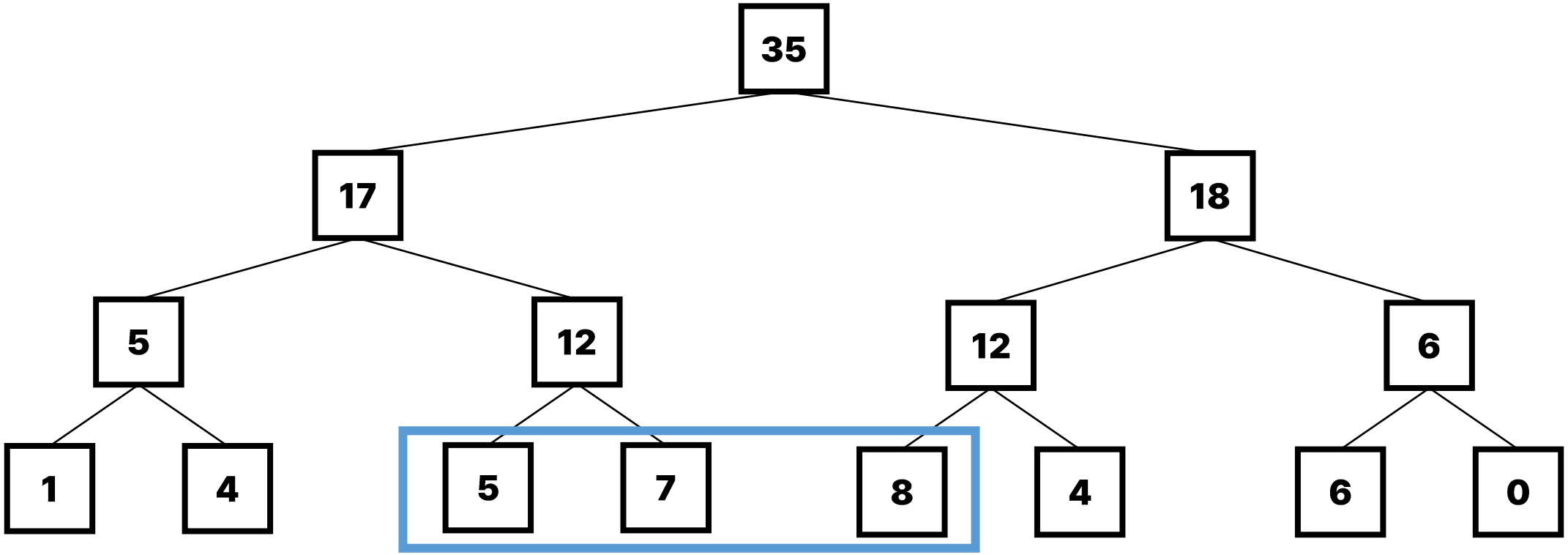
- 세그먼트 트리는 각 노드가 구간의 대푯값을 저장하고 있는 자료구조
- 주어진 쿼리를 분할정복해서 풀이하는 대표적인 예
- [1, 8] 쿼리는 [1, 4] 쿼리와 [5, 8] 쿼리를 합성해서 답을 만들어낼 수 있다
- 연산은 결합법칙이 성립해야 하고, 항등원을 적절히 설정해야 한다

Segment Tree with Lazy Propagation

- Lazy Propagation: 게으르게 전파
- 업데이트 쿼리가 들어올 경우, 구간에 해당하는 노드에 바로 업데이트하지 않는다
- 추가적인 $lazy$ 배열을 활용한다

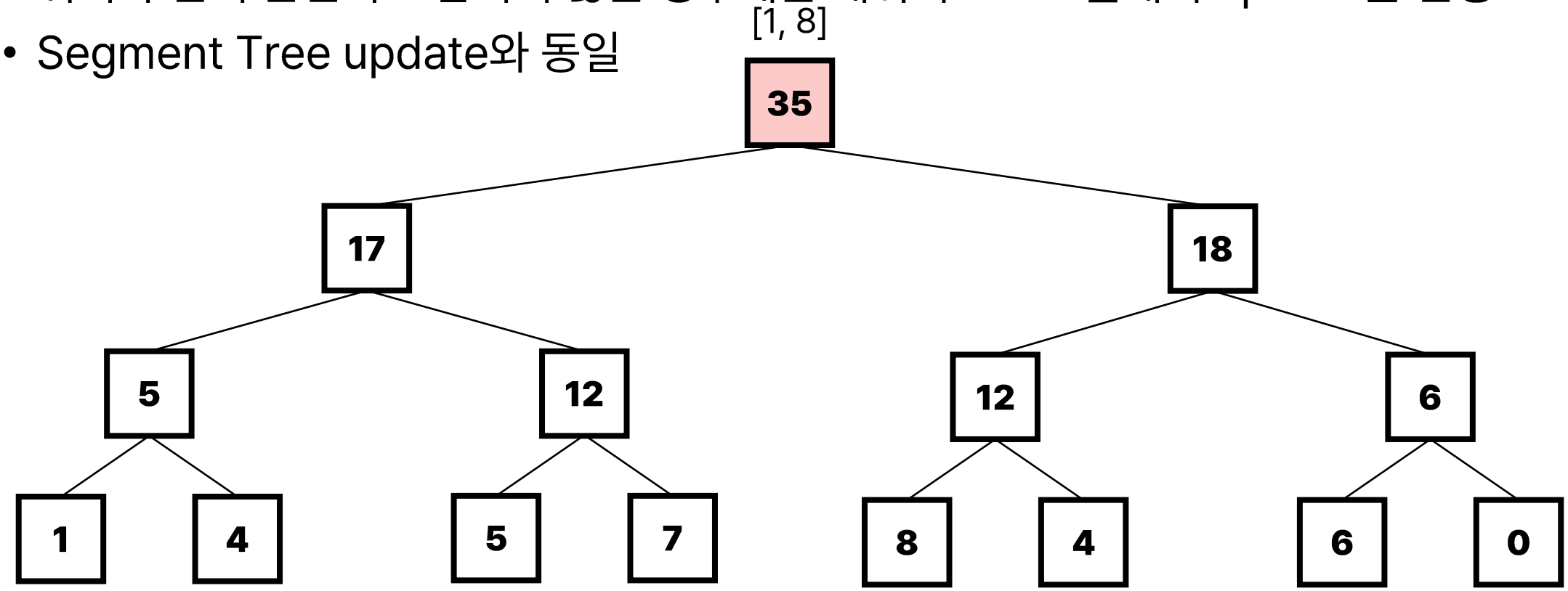
Update

- (분할 정복) 구간에 포함될 때까지 재귀적으로 탐색한다
- [3, 5] 구간에 1을 더하는 연산



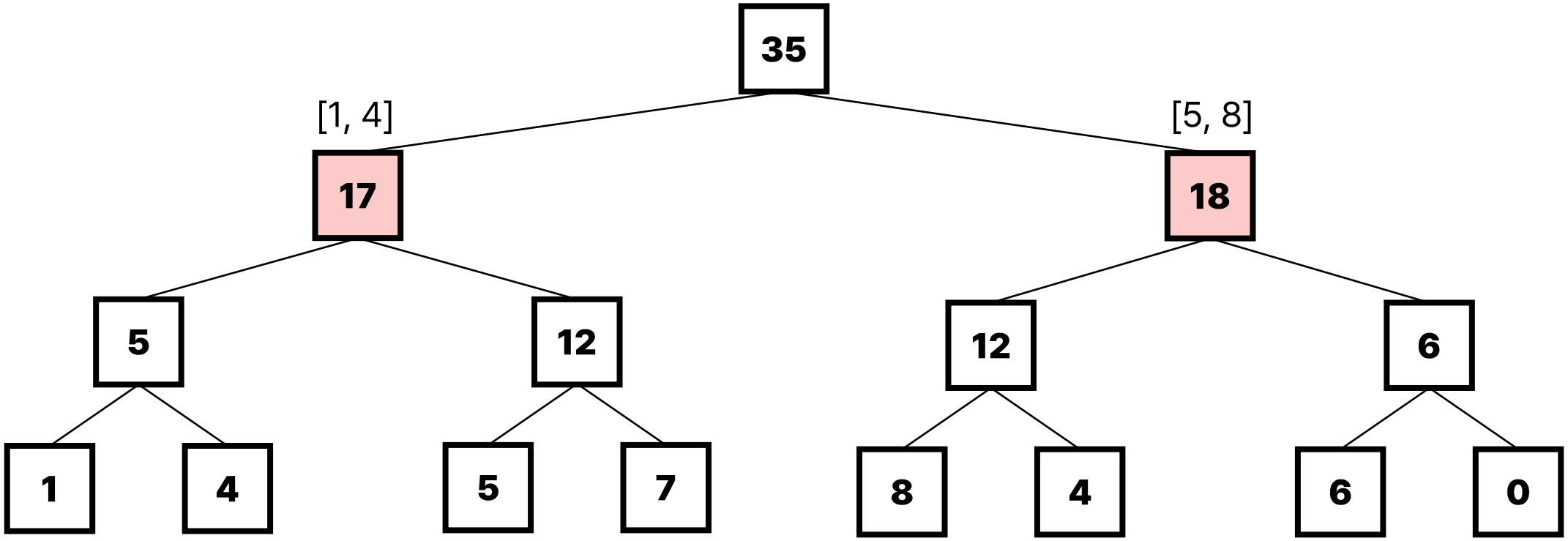
Update

- [3, 5] 구간에 1을 더하는 연산
- 쿼리 구간이 완전히 포함되지 않는 경우에는 재귀적으로 호출해서 update를 진행
- Segment Tree update와 동일



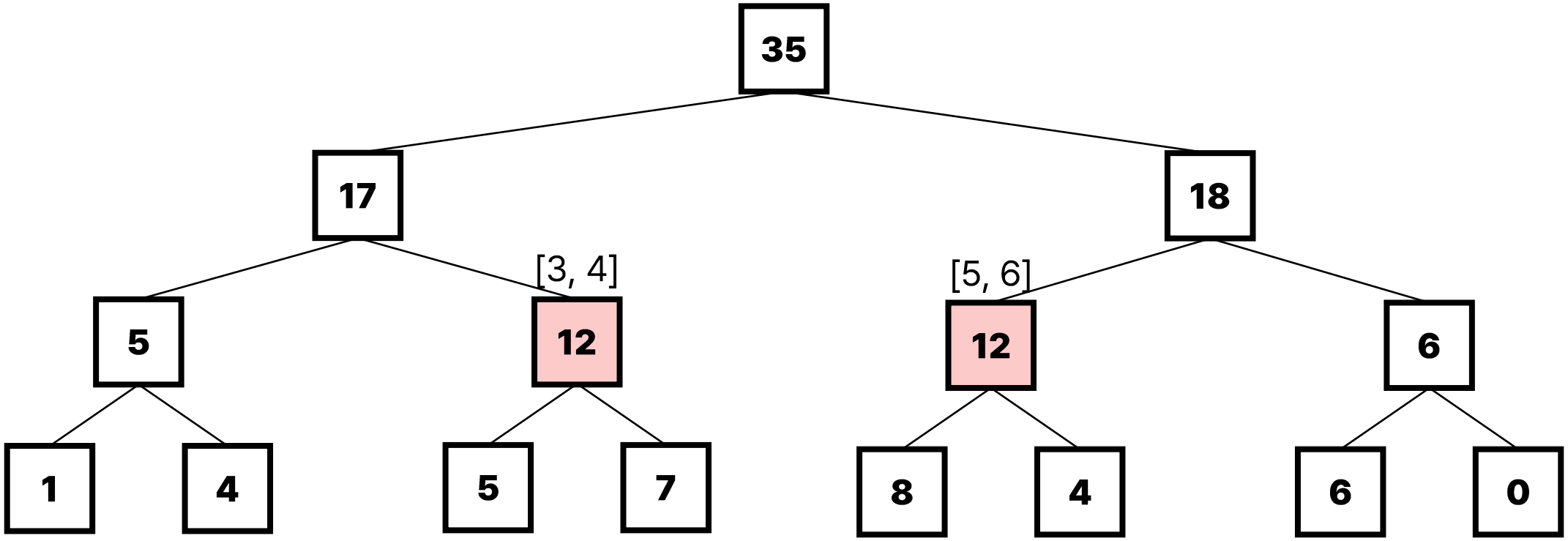
Update

- [3, 5] 구간에 1을 더하는 연산



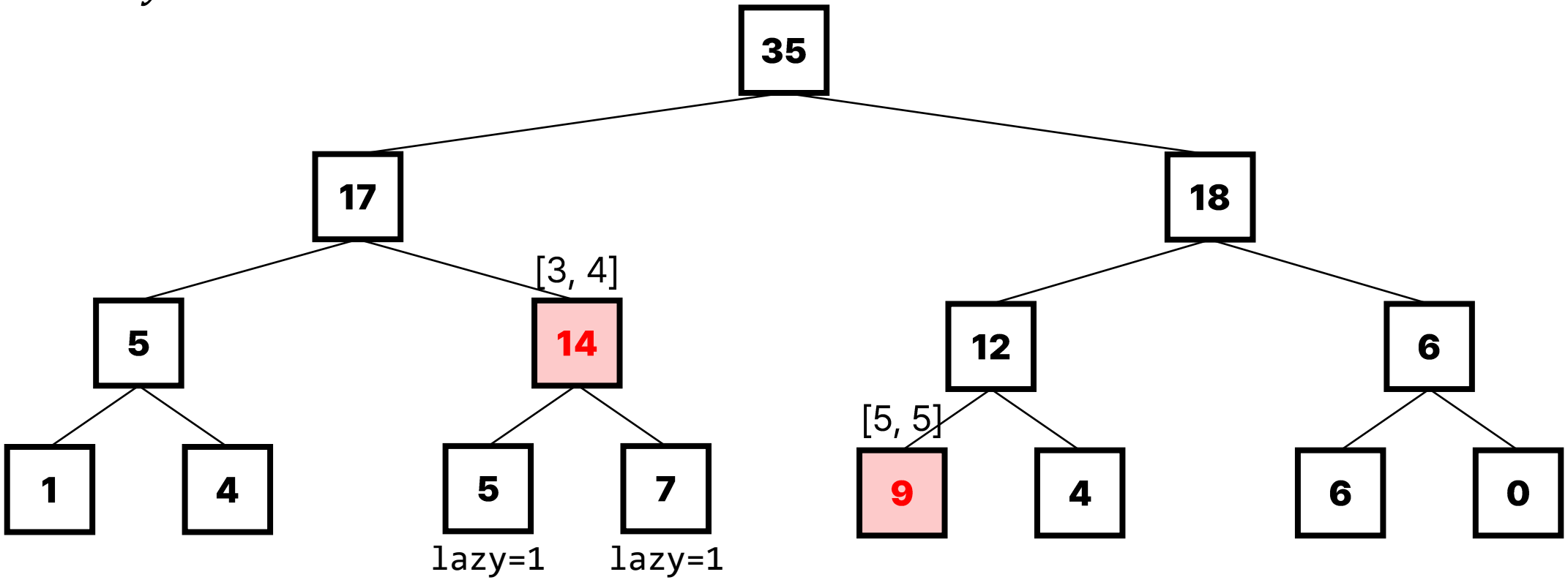
Update

- [3, 5] 구간에 1을 더하는 연산



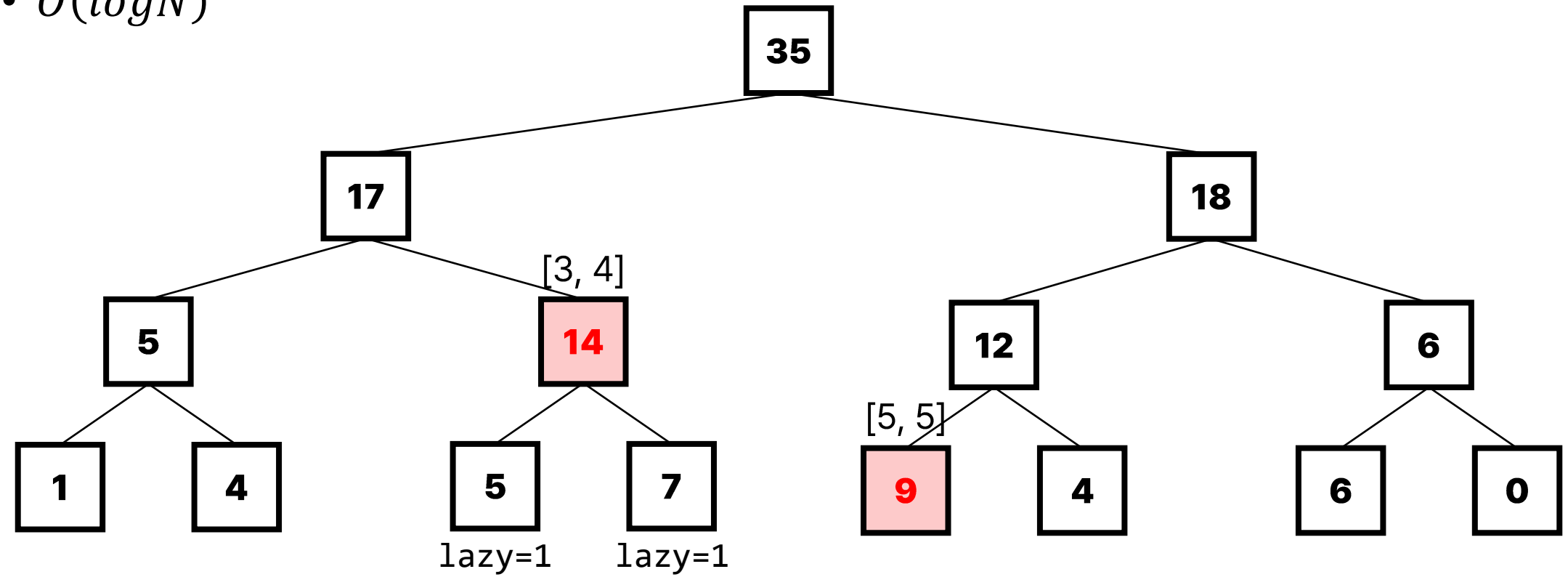
Update

- [3, 5] 구간에 1을 더하는 연산
- 구간에 포함된다면, 해당 노드의 값을 업데이트한 뒤 자식들은 나중에 업데이트한다는 의미로 $lazy$값을 전파



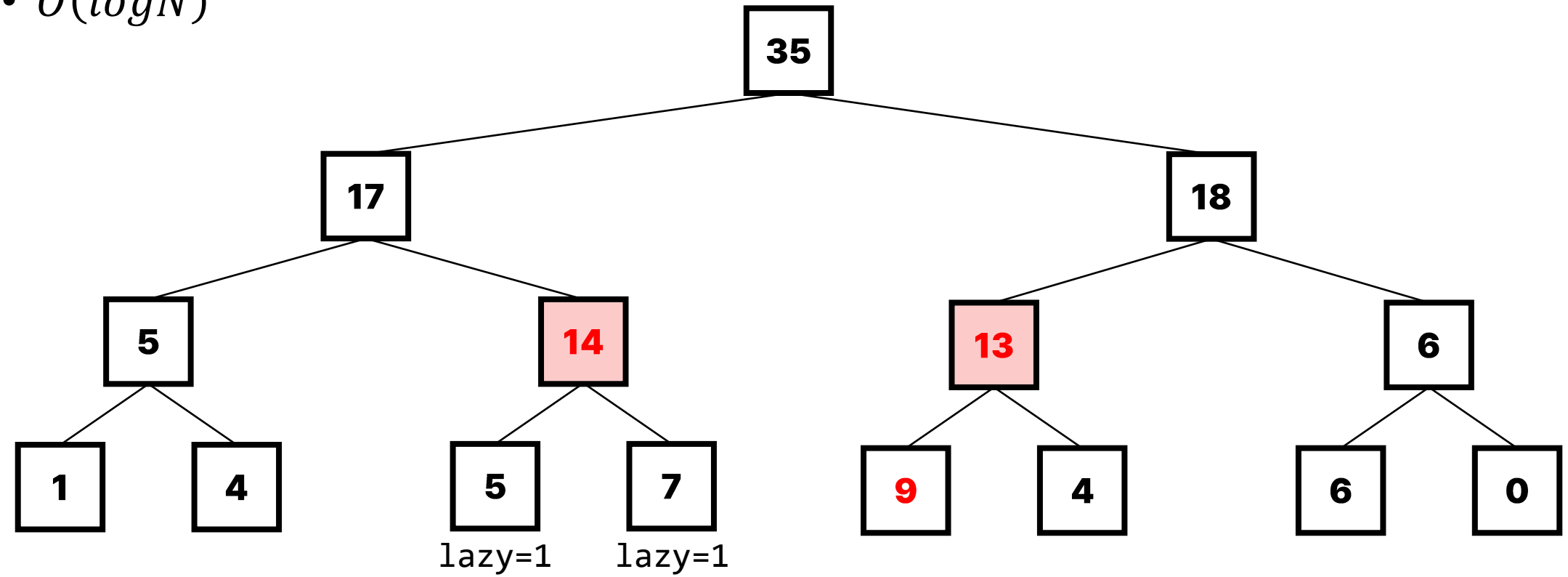
Update

- 업데이트가 자료구조에 곧바로 적용될 필요가 없으므로 이와 같이 최적화할 수 있다
- 추가적인 $lazy$ 배열 공간이 필요하지만, 시간복잡도는 세그먼트 트리의 업데이트와 동일
- $O(logN)$



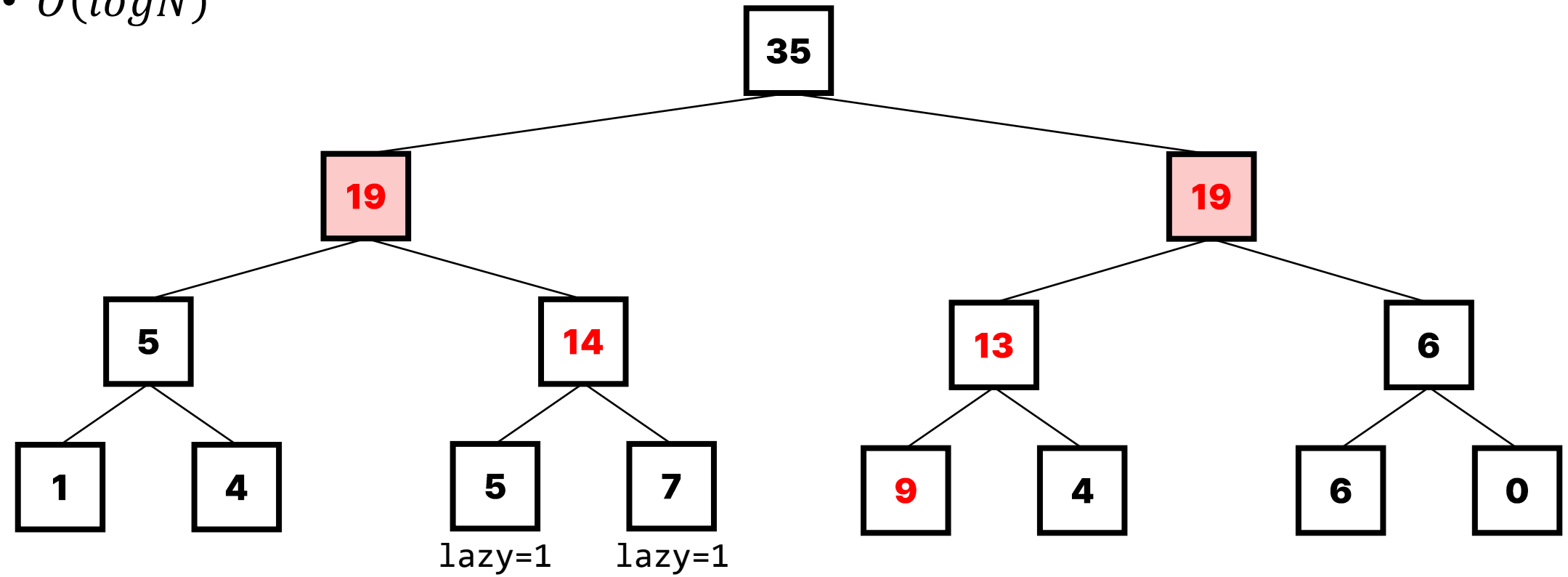
Update

- 업데이트가 자료구조에 곧바로 적용될 필요가 없으므로 이와 같이 최적화할 수 있다
- 추가적인 $lazy$ 배열 공간이 필요하지만, 시간복잡도는 세그먼트 트리의 업데이트와 동일
- $O(logN)$



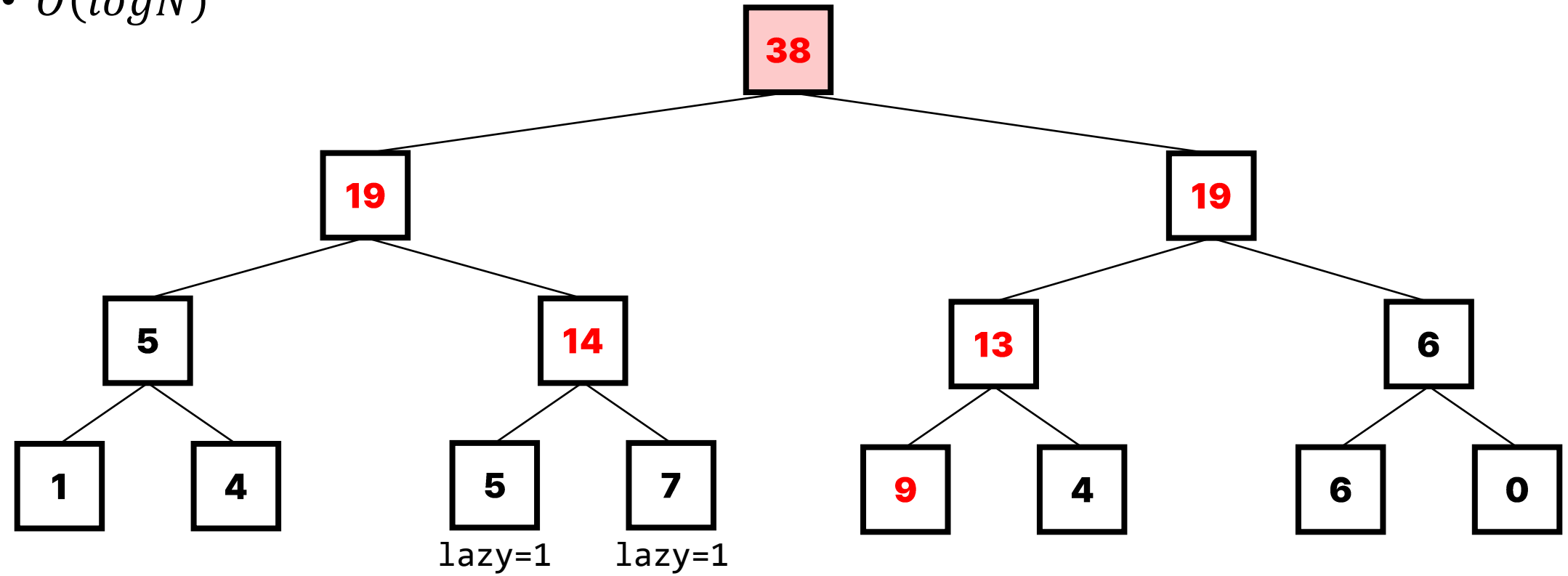
Update

- 업데이트가 자료구조에 곧바로 적용될 필요가 없으므로 이와 같이 최적화할 수 있다
- 추가적인 $lazy$ 배열 공간이 필요하지만, 시간복잡도는 세그먼트 트리의 업데이트와 동일
- $O(logN)$



Update

- 업데이트가 자료구조에 곧바로 적용될 필요가 없으므로 이와 같이 최적화할 수 있다
- 추가적인 $lazy$ 배열 공간이 필요하지만, 시간복잡도는 세그먼트 트리의 업데이트와 동일
- $O(logN)$

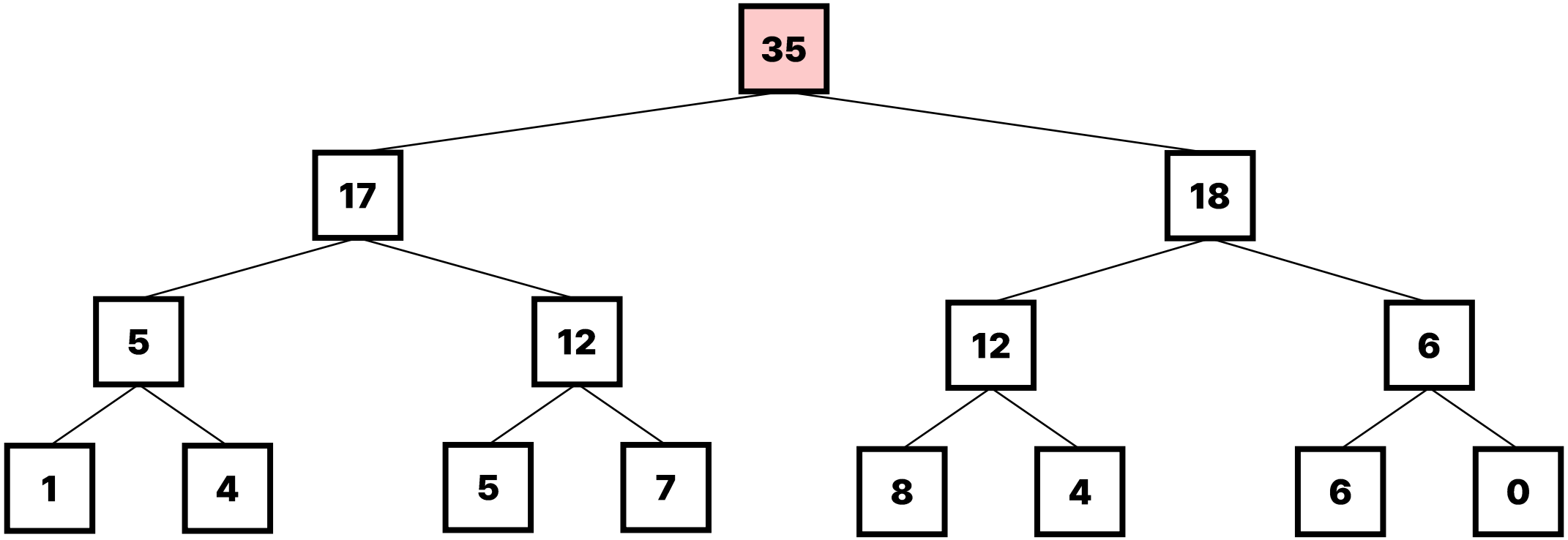


Update

- 기본적인 세그먼트 트리와 같이 업데이트
- 해당 노드의 $lazy$값이 존재할 경우에는 자식에게 전파하는 것을 우선
- 노드의 범위가 쿼리에 포함되는 경우에는 노드를 업데이트한 뒤 자식에게 $lazy$ 업데이트
- 연산에 따라 $lazy$를 노드에 적용하는 방법이 달라질 수 있음

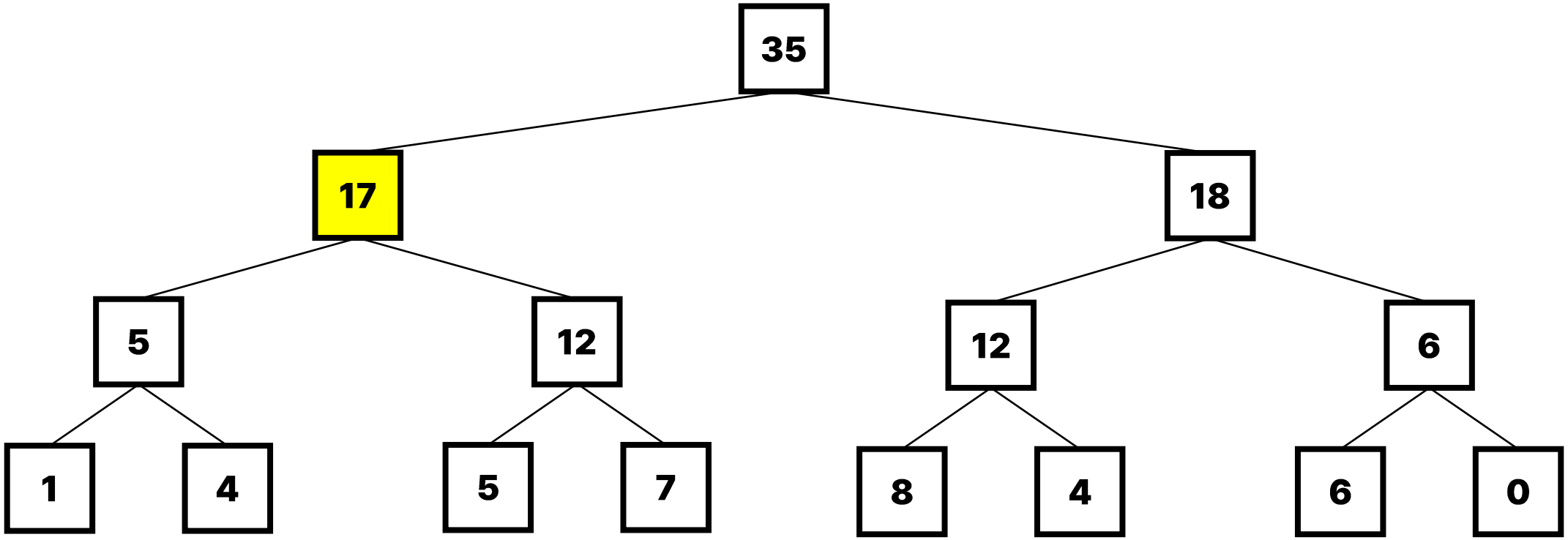
Update

- [1, 4] 구간에 5를 더한다고 하자



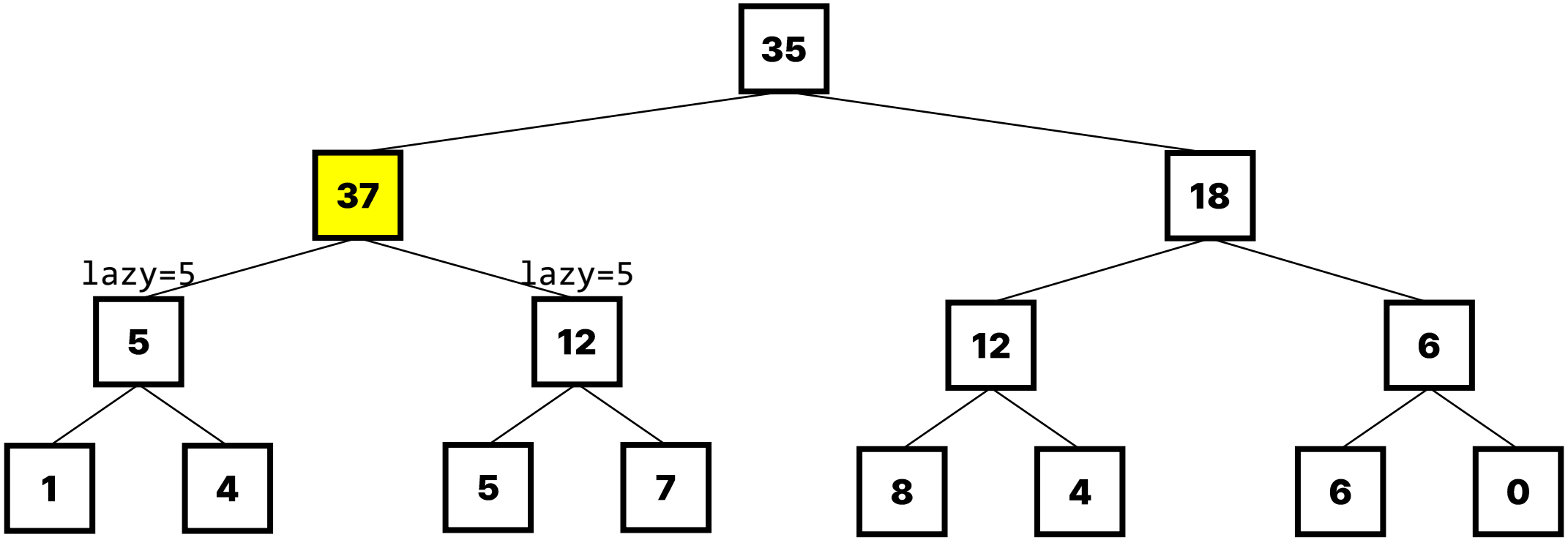
Update

- [1, 4] 구간에 5를 더한다고 하자
- 구간 합 쿼리의 경우, 해당 구간의 길이가 $lazy$를 업데이트하는 데 영향을 준다



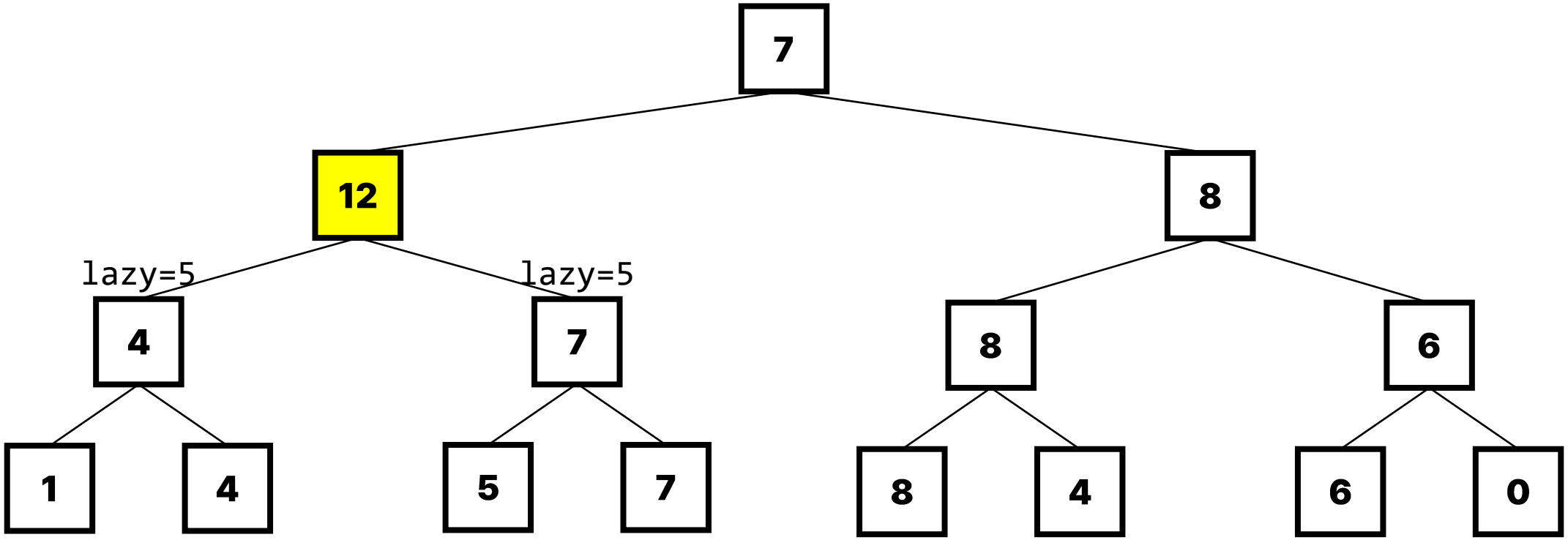
Update

- [1, 4] 구간에 5를 더한다고 하자
- 구간 합 쿼리의 경우, 해당 구간의 길이가 $lazy$를 업데이트하는 데 영향을 준다



Update

- [1, 4] 구간에 5를 더한다고 하자
- 구간 최댓값 쿼리의 경우, 해당 구간의 길이가 $lazy$를 업데이트하는 데 영향을 주지 않는다



Update

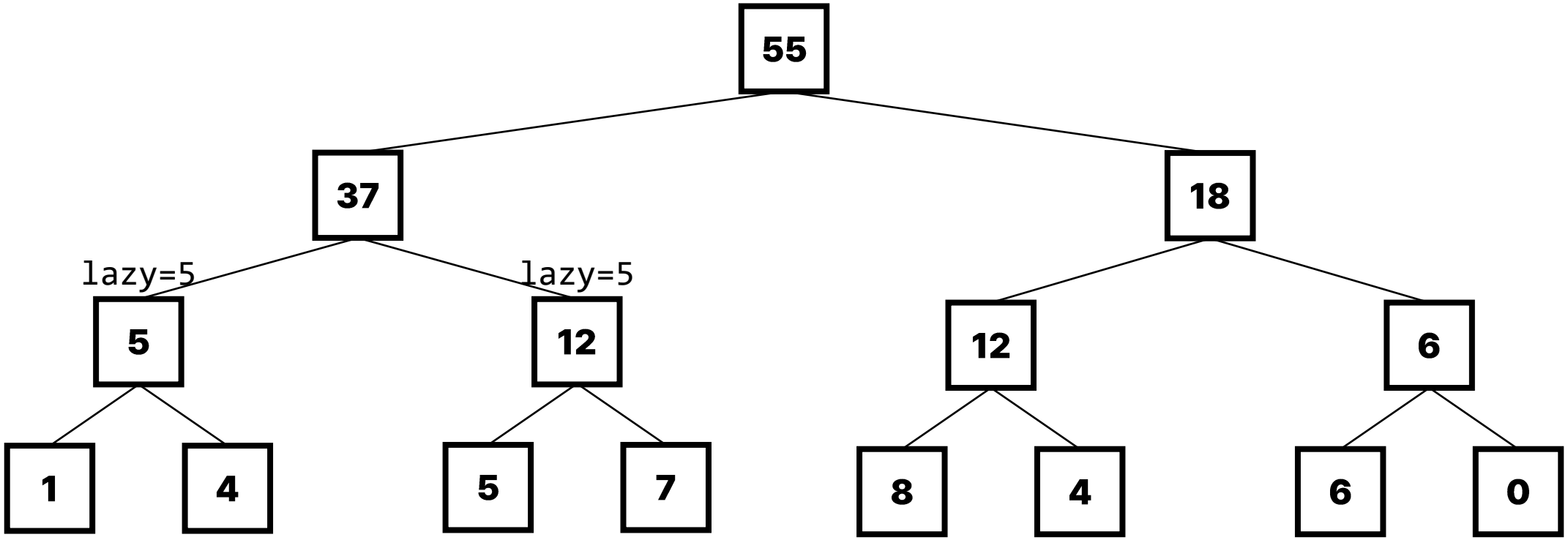
```
void update(int node, int s, int e, int l, int r, ll value){
    push(node, s, e);
    if (r < s || e < l) return;
    if (l <= s && e <= r){
        lazy[node] += value;
        push(node, s, e);
        return;
    }
    int mid = (s + e) / 2;
    update(node*2, s, mid, l, r, value);
    update(node*2+1, mid+1, e, l, r, value);
    tree[node] = f(tree[node*2], tree[node*2+1]);
}
```

Update

```
void push(int node, int s, int e){
    if (!lazy[node]) return;
    tree[node] += lazy[node] * (e-s+1);
    // tree[node] += lazy[node] when max query
    if (s != e){
        lazy[node*2] += lazy[node];
        lazy[node*2+1] += lazy[node];
    }
    lazy[node] = 0;
}
```

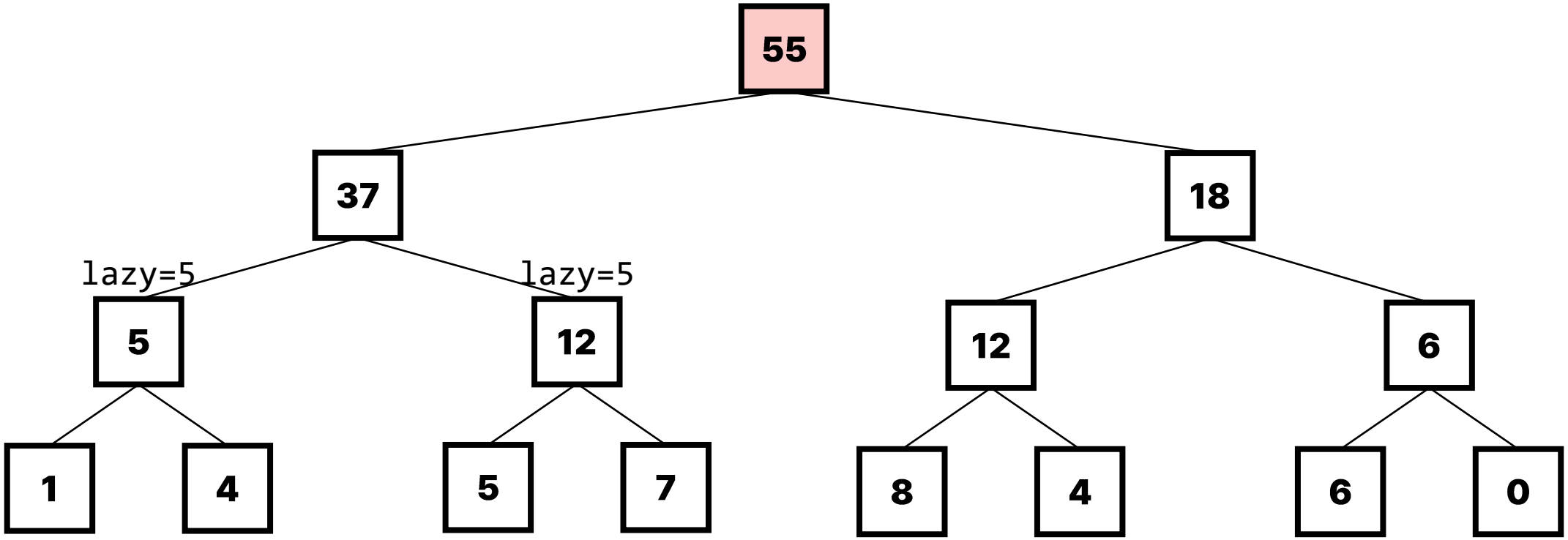
Query

- Update와 같은 방식으로 진행
- 기본 세그먼트 트리의 쿼리 함수에 lazy를 확인하는 것만 추가



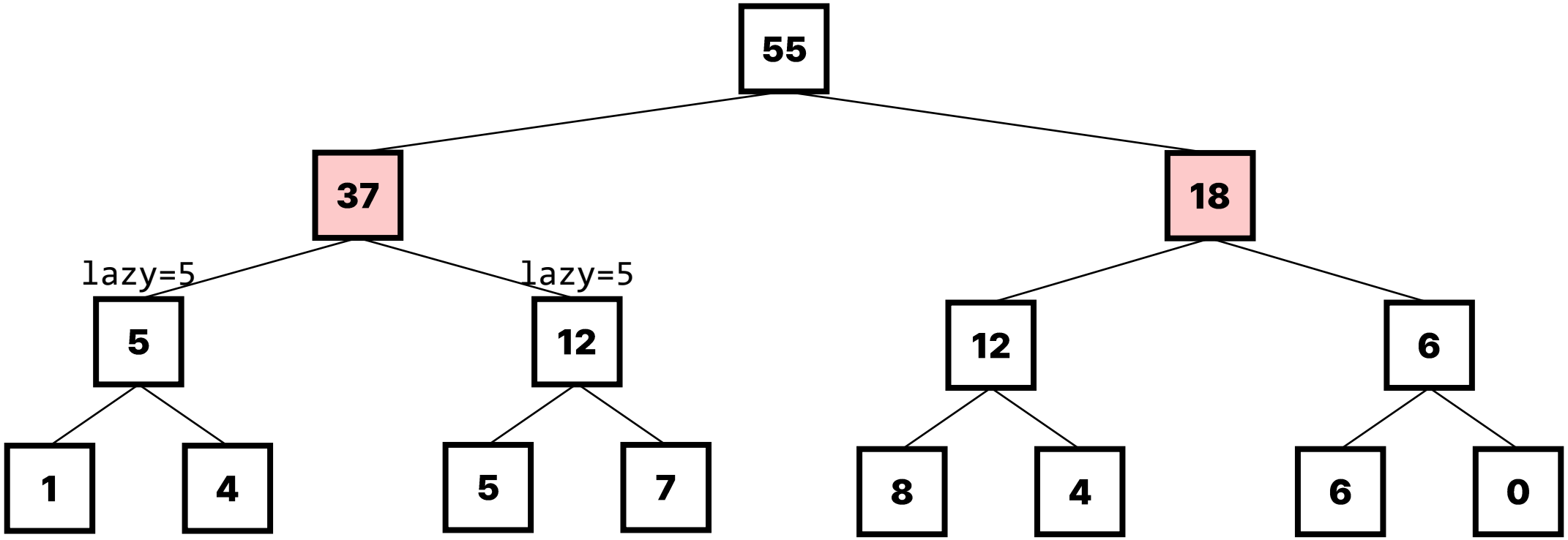
Query

- Update와 같은 방식으로 진행
- [2, 5]의 합을 구해 보자



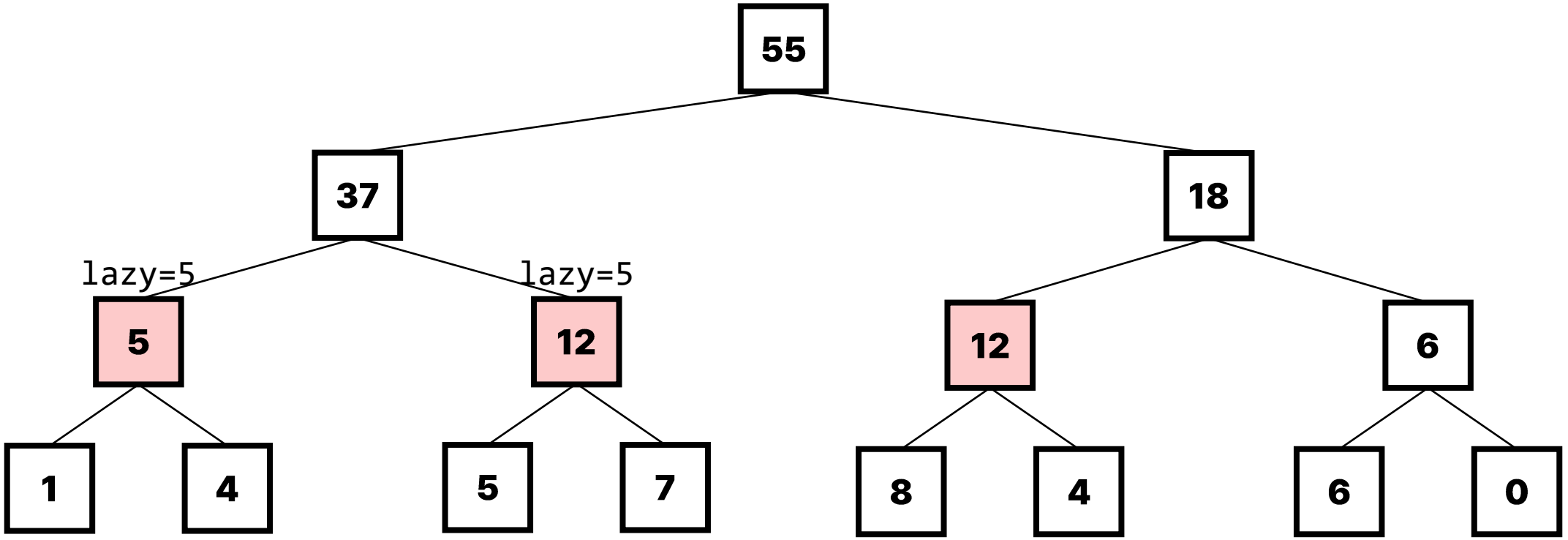
Query

- Update와 같은 방식으로 진행
- [2, 5]의 합을 구해 보자



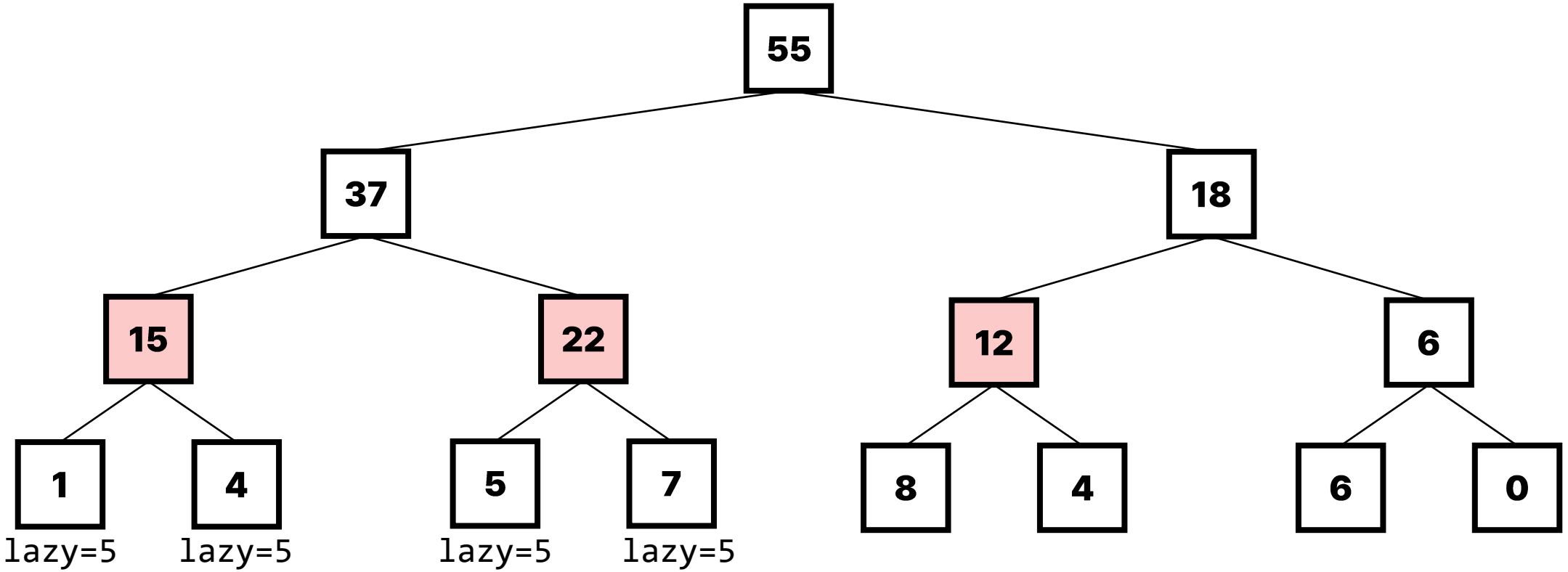
Query

- Update와 같은 방식으로 진행
- [2, 5]의 합을 구해 보자



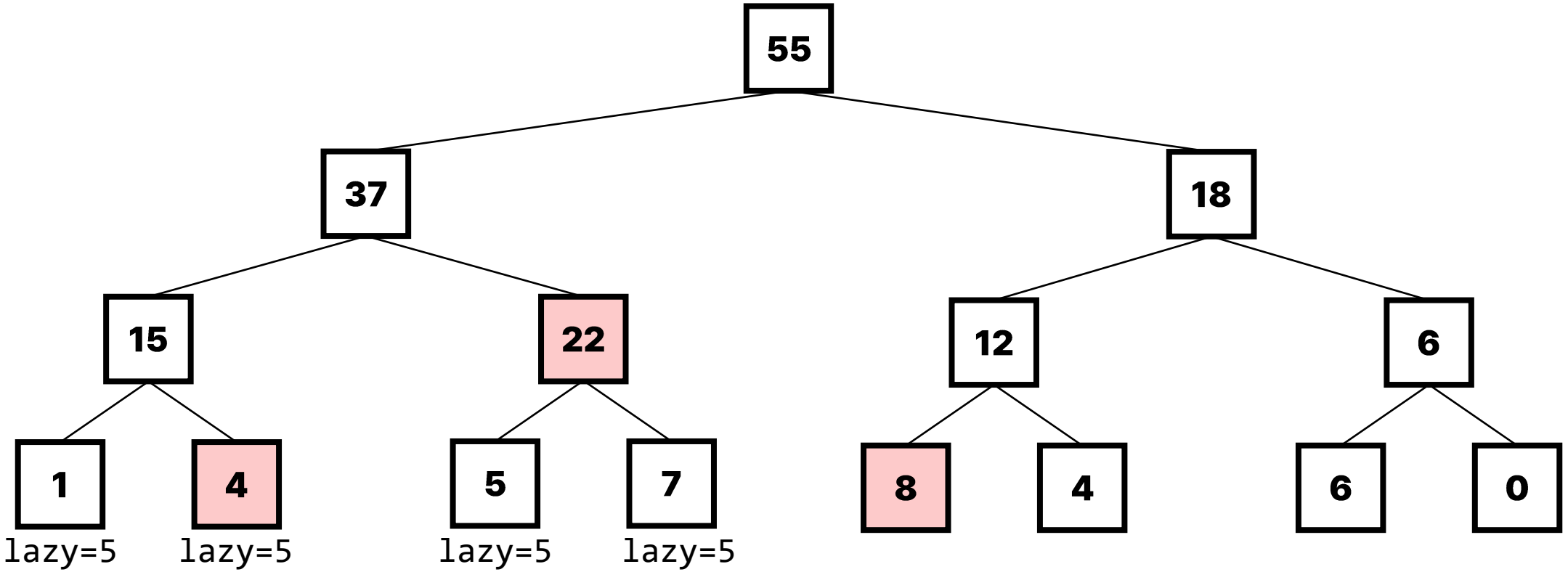
Query

- [2, 5]의 합을 구해 보자
- $lazy$가 존재하는 두 개의 노드는 우선 자신에게 $lazy$를 적용한 뒤, 자식에게 전파한다



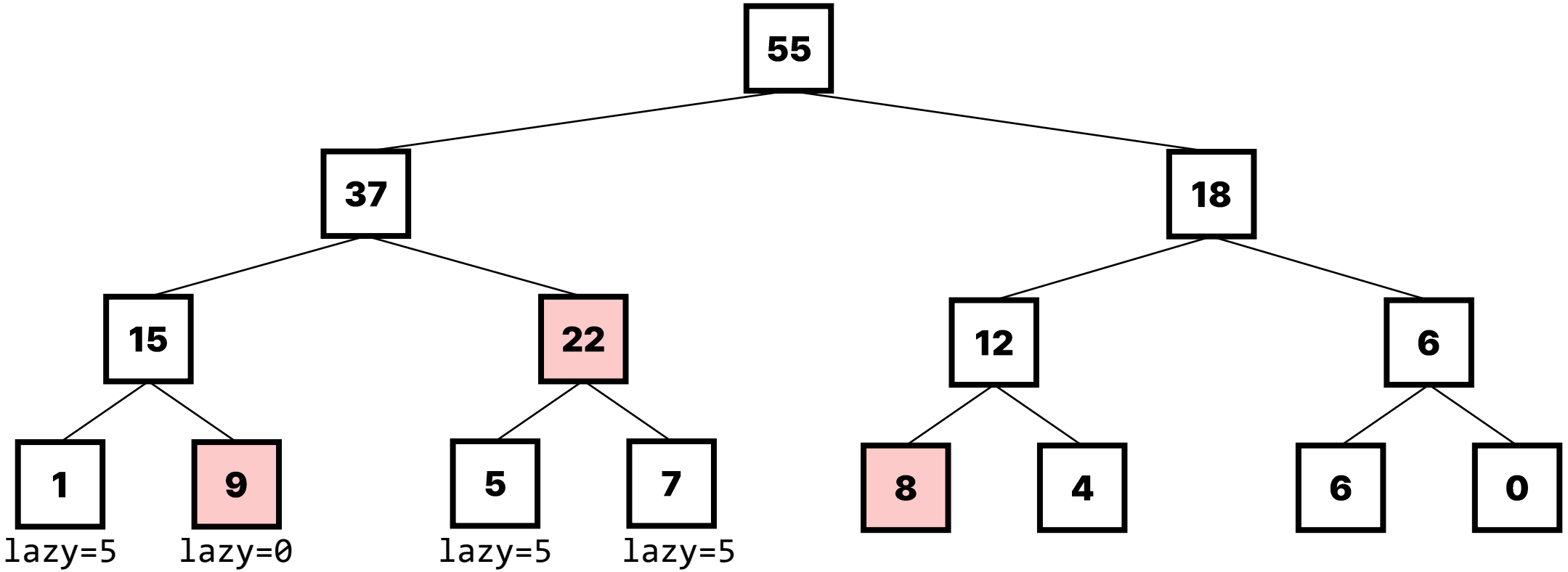
Query

- [2, 5]의 합을 구해 보자
- 구간에 포함되지 않는 경우 재귀적으로 호출



Query

- [2, 5]의 합을 구해 보자
- 구간에 포함되지 않는 경우 재귀적으로 호출



Query

```
ll query(int node, int s, int e, int l, int r){
    push(node, s, e);
    if (r < s || e < l) return default_value;
    if (l <= s && e <= r) return tree[node];
    int mid = (s + e) / 2;
    ll n1 = query(node*2, s, mid, l, r);
    ll n2 = query(node*2+1, mid+1, e, l, r);
    return f(n1, n2);
}
```

XOR BOJ 12844

- 길이 N인 배열 arr이 주어진다
- l, r $(1 \leq l, r \leq N)$이 주어졌을 때, 다음 두 가지 연산 중 하나를 Q번 수행한다.
- arr[l] $\oplus$ arr[l+1] $\oplus$ ... $\oplus$ arr[r-1] $\oplus$ arr[r] 의 값을 구하기
- arr[l..r]의 각각의 원소에 정수 k를 XOR

XOR BOJ 12844

- Segment Tree를 사용하기 전에 확인해야 할 것
- 결합법칙이 성립하는가?
- 항등원을 구할 수 있는가?

- Lazy Propagation에서 신경써야 할 것
- Lazy값을 어떻게 구간 노드에 반영할 것인가?

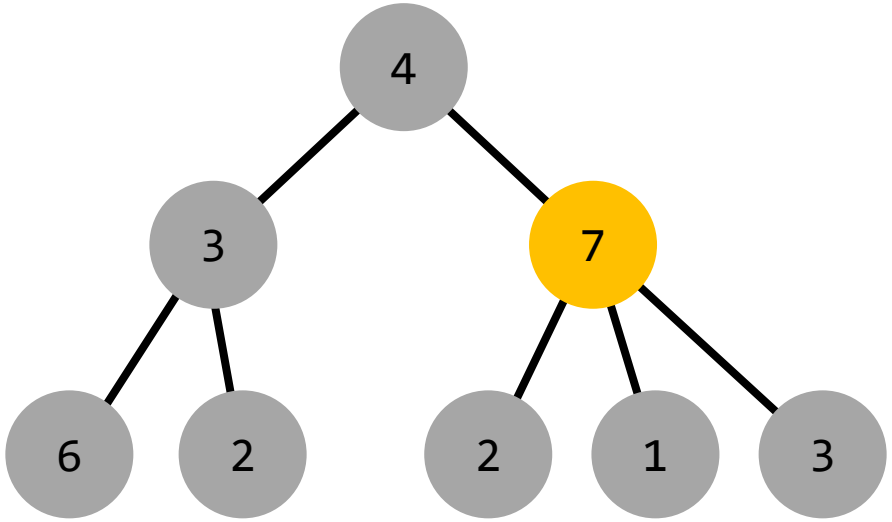
XOR BOJ 12844

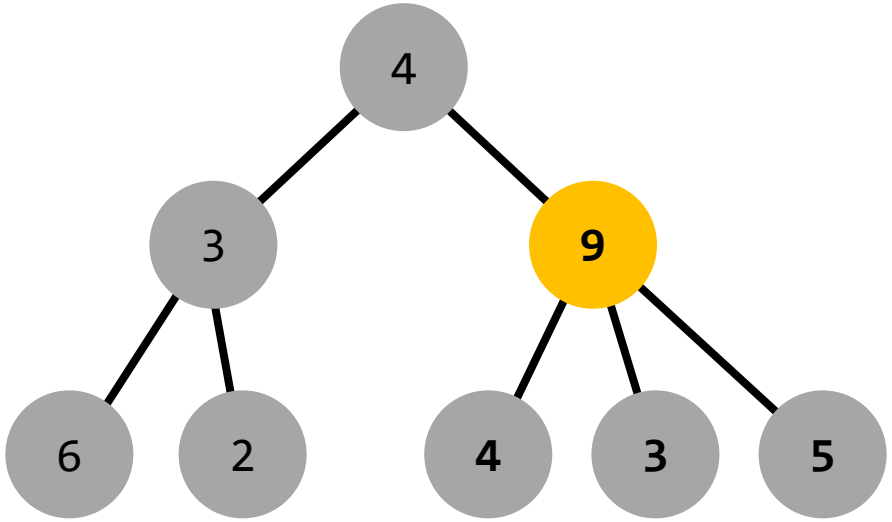
- Segment Tree를 사용하기 전에 확인해야 할 것
- 결합법칙이 성립하는가? **YES** $(A \oplus B) \oplus C = A \oplus (B \oplus C)$
- 항등원을 구할 수 있는가? **YES** $0 \oplus x = x$

- Lazy Propagation에서 신경써야 할 것
- Lazy값을 어떻게 구간 노드에 반영할 것인가?

회사 문화 2 BOJ 14268

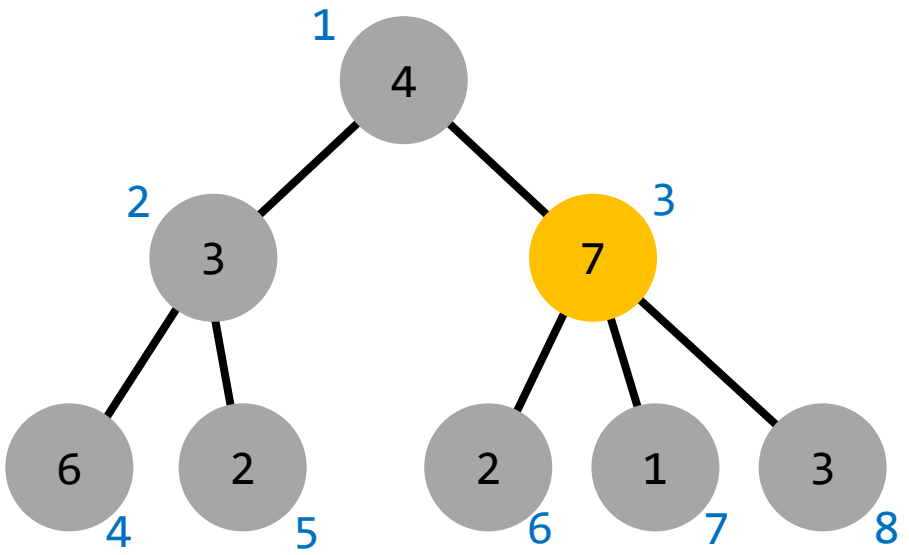
- 노드 개수가 N인 트리가 주어진다. 노드에는 값을 저장하고 아래 두 가지 쿼리를 수행
- 1. i번 노드를 포함하는 모든 서브트리에 값을 k만큼 더하기
- 2. i번 노드의 값을 출력하기





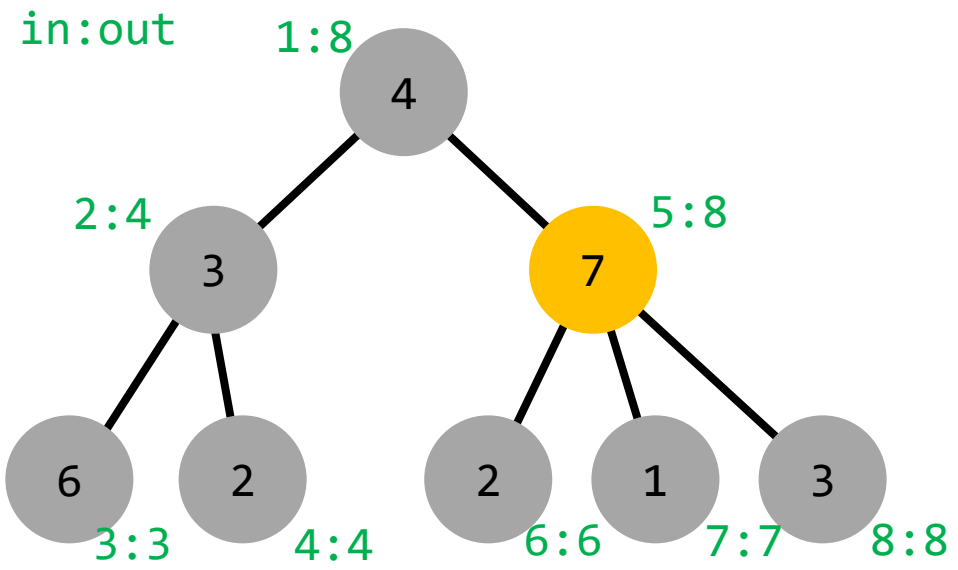
Euler Tour Technique

- 비선형 자료구조인 트리를 선형 자료구조인 배열에 매핑하는 테크닉
- Euler Tour 오일러 회로: 모든 간선을 최대 한 번 사용해 시작점으로 돌아오는 경로
- 양방향 간선을 두 개의 단방향 간선으로 생각하고 오일러 회로를 구해보자
- 1 2 4 4 2 5 5 2 1 3 6 6 3 7 7 3 8 8 3 1



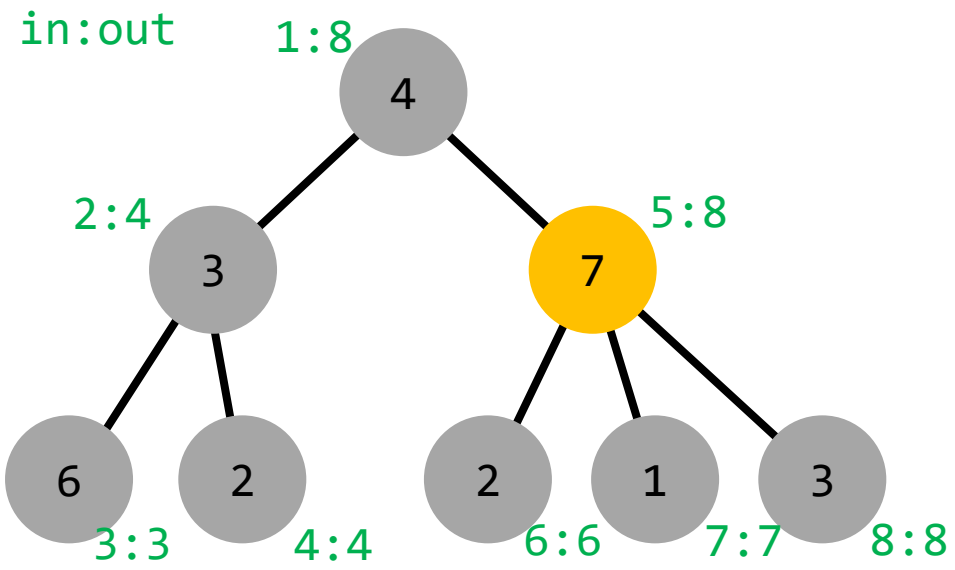
Euler Tour Technique

- **1 2 4 4** 2 **5 5** 2 1 **3 6 6** 3 **7 7** 3 **8 8 3 1**
- 특정 노드의 자식은 반드시 **처음 등장한 시점**과 **마지막으로 등장한 시점** 사이에 존재
- 트리의 구조를 연속된 배열의 구간으로 매핑할 수 있음
- 서브트리 또한 배열의 구간으로 나타낼 수 있다



Euler Tour Technique

- 만약 노란색 노드를 포함하는 서브트리에 모두 2를 더한다면
- 오일러 투어를 통해 만들어진 **배열의 연속된 구간에 2를 더하는 것**과 같다
- Segment Tree with Lazy Propagation 활용 가능
- Segment Tree 이외에도 누적 합, LCA와 같이 활용할 수 있다



4	3	6	2	7	2	1	3

4	3	6	2	9	4	3	5

하늘에서 떨어지는 1, 2, …, R-L+1개의 별 BOJ 17353

- 길이 N인 배열 arr이 주어진다
- l, r 또는 i $(1 \leq l, r, i \leq N)$ 가 주어졌을 때, 두 가지 연산 중 하나를 Q번 수행한다.
- arr[i]의 값을 구하기
- arr[l]에 1, arr[2]에 2, …, arr[r]에 R-L+1 더하기

- 참신한 아이디어 문제, 더할 때의 규칙을 잘 보고 어떻게 적용할지 생각해 보자

연습 문제

10999 : 구간 합 구하기 2
12844 : XOR
10277 : JuQueen
17353 : 하늘에서 떨어지는 1, 2, ..., R-L+1개의 별

14268 : 회사 문화 2
16404 : 주식회사 승범이네
18437 : 회사 문화 5

Reference

- https://book.acmicpc.net/ds/segment-tree-lazy-propagation
- https://teferi.net/ps/%EA%B5%AC%EA%B0%84_%EC%BF%BC%EB%A6%AC#lazy_propagation
- https://www.acmicpc.net/blog/view/117
- https://mathsciforstudent.tistory.com/182
- https://www.acmicpc.net/blog/view/26