

RSQ 구현의 최소값
 찾기

RSQ

최백준 choi@startlink.io

누적합

누적합

Prefix Sum

3

- 수열 $A[1], A[2], \dots, A[N]$ 이 있을 때
- $A[i] + \dots + A[j]$ 를 구하는 문제

$$A[1] \sim A[N]$$

$$S[i] = \sum_{k=1}^i A[k] \quad (1 \leq i \leq N)$$

$$O(1)$$

- $S[i] = A[1] + A[2] + \dots + A[i]$

$$A[i] + \dots + A[i]$$

$$S[i] = A[1] + \dots + A[i]$$

$$- S[i-1] = A[1] + \dots + A[i-1]$$

$$S[i] - S[i-1] = A[i]$$

누적합

Prefix Sum

- $A[i] + \dots + A[j]$ 를 구하는 문제
- $S[j] = A[1] + A[2] + \dots + A[i-1] + A[i] + \dots + A[j]$
- $S[i-1] = A[1] + A[2] + \dots + A[i-1]$
- $S[j] - S[i-1] = A[i] + \dots + A[j]$

공간: $O(N)$
선처리: $O(N)$
쿼리: $O(1)$
반경: $O(N)$

구간 합 구하기 4

5

<https://www.acmicpc.net/problem/11659>

- 수 N 개가 주어졌을 때, i 번째 수부터 j 번째 수까지 합을 구하는 문제

구간 합 구하기 4

<https://www.acmicpc.net/problem/11659>

- C++: <https://gist.github.com/Baekjoon/2c28f410a9cae50d3632>

나머지 합

<https://www.acmicpc.net/problem/10986>

$O(N^2)$

- 수 N 개 $A[1], A[2], \dots, A[N]$ 이 주어진다.
- 연속된 부분 구간의 합이 M 으로 나누어 떨어지는 구간의 개수를 구하는 문제
- 즉, $A[i] + \dots + A[j]$ ($i \leq j$) 의 합이 M 으로 나누어 떨어지는 (i, j) 쌍의 개수를 구해야 한다.

$$\underbrace{(A[i] + \dots + A[j])}_{O(N)} \% M == 0$$

$O(N)$

나머지 합

<https://www.acmicpc.net/problem/10986>

- $S[i] = A[1] + \dots + A[i]$ 라고 하자
- $A[i] + \dots + A[j] = S[j] - S[i-1]$
- $(A[i] + \dots + A[j]) \% M = (S[j] - S[i-1]) \% M$
- $(A[i] + \dots + A[j]) \% M == 0$ 인 것의 개수를 구해야 한다
- $(S[j] - S[i-1]) \% M == 0$ 와 같다
- 나눈 나머지가 0이 되려면
- $S[j] \% M == S[i-1] \% M$ 이 되어야 한다

$$(A[1] + \dots + A[i]) \% M == 0$$

8

$$(S[j] - S[i-1]) \% M == 0$$

$$S[j] \% M - S[i-1] \% M == 0$$

$$S[j] \% M = S[i-1] \% M$$

상

$$Cnt[k] = \boxed{S[i] \% M == k}$$

인 것의 개수

$$\sum Cnt[k] C_2$$

나머지 합

<https://www.acmicpc.net/problem/10986>

- 이 문제는
- $S[j] \% M == S[i-1] \% M$ 이 되어야 한다
- 를 만족하는 (i, j) 쌍의 개수를 구하는 문제가 된다.
- $\text{cnt}[k]$ 를 $S[i] \% M == k$ 인 i 의 개수라고 하면
- $0 \leq k < M$ 인 k 에 대해서
- $\text{cnt}[k] * (\text{cnt}[k] - 1) / 2$ 의 합을 구하면 된다.

$$\overline{i} = j$$

$$S[j] \% M == \frac{S[\overline{i-1}] \% M}{\text{SCD}}$$

$$A[i] \sim$$

나머지 합

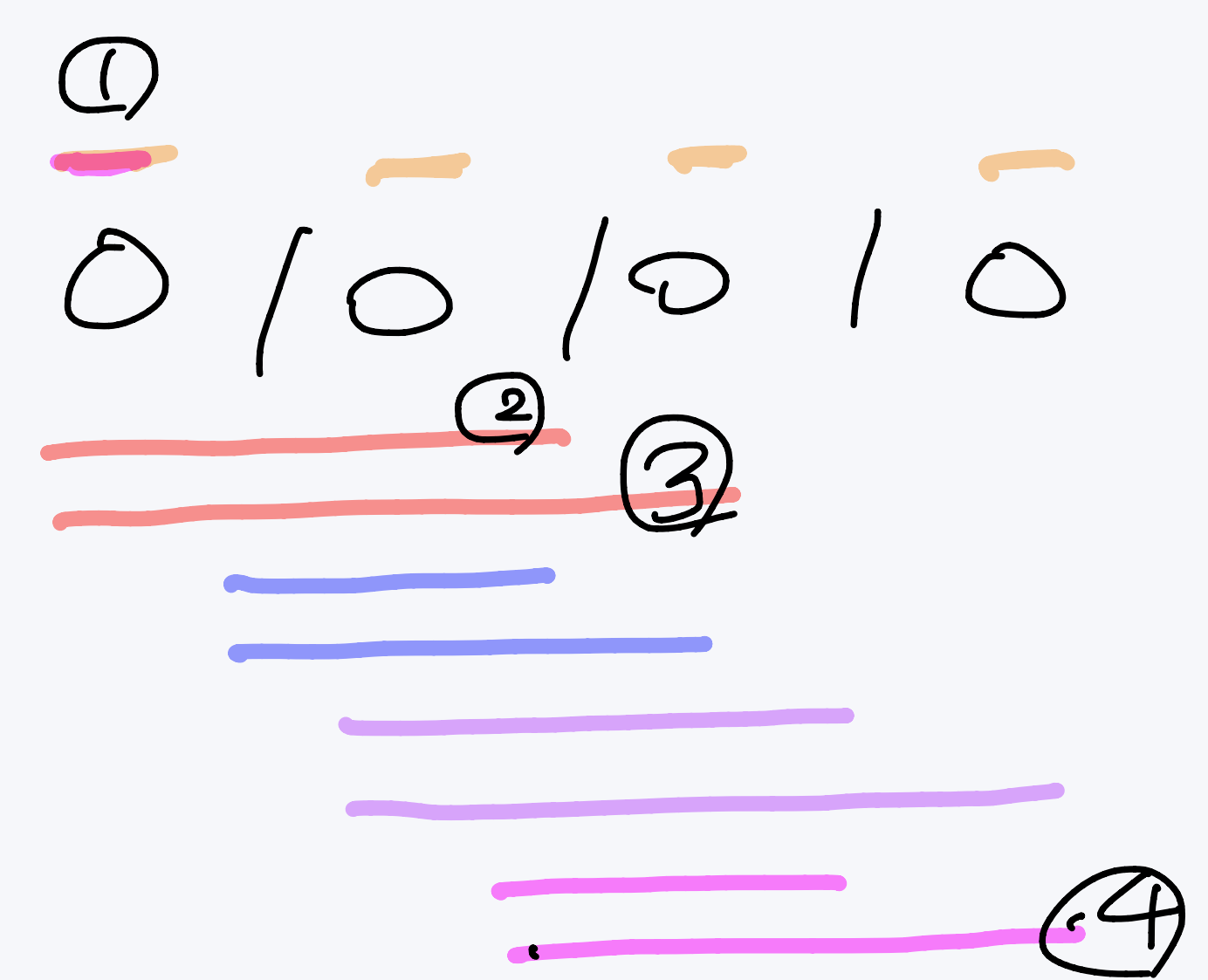
$J=5$
 $I-1=0$

$M=2$

<https://www.acmicpc.net/problem/10986> $1 \sim 5$

• C++: <https://gist.github.com/Baekjoon/6d45e5ef5d7eed6039b5>

	0	1	2	3	4	5	6	7
$A[i]$	0	0	1	0	1	0	1	0
$S[i]$	0	0	1	1	2	2	3	3
	0	0	1	1	0	0	1	1



i	0	1
$Cut[i]$	4	4

$J=1$
 $I-1=0$

$J=4$
 $I-1=0$

$4C_2 + 4C_2 = \frac{2 \times 3}{2} = 3$

$6 + 6 = 12$

2차원 누적합

구간 합 구하기 5

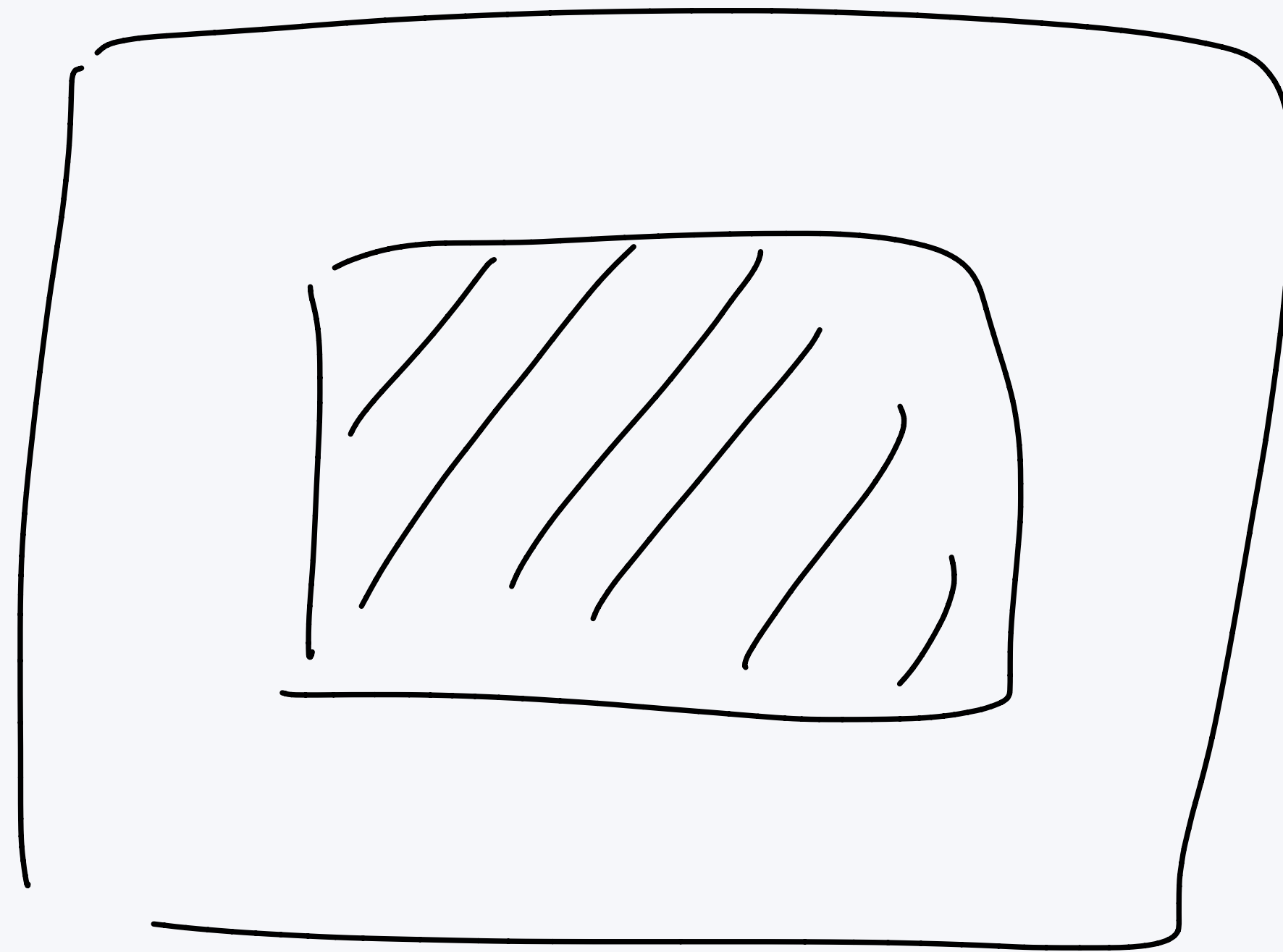
12

<https://www.acmicpc.net/problem/11660>

- 2차원 배열에서 왼쪽 윗 칸이 $(x1, y1)$, 오른쪽 아랫 칸이 $(x2, y2)$ 인 직사각형에 들어있는 수의 합을 구하는 문제

행 열

행 열



구간 합 구하기 5

<https://www.acmicpc.net/problem/11660>

- 합을 효율적으로 구하는 방법
- $S[i][j] = (1, 1) \sim (i, j)$ 까지 합
- $S[i][j] = S[i-1][j] + S[i][j-1] - S[i-1][j-1] + A[i][j]$

왼쪽
(1, 1)

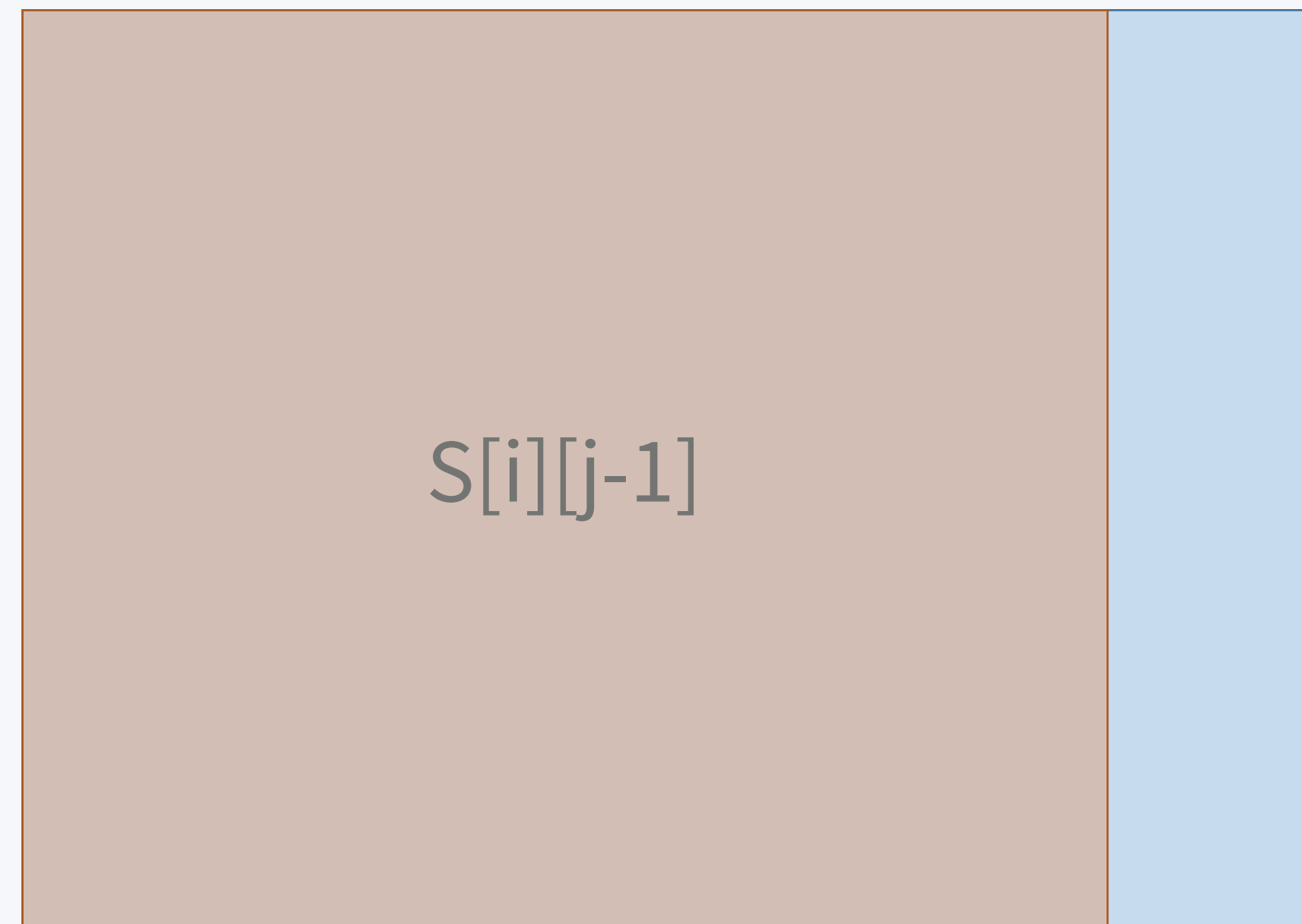
오른 아래
(i, j)

구간 합 구하기 5

14

<https://www.acmicpc.net/problem/11660>

- 합을 효율적으로 구하는 방법
- $S[i][j] = (1, 1) \sim (i, j)$ 까지 합
- $S[i][j] = S[i-1][j] + S[i][j-1] - S[i-1][j-1] + A[i][j]$

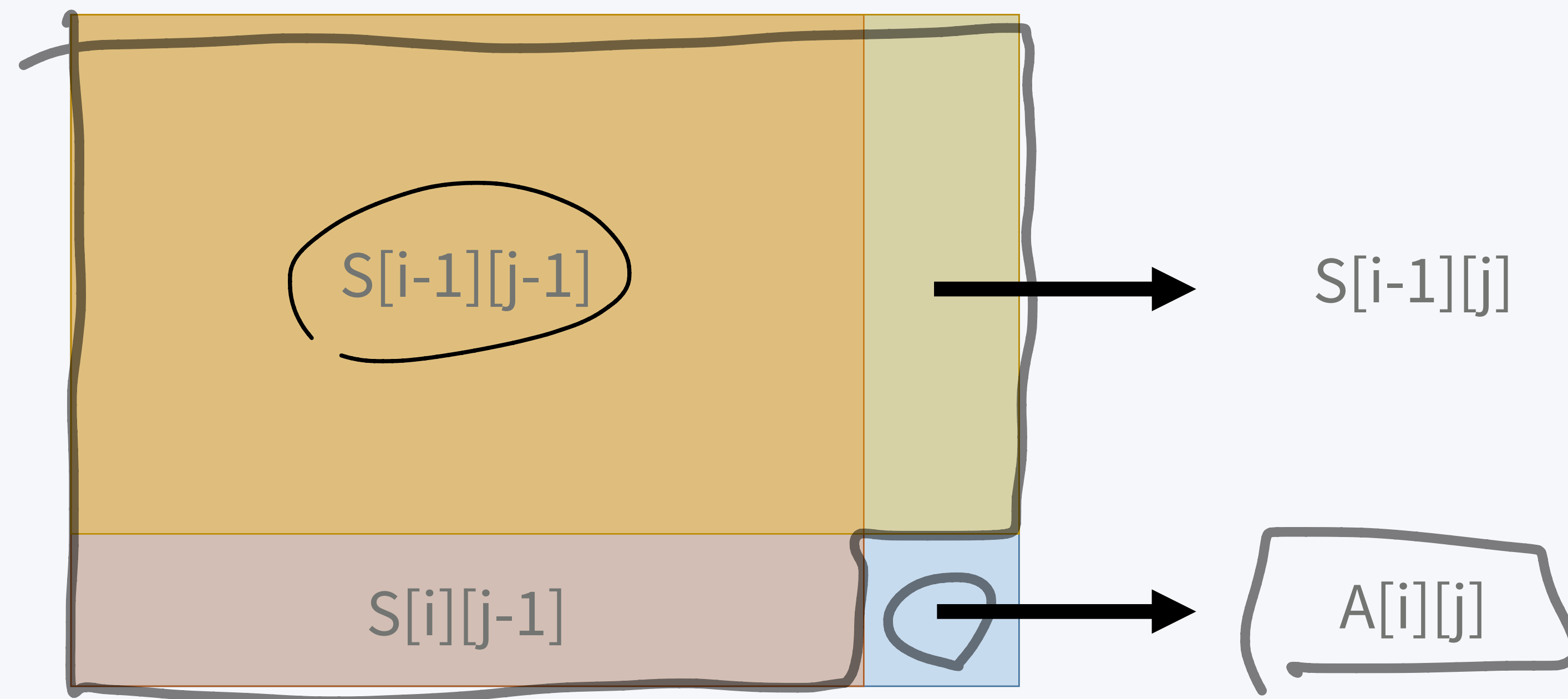


구간 합 구하기 5

15

<https://www.acmicpc.net/problem/11660>

- 합을 효율적으로 구하는 방법
- $S[i][j] = (1, 1) \sim (i, j)$ 까지 합
- $S[i][j] = S[i-1][j] + S[i][j-1] - S[i-1][j-1] + A[i][j]$



구간 합 구하기 5

16

<https://www.acmicpc.net/problem/11660>

- $(a,b) \sim (c,d)$ 합 구하기

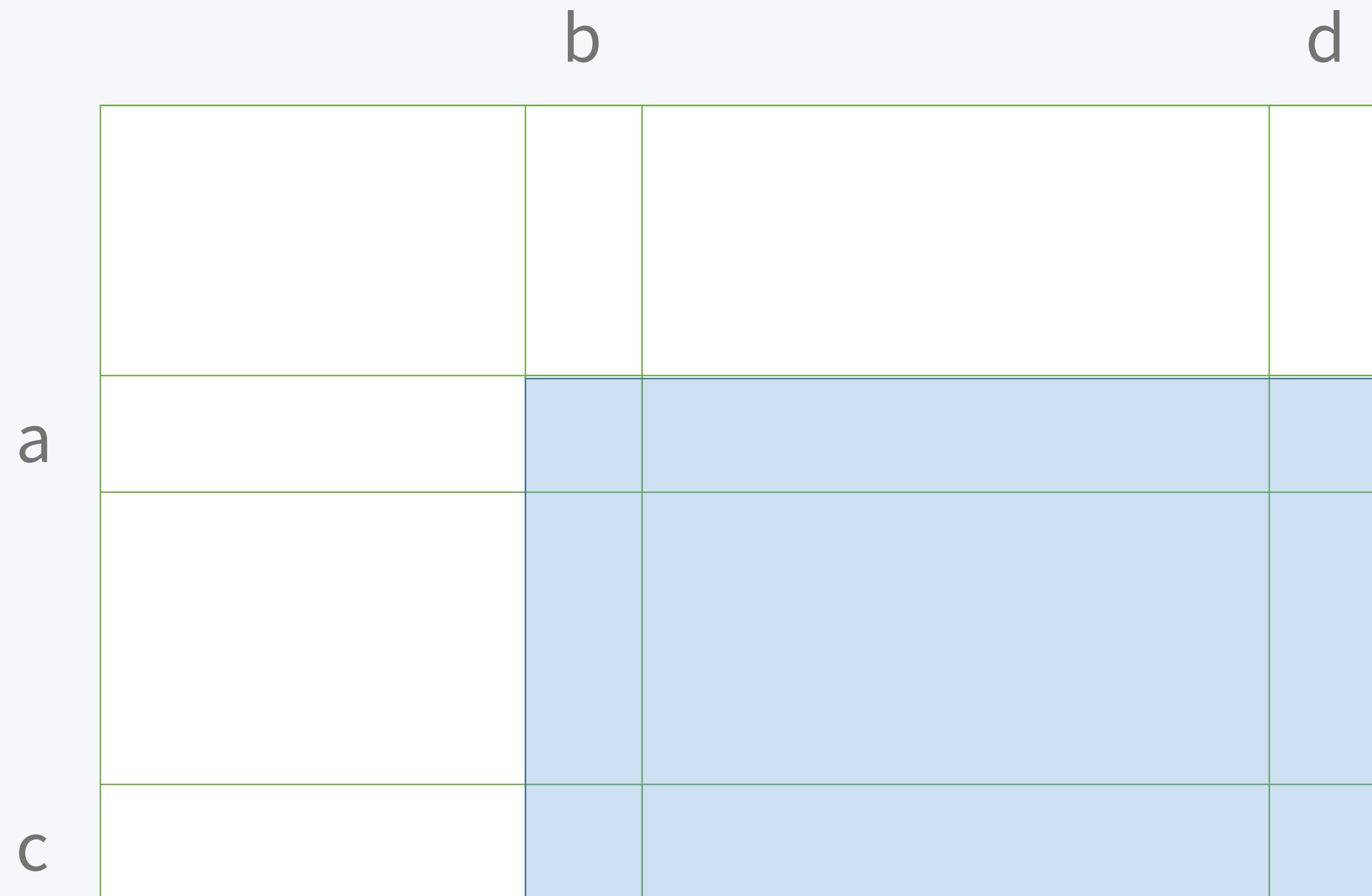
	b		d	
a				
c				

구간 합 구하기 5

17

<https://www.acmicpc.net/problem/11660>

- $(a,b) \sim (c,d)$ 합 구하기

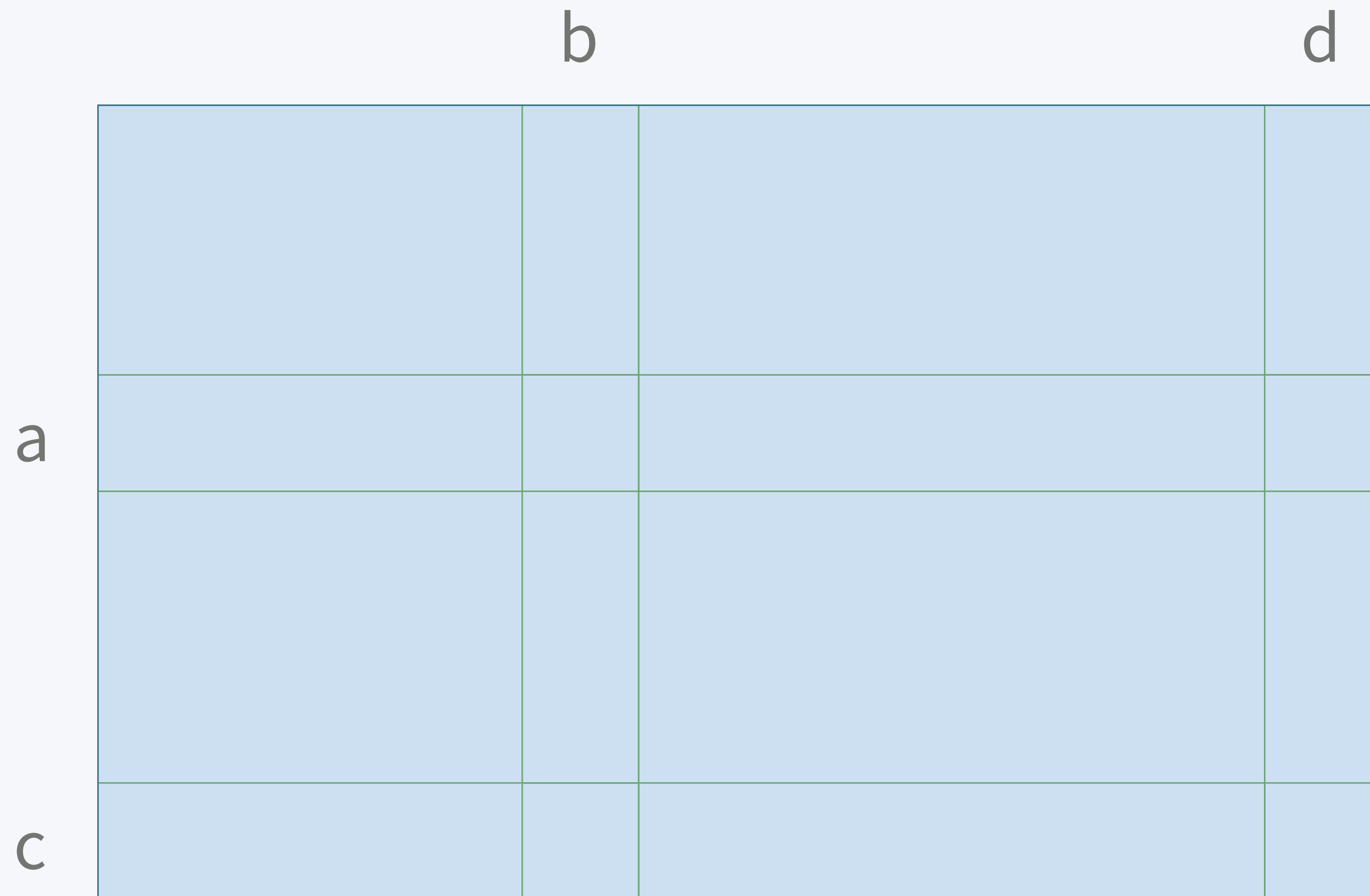


구간 합 구하기 5

18

<https://www.acmicpc.net/problem/11660>

- $S[c][d]$

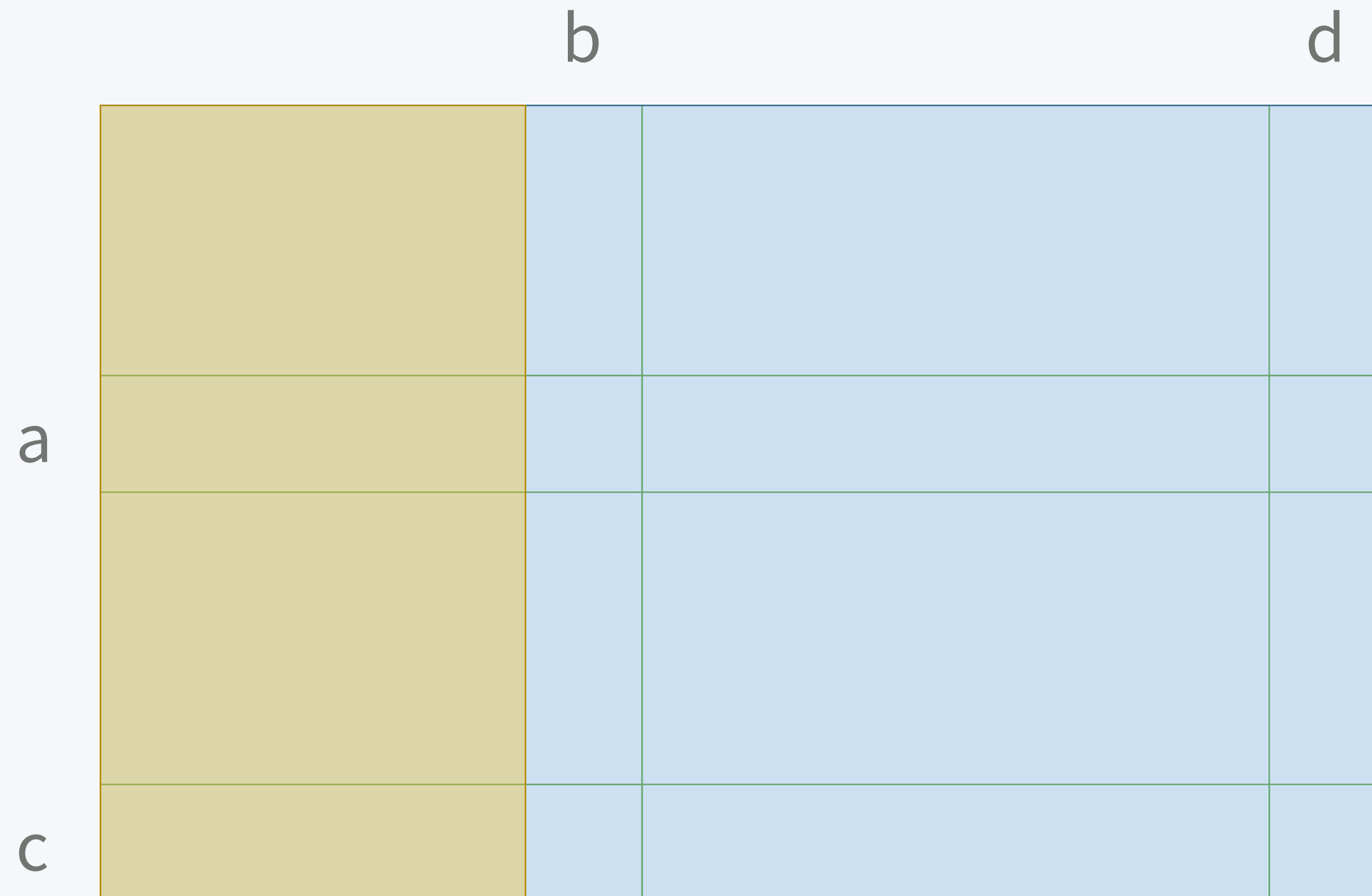


구간 합 구하기 5

19

<https://www.acmicpc.net/problem/11660>

- $S[c][d] - S[c][b-1]$

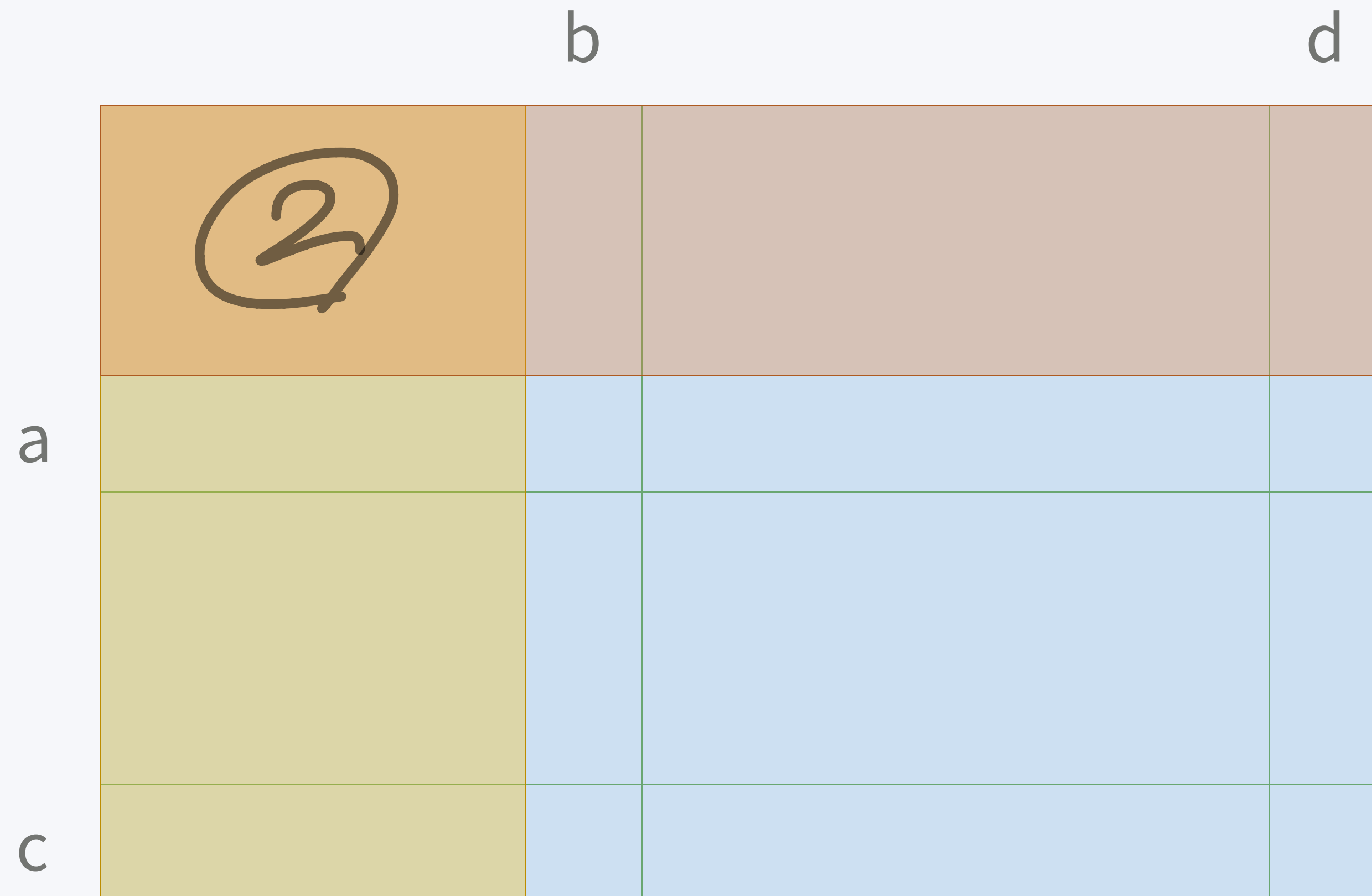


구간 합 구하기 5

20

<https://www.acmicpc.net/problem/11660>

- $S[c][d] - S[c][b-1] - S[a-1][d]$

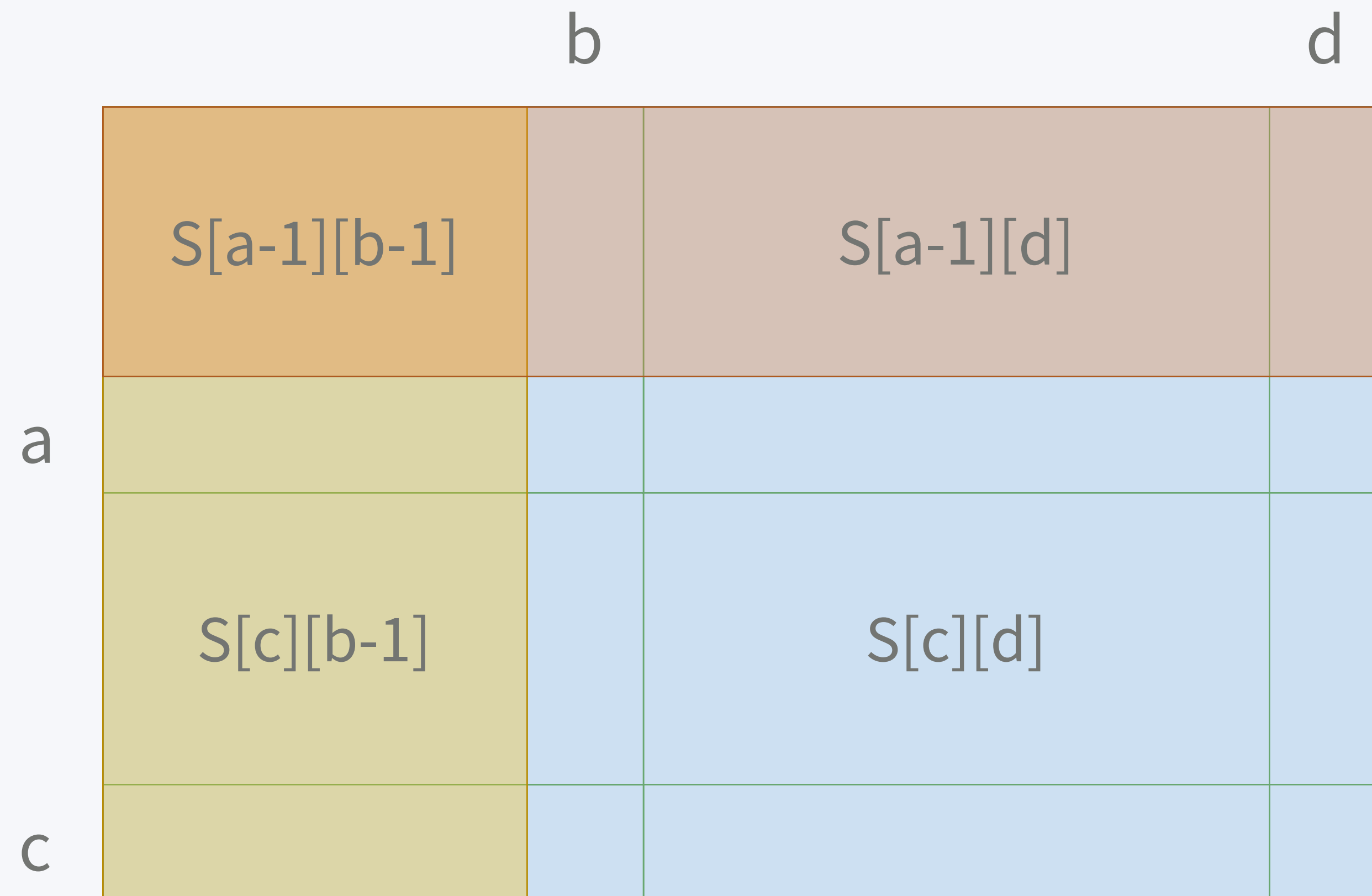


구간 합 구하기 5

21

<https://www.acmicpc.net/problem/11660>

- $S[c][d] - S[c][b-1] - S[a-1][d] + S[a-1][b-1]$



구간 합 구하기 5

22

<https://www.acmicpc.net/problem/11660>

- C/C++: <https://gist.github.com/Baekjoon/16cd8728b75c0449daa016d8890e0161>

세그먼트 트리

구분 $O(1)$
변경 $O(N)$

$O(\lg N)$

구간 합 구하기

<https://www.acmicpc.net/problem/2042>

- 구간의 최소값이 아니고 합을 구하는 경우에는
- min 대신 +를 하면 된다
- 구간의 합을 구하는 경우에는 중간에 수를 변경할 수 있다.

$\text{min}, +$

구간 합 구하기

2가2-1

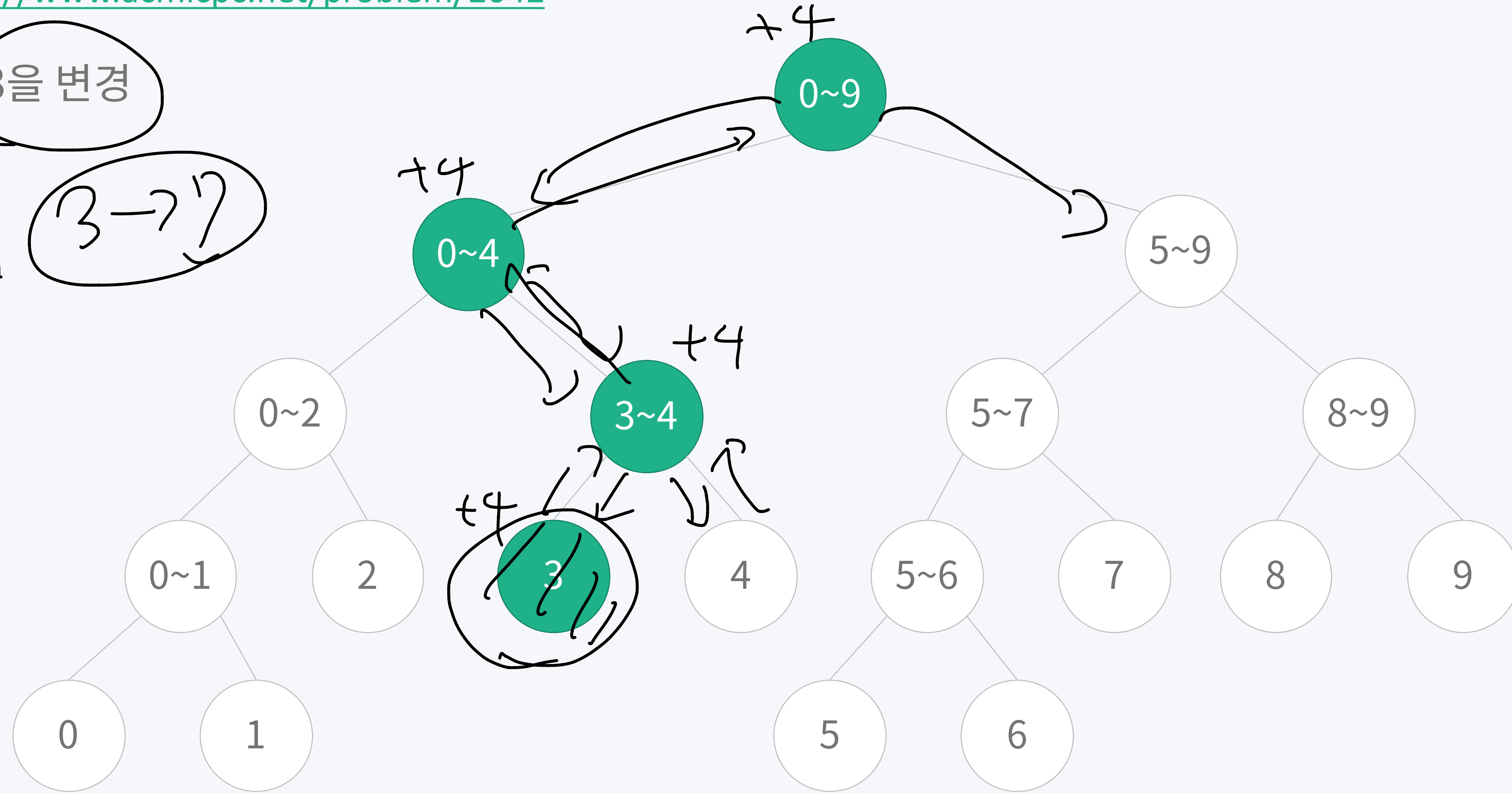
25

<https://www.acmicpc.net/problem/2042>

- 3을 변경

A

3-77

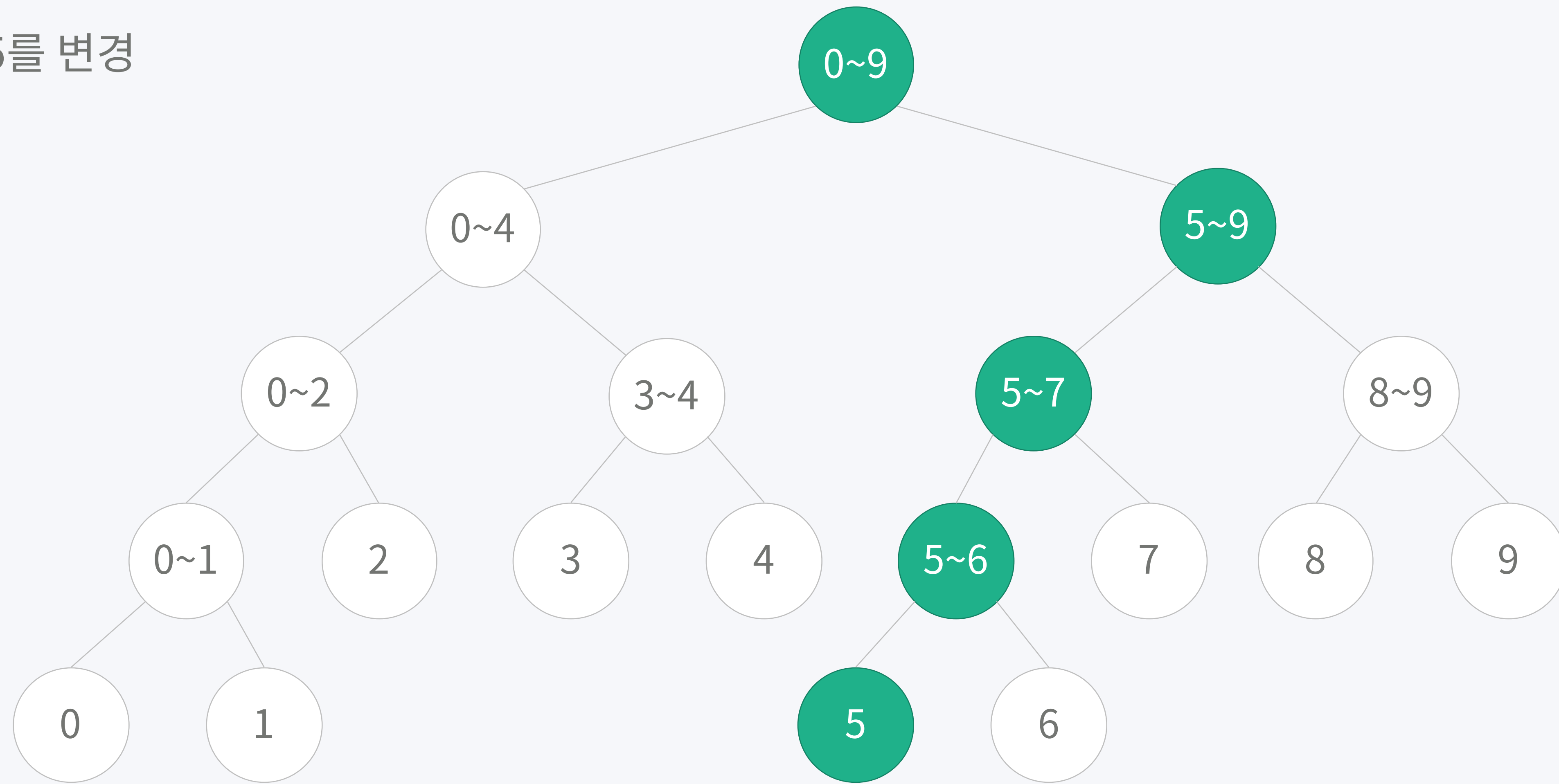


구간 합 구하기

26

<https://www.acmicpc.net/problem/2042>

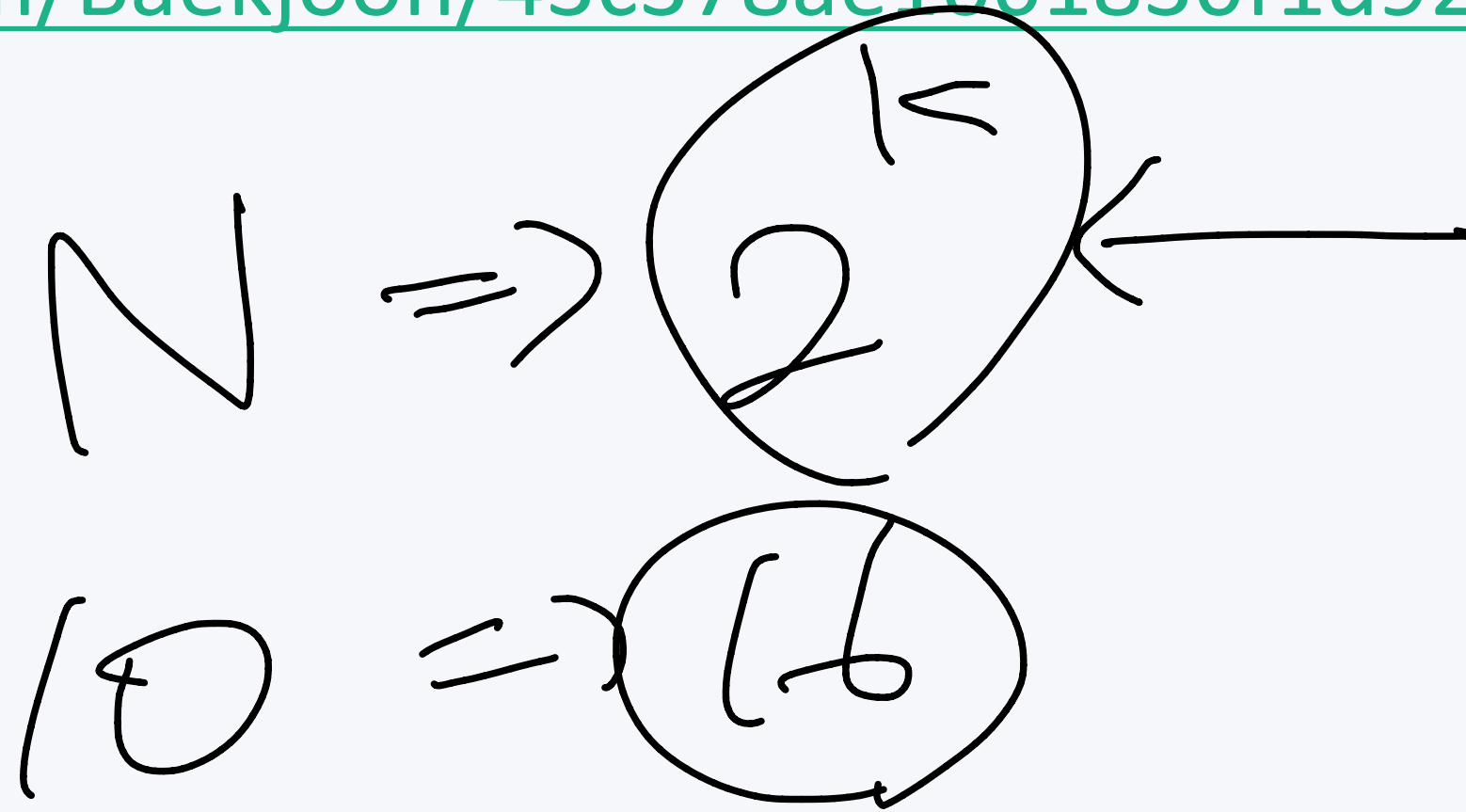
- 5를 변경



구간 합 구하기

<https://www.acmicpc.net/problem/2042>

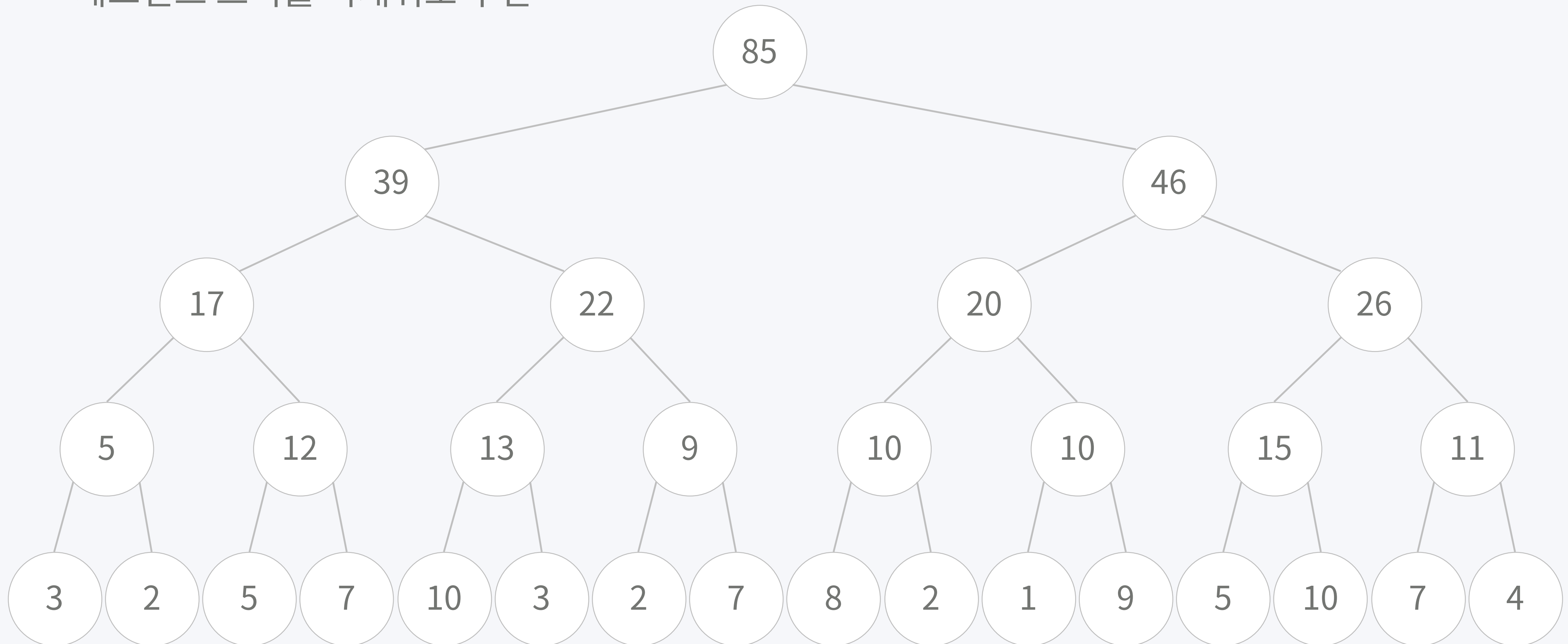
- C/C++: <https://gist.github.com/Baekjoon/43c378ae1661830f1d92>



세그먼트 트리 비재귀 구현

구간의 합 구하기

- 세그먼트 트리를 비재귀로 구현



세그먼트 트리 비재귀 구현

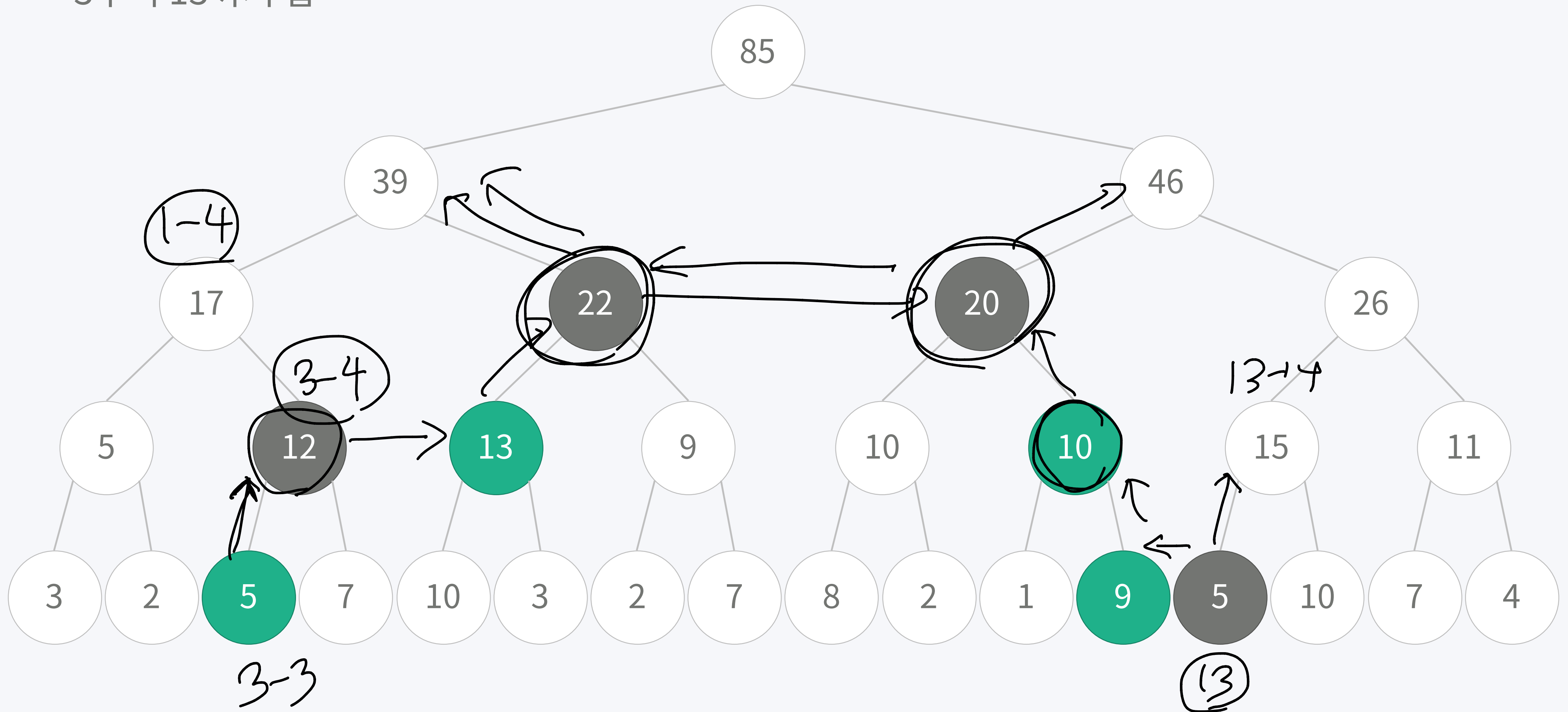
29

구간의 합 구하기

검증: 합을 더함

3-13

- 3부터 13까지 합

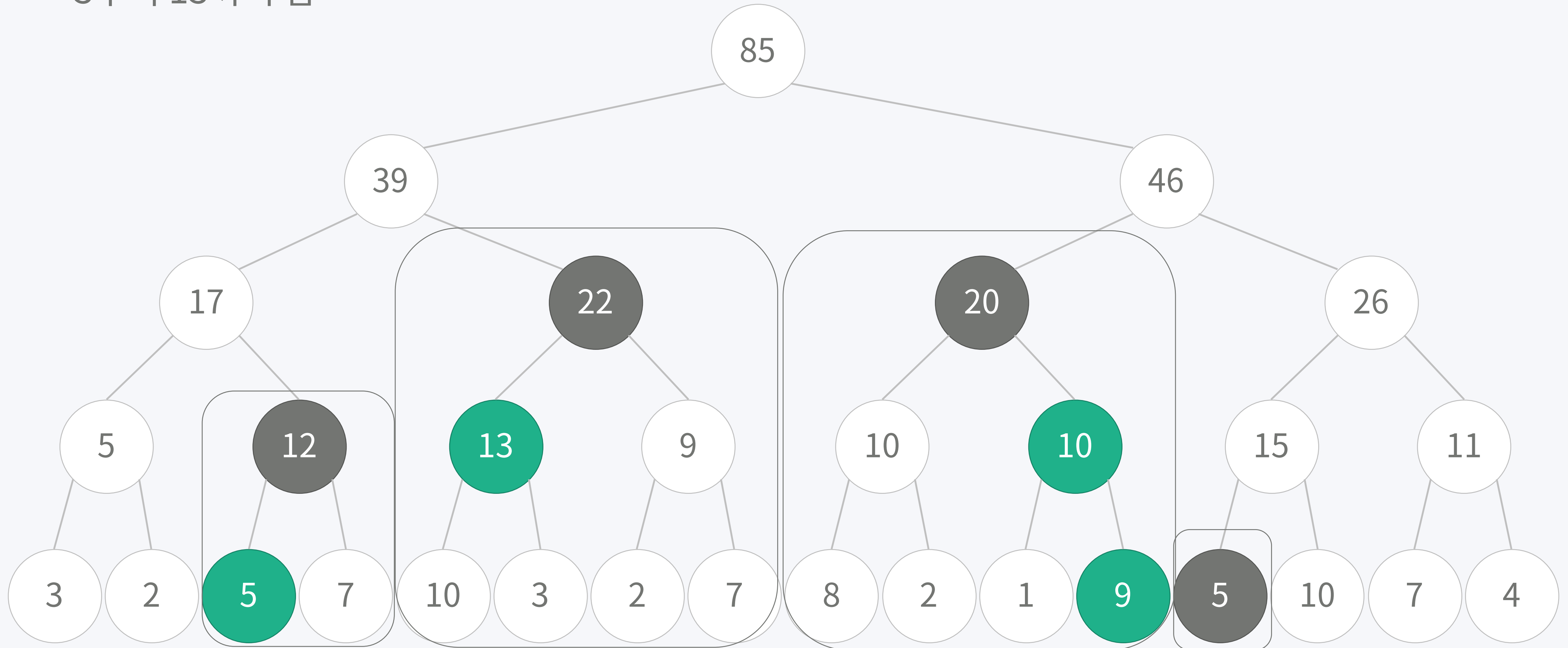


세그먼트 트리 비재귀 구현

30

구간의 합 구하기

- 3부터 13까지 합

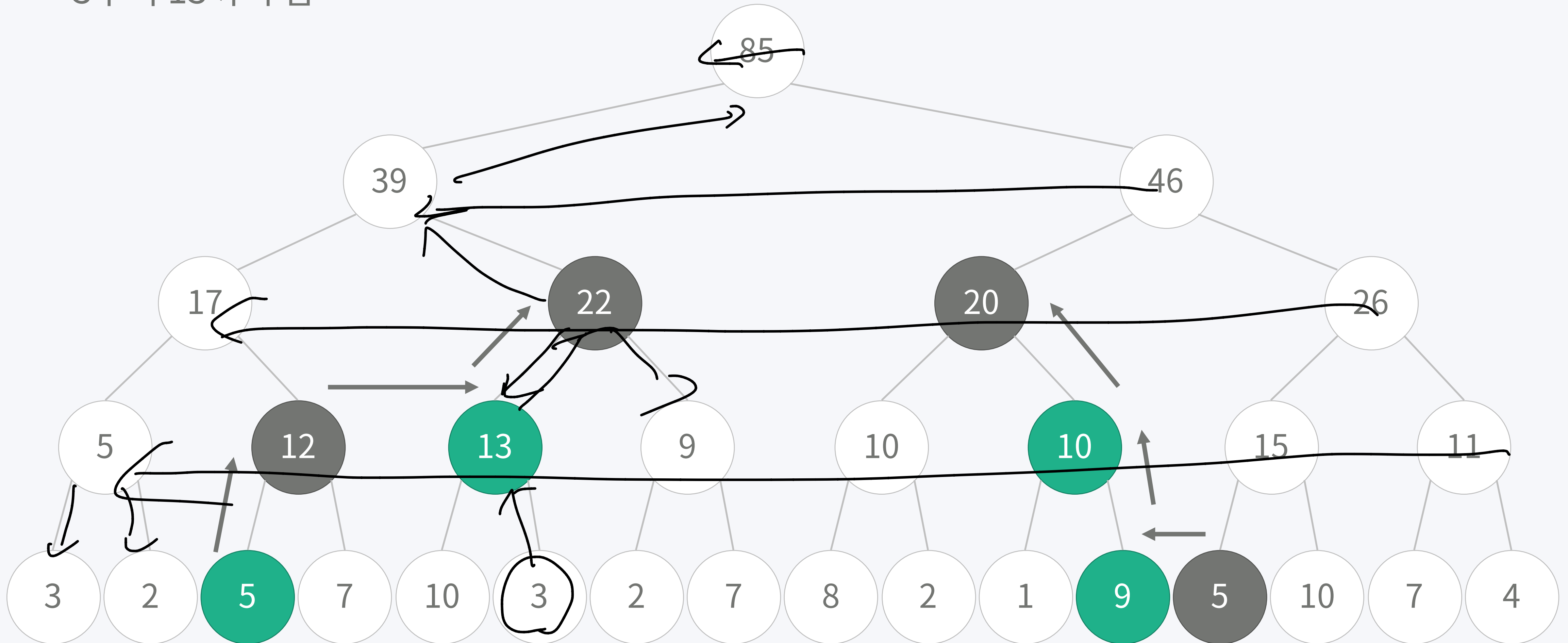


세그먼트 트리 비재귀 구현

31

구간의 합 구하기

- 3부터 13까지 합

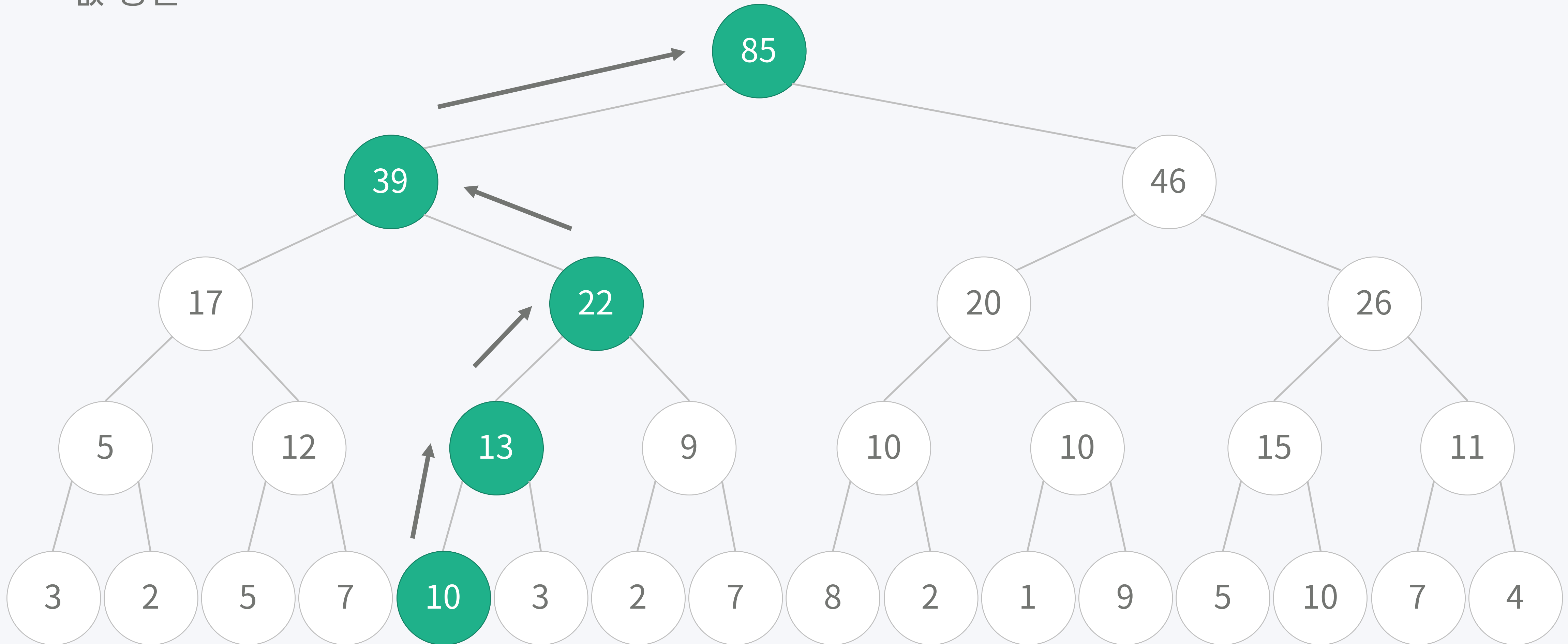


세그먼트 트리 비재귀 구현

32

구간의 합 구하기

- 값 갱신



세그먼트 트리 비재귀 구현

구간의 합 구하기

- 합 구하는 방법
- 왼쪽
 - 왼쪽 자식이면 올라간다
 - 오른쪽 자식이면 답을 더하고, 오른쪽 칸으로 이동
- 오른쪽
 - 오른쪽 자식이면 올라간다
 - 왼쪽 자식이면 답을 더하고, 왼쪽 칸으로 이동

세그먼트 트리 비재귀 구현

구간의 합 구하기

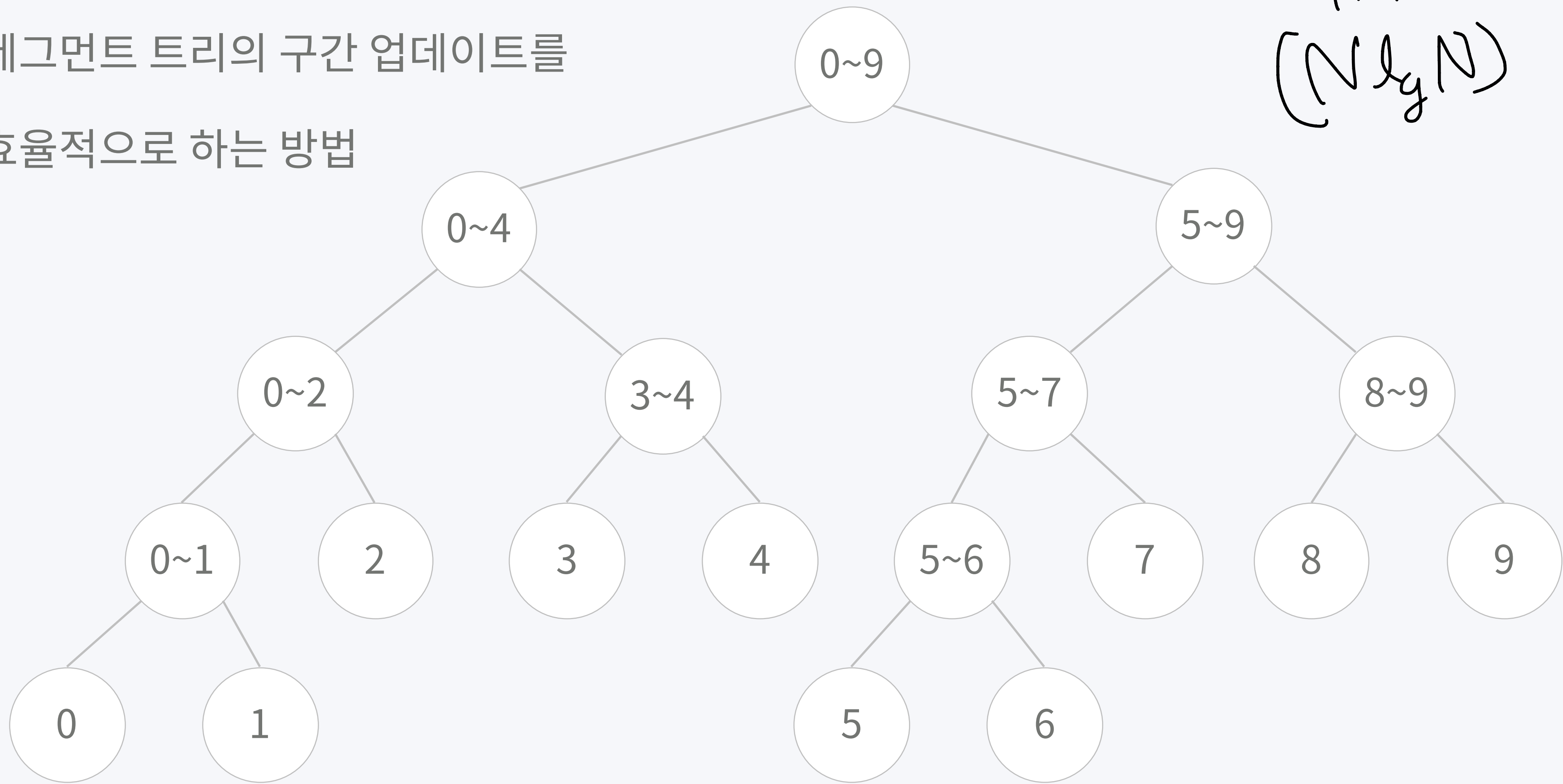
- C/C++: <https://gist.github.com/Baekjoon/fe54fa44050f0e1f4826>

Lazy Propagation

구간의 합구하기

- 세그먼트 트리의 구간 업데이트를
- 효율적으로 하는 방법

업데이트: $\frac{O(\lg N)}{17H \uparrow}$
 $(N \lg N)$



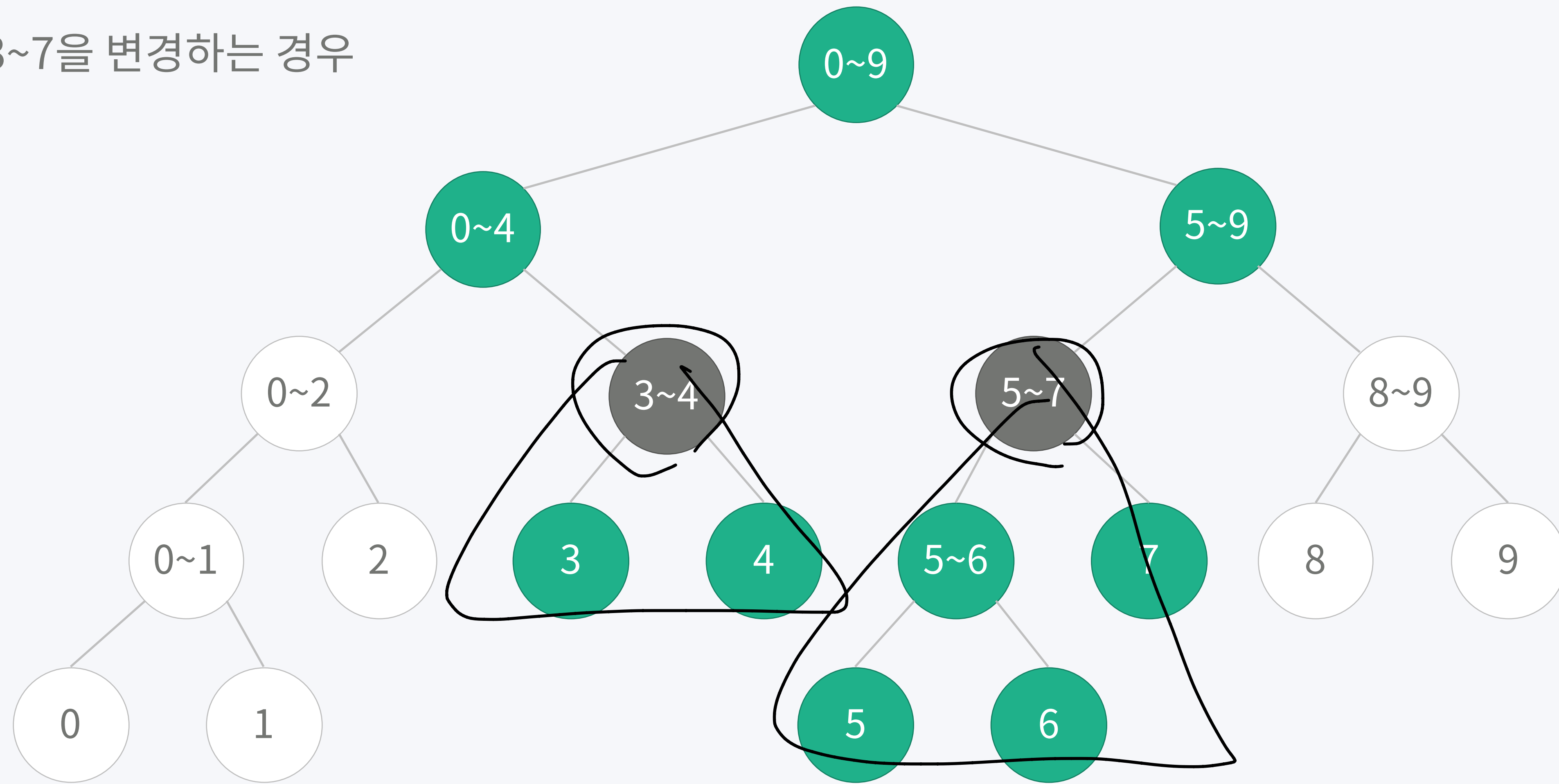
Lazy Propagation

3~7

36

구간의 합구하기

- 3~7을 변경하는 경우

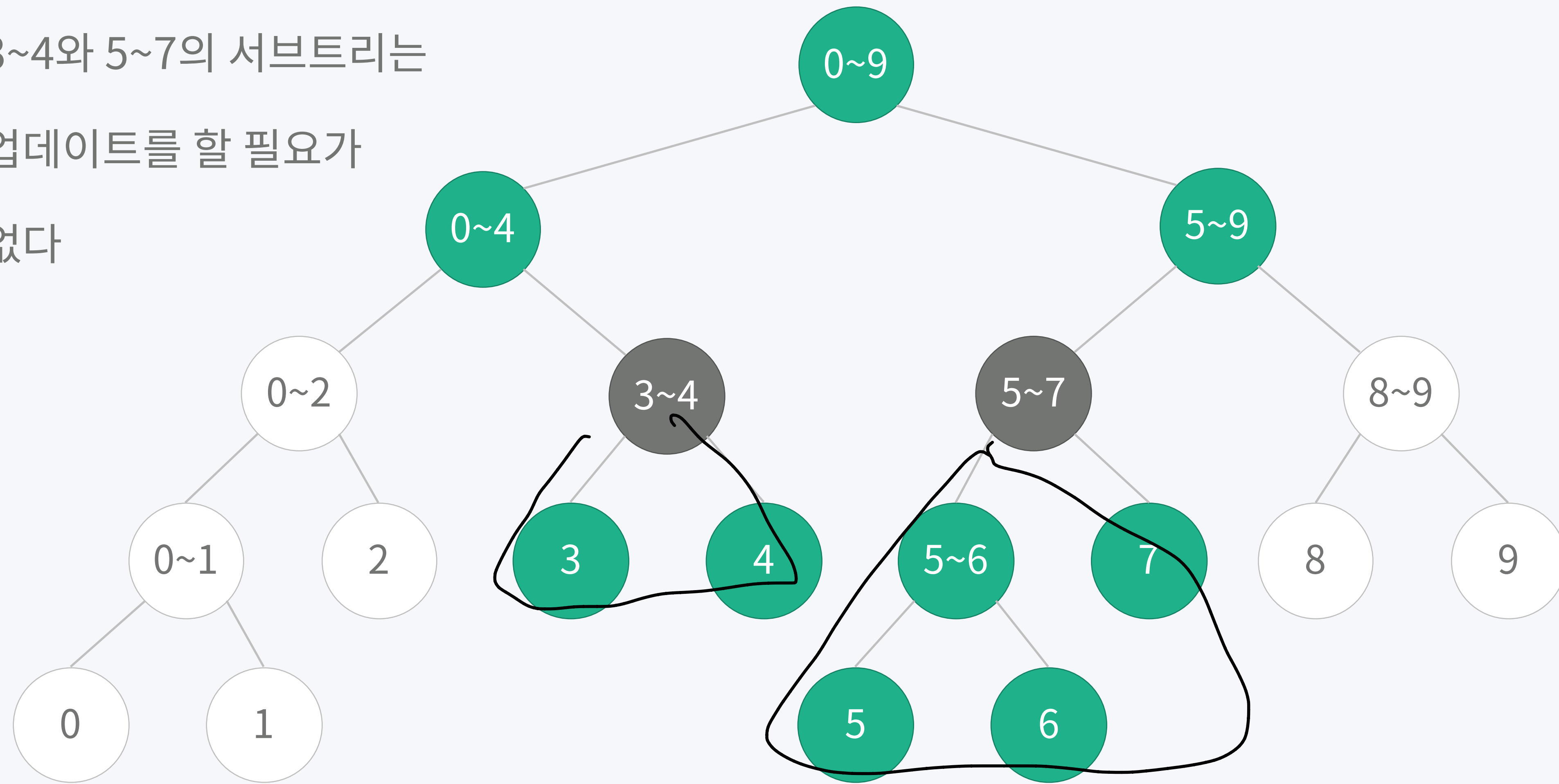


Lazy Propagation

37

구간의 합 구하기

- 3~4와 5~7의 서브트리는
- 업데이트를 할 필요가
- 없다

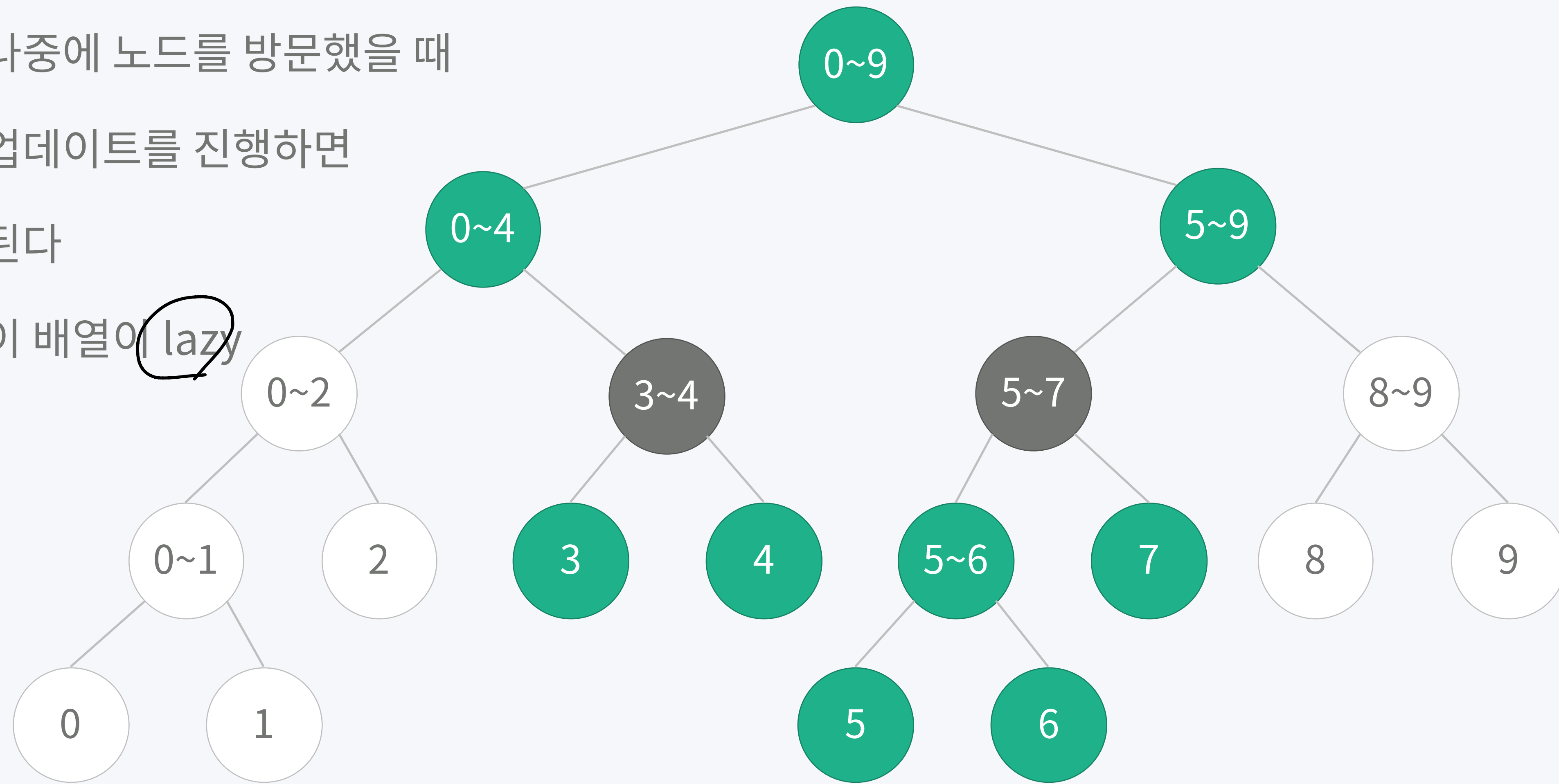


Lazy Propagation

38

구간의 합 구하기

- 나중에 노드를 방문했을 때
- 업데이트를 진행하면
- 된다
- 이 배열이 lazy



Lazy Propagation

구간의 합 구하기

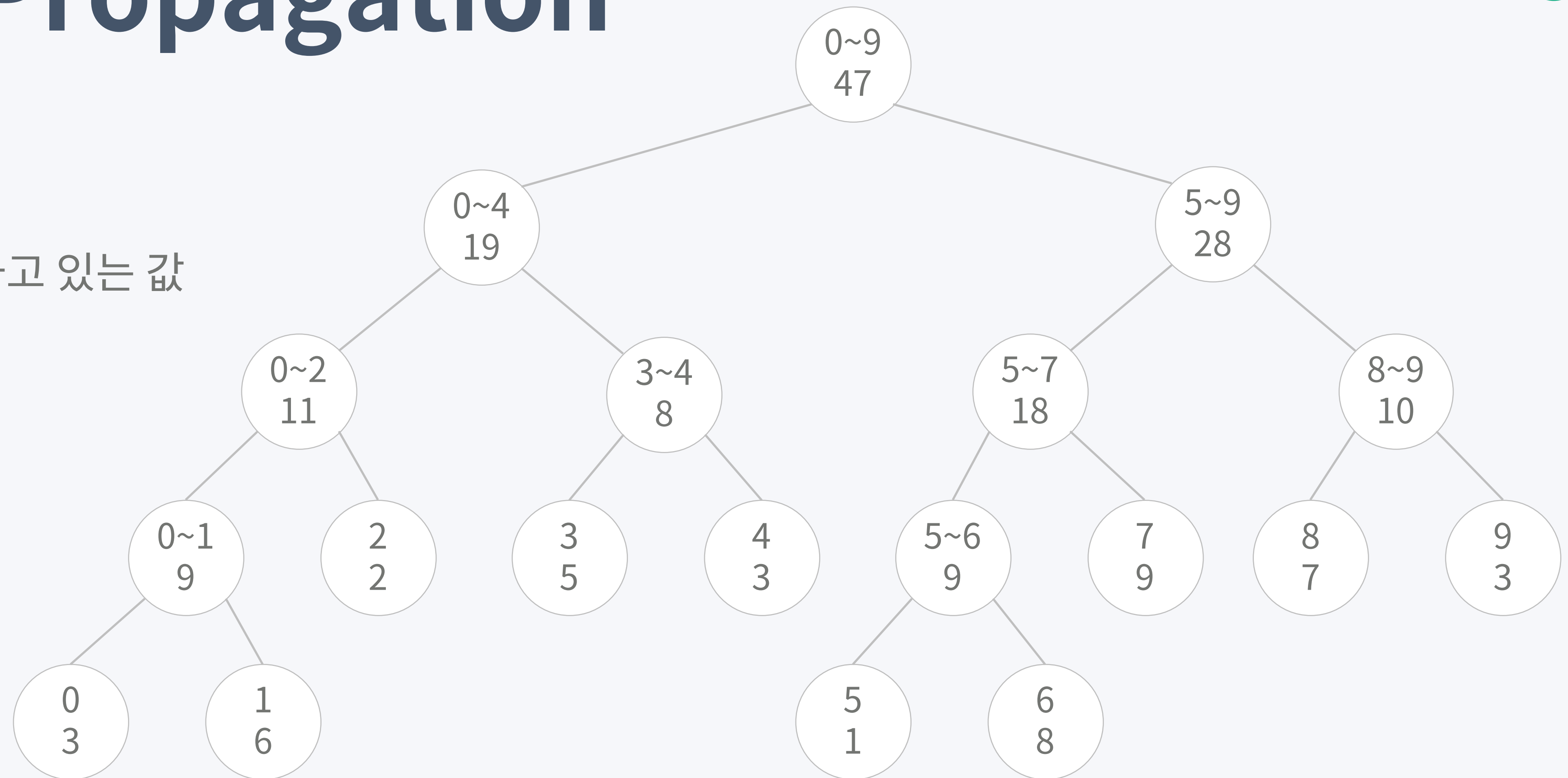
- 3~4를 나타내는 노드의 lazy에 10이 저장되어 있으면
- 3번째 수와 4번째 수에 10을 더해야 하는데, 나중에 10을 더하겠다는 의미
- 5~7의 lazy에 20이 저장되어 있다면
- 5, 6, 7번째 수에 20을 더해야 하지만, 지금은 더하지 않고 나중에 더하겠다는 의미

Lazy Propagation

40

구간의 합 구하기

- 위: 범위
- 아래: 저장하고 있는 값

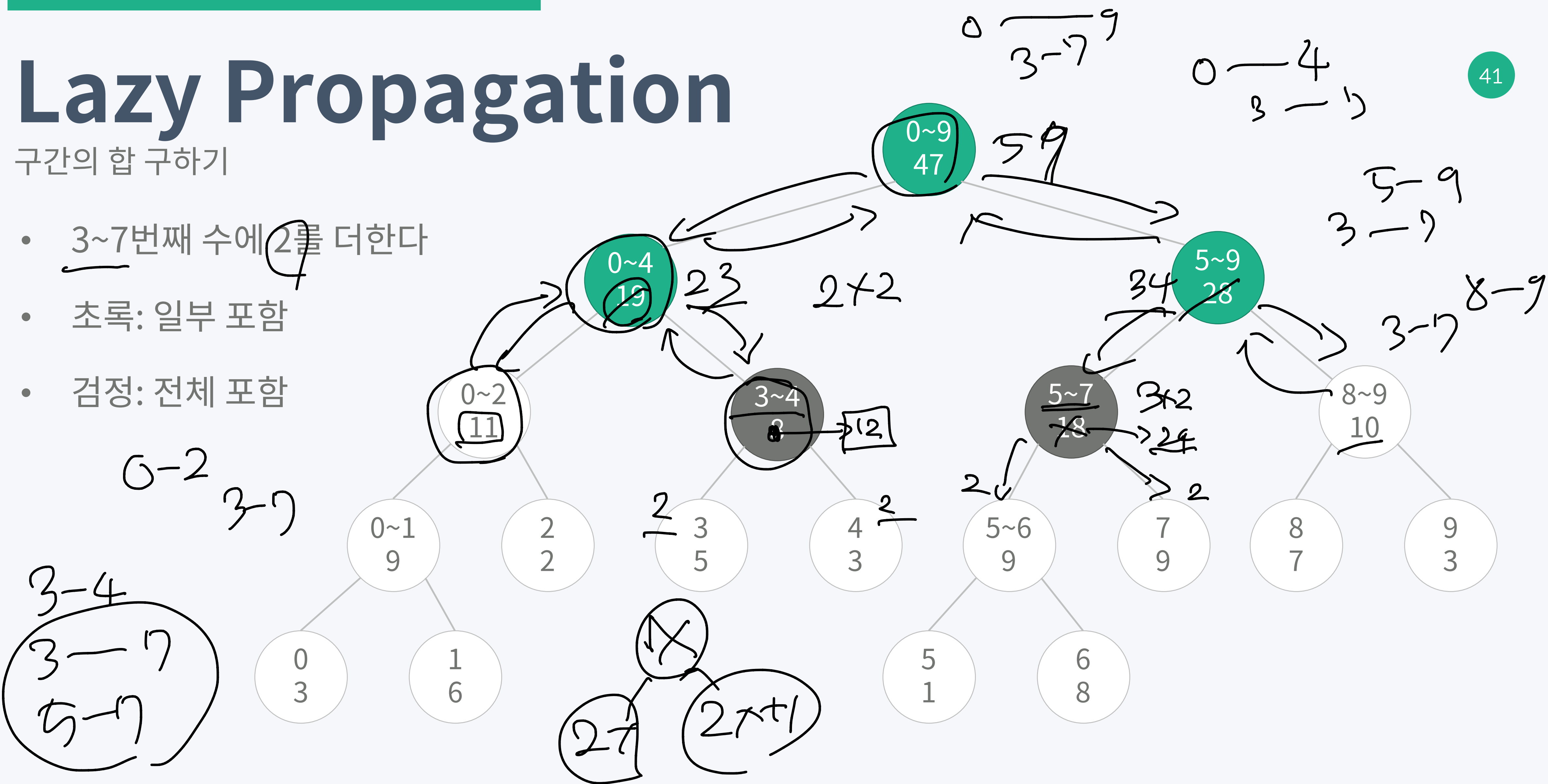


A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
3	6	2	5	3	1	8	9	7	3

Lazy Propagation

구간의 합 구하기

- 3~7번째 수에 2를 더한다
- 초록: 일부 포함
- 검정: 전체 포함

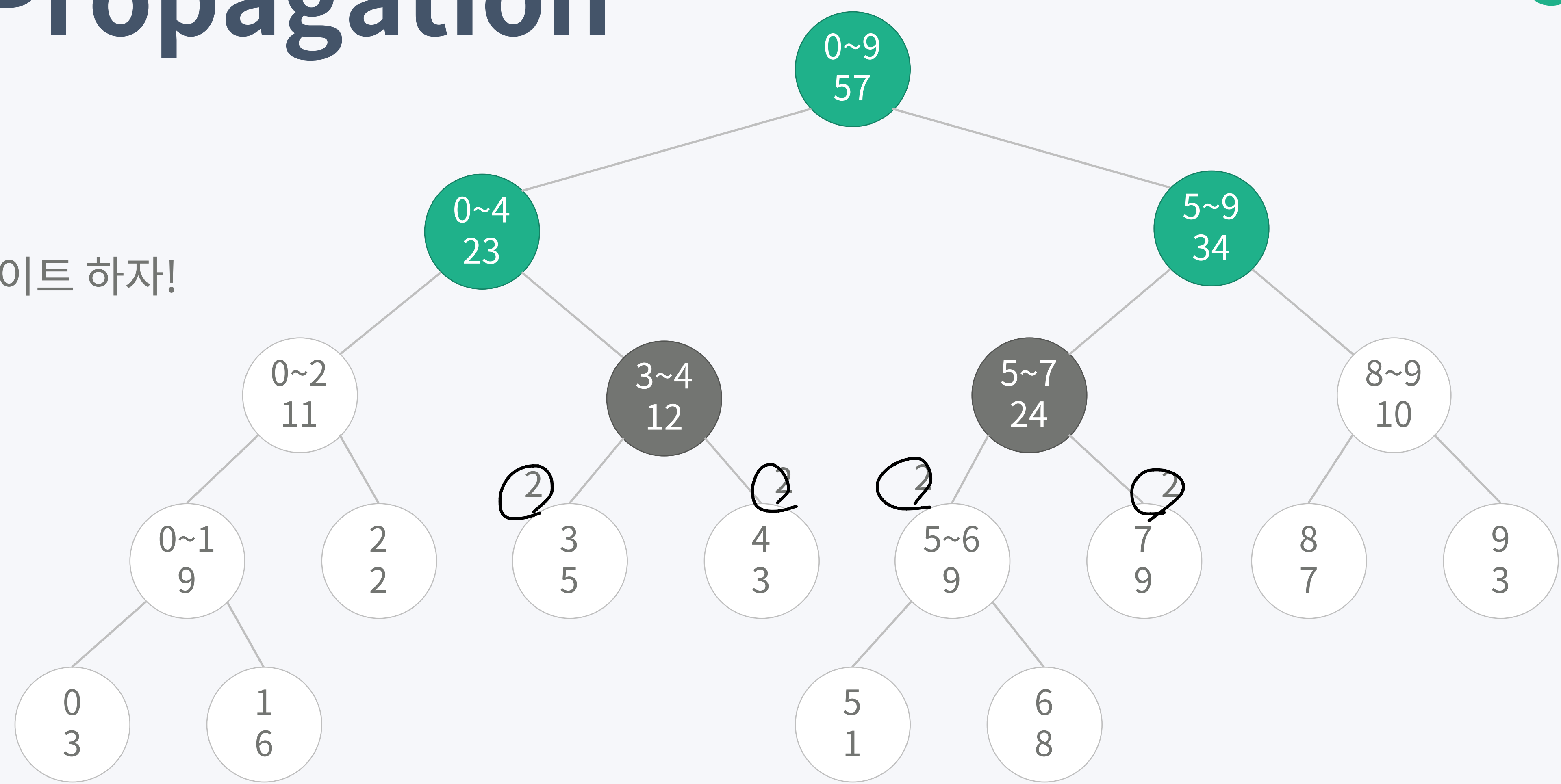


A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
3	6	2	5	3	1	8	9	7	3

Lazy Propagation

구간의 합 구하기

- 검정 아래는
- 나중에 업데이트 하자!



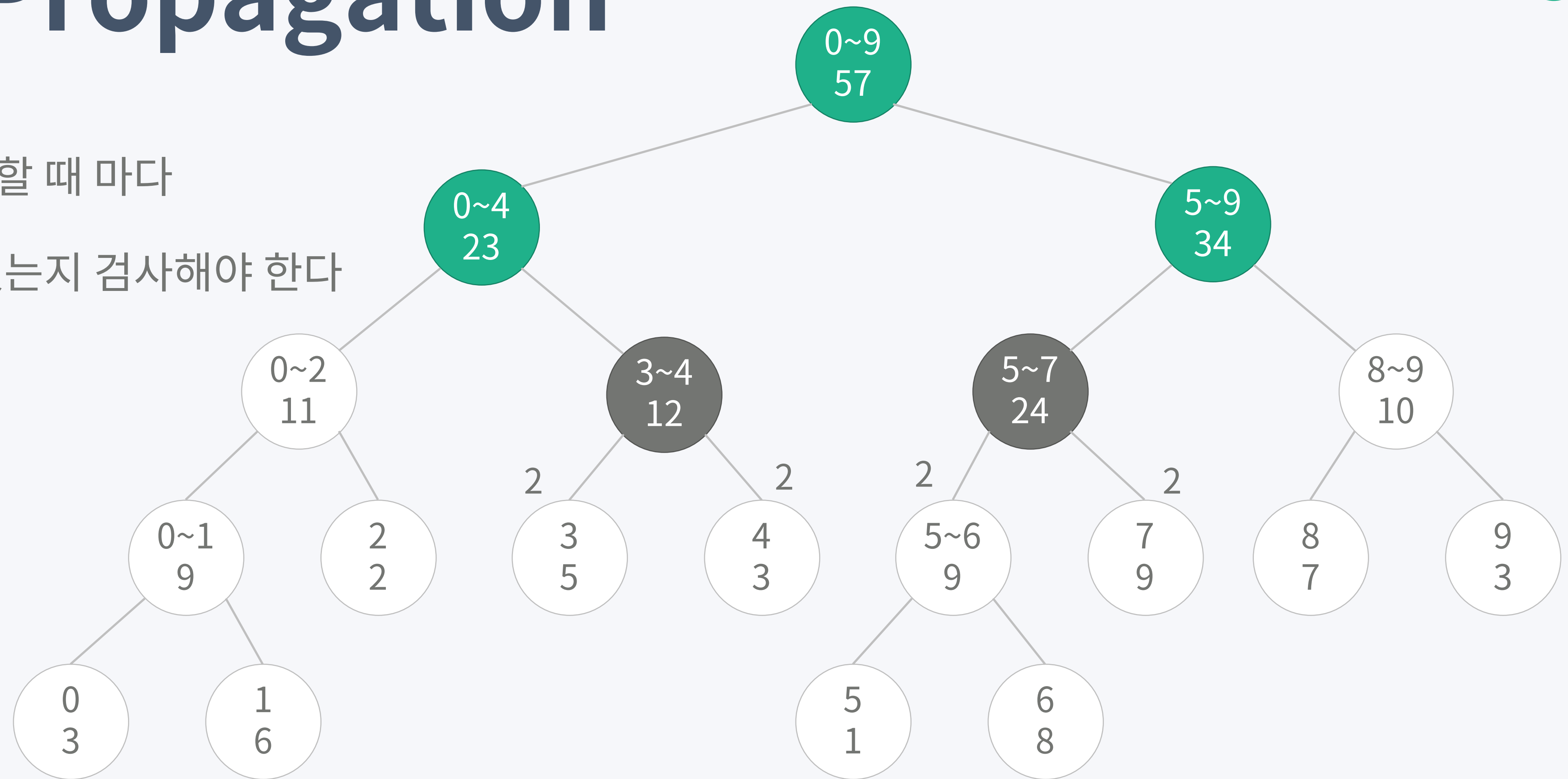
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
3	6	2	7	5	3	10	11	7	3

Lazy Propagation

43

구간의 합 구하기

- 노드를 방문할 때 마다
- lazy 값이 있는지 검사해야 한다

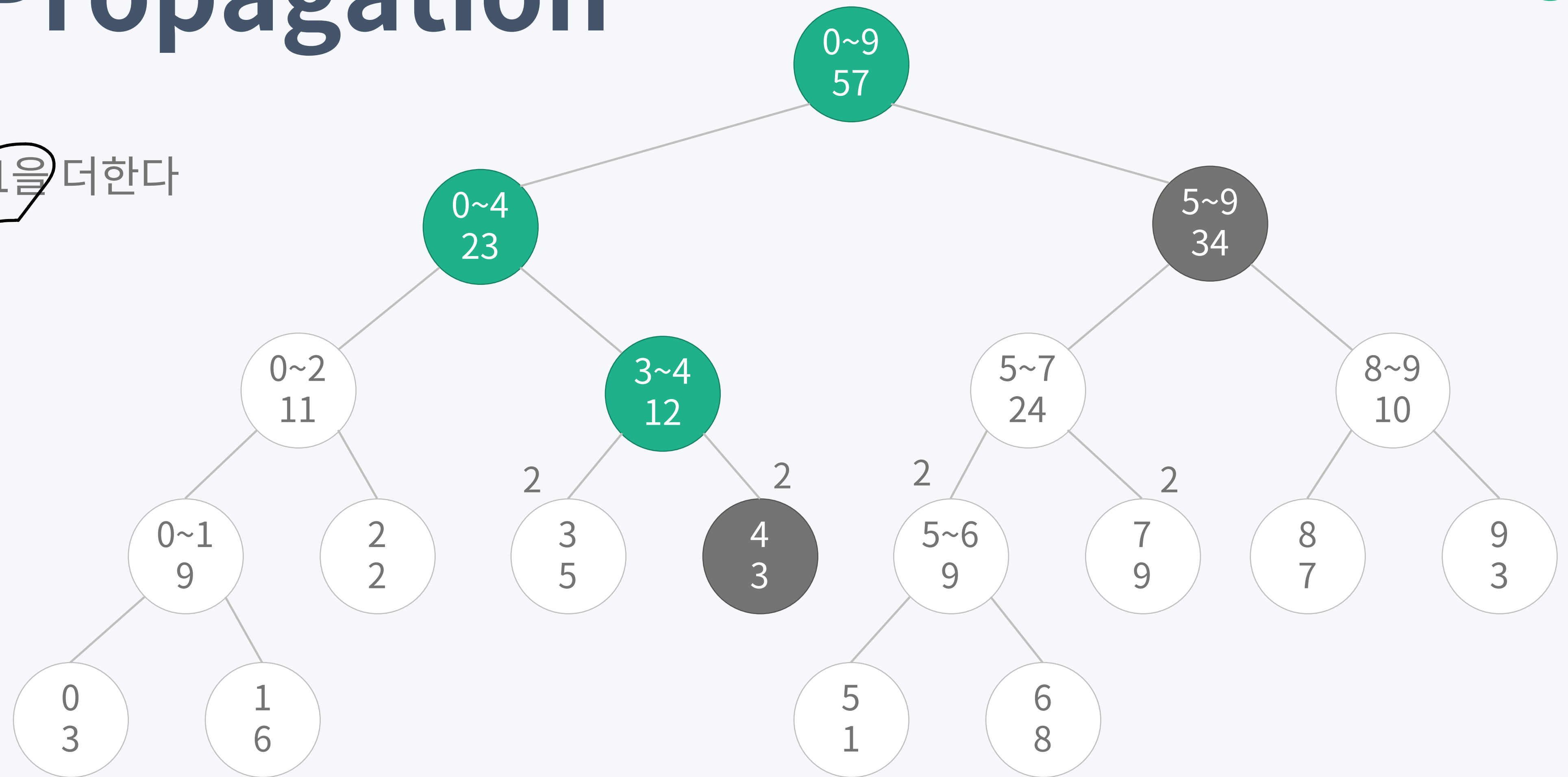


A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
3	6	2	7	5	3	10	11	7	3

Lazy Propagation

구간의 합 구하기

- 4~9번째에 1을 더한다



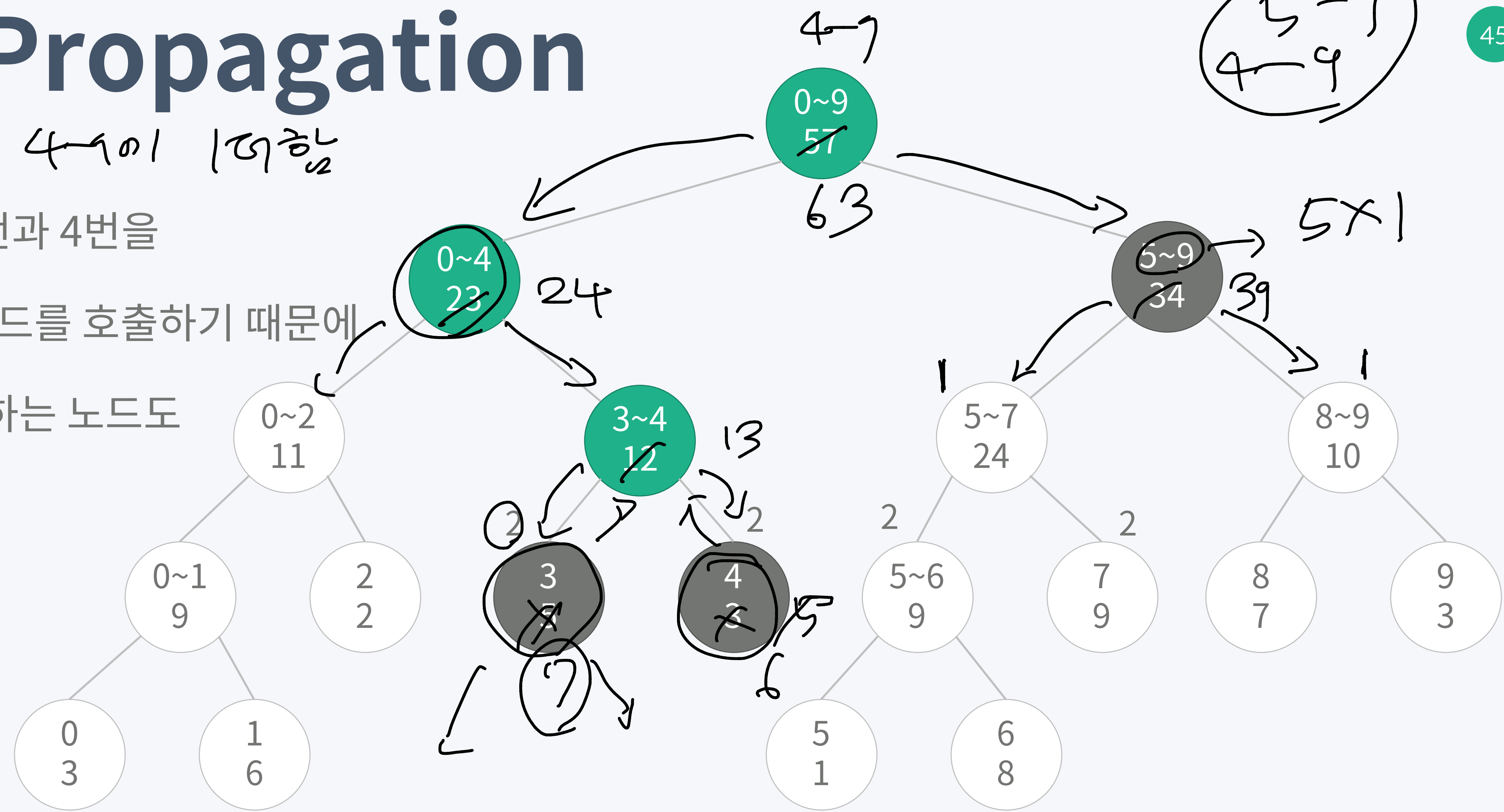
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
3	6	2	7	5	3	10	11	7	3

Lazy Propagation

구간의 합 구하기 $4 \sim 9$ 이 12 만큼

- 3~4에서 3번과 4번을
- 담당하는 노드를 호출하기 때문에
- 3번만 담당하는 노드도
- 호출

$4 \sim 9$
 $0 \sim 4$
 $3 \sim 4$
 $4 \sim 9$



$5 \sim 9$
 $4 \sim 9$

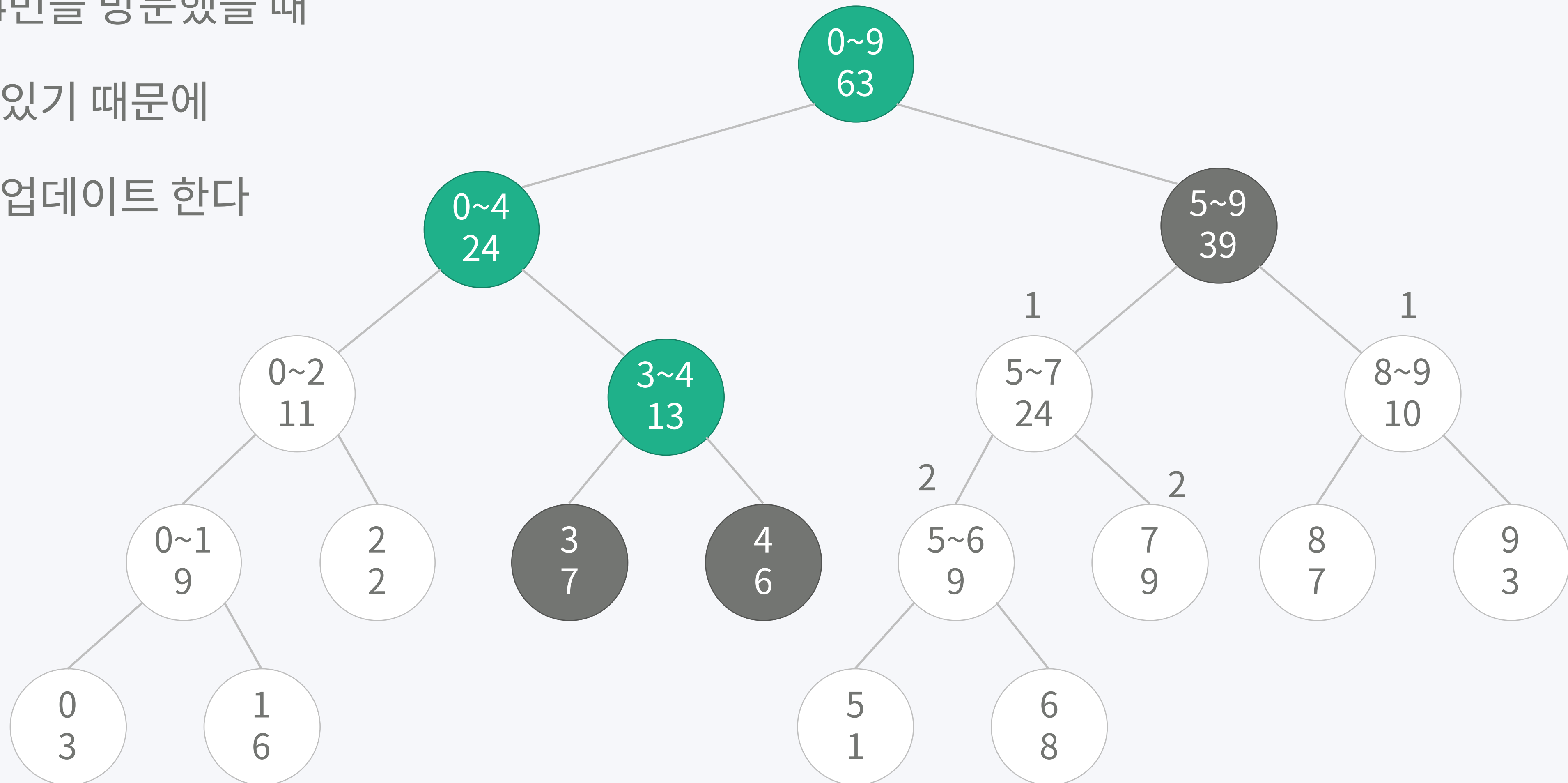
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
3	6	2	7	5	3	10	11	7	3

Lazy Propagation

46

구간의 합 구하기

- 3번과 4번을 방문했을 때
- lazy가 있기 때문에
- lazy를 업데이트 한다

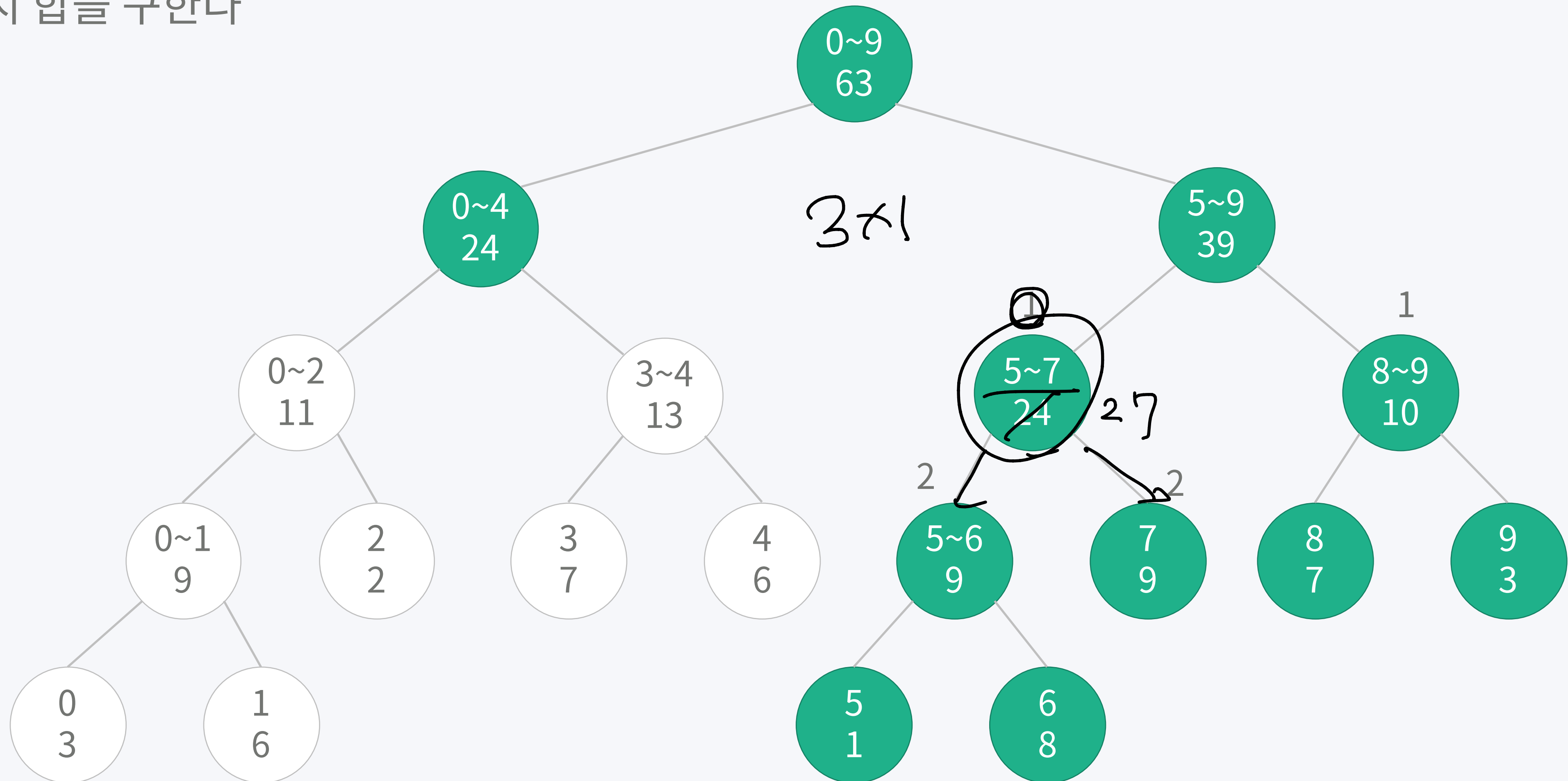


Lazy Propagation

47

구간의 합 구하기

- 6~8까지 합을 구한다

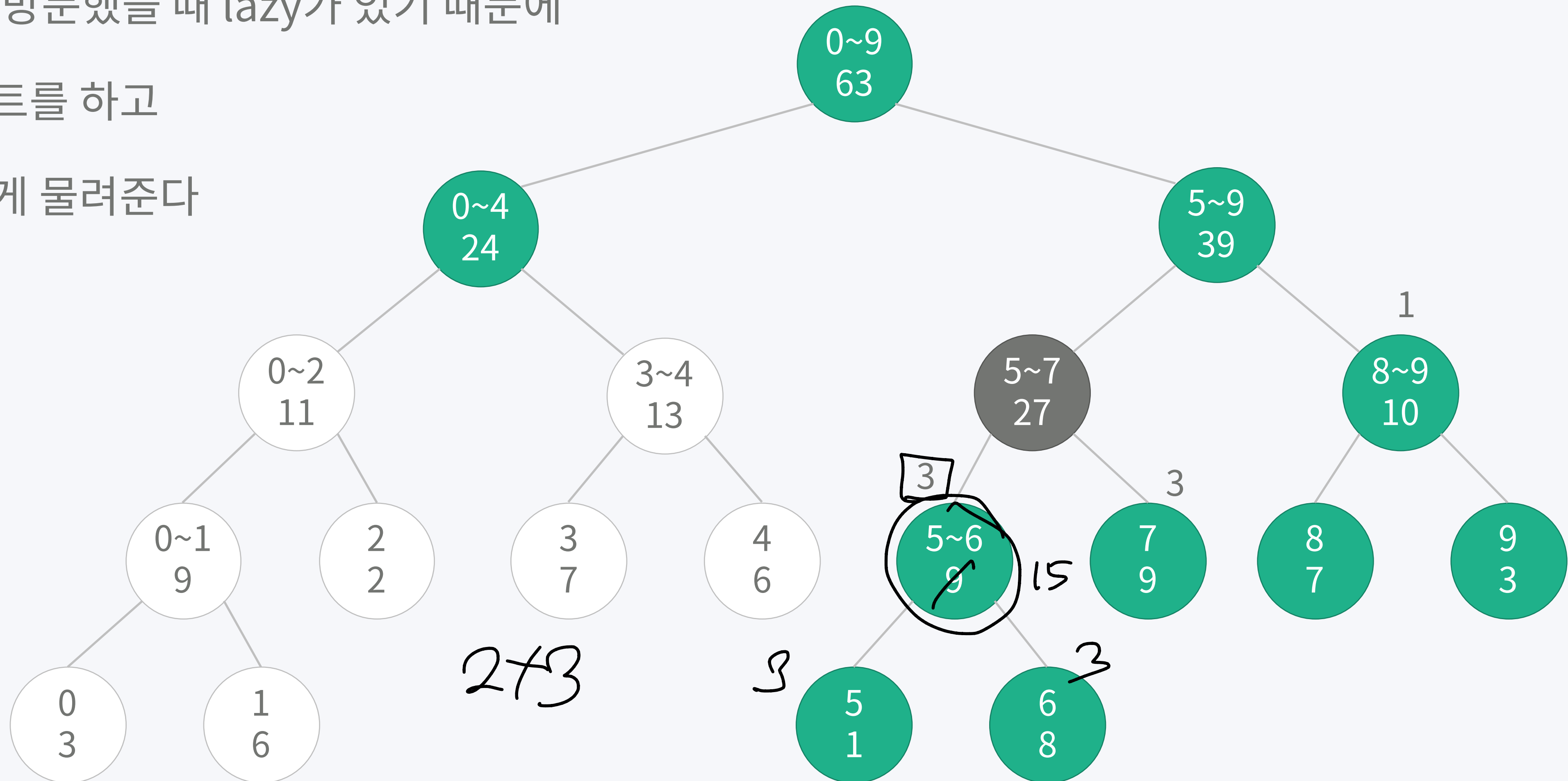


Lazy Propagation

48

구간의 합 구하기

- 5~7 을 방문했을 때 lazy가 있기 때문에
- 업데이트를 하고
- 자식에게 물려준다

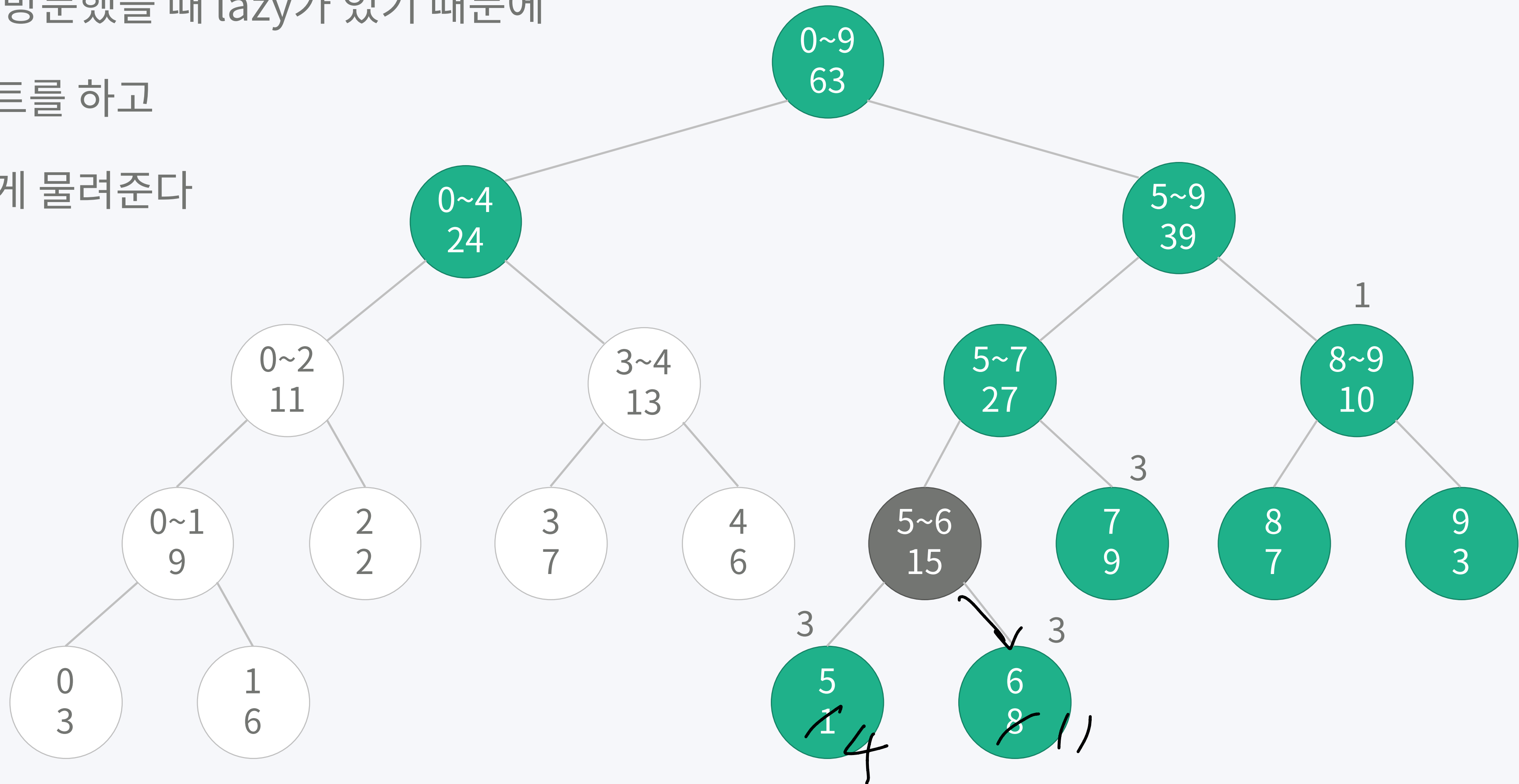


Lazy Propagation

49

구간의 합 구하기

- 5~6 을 방문했을 때 lazy가 있기 때문에
- 업데이트를 하고
- 자식에게 물려준다

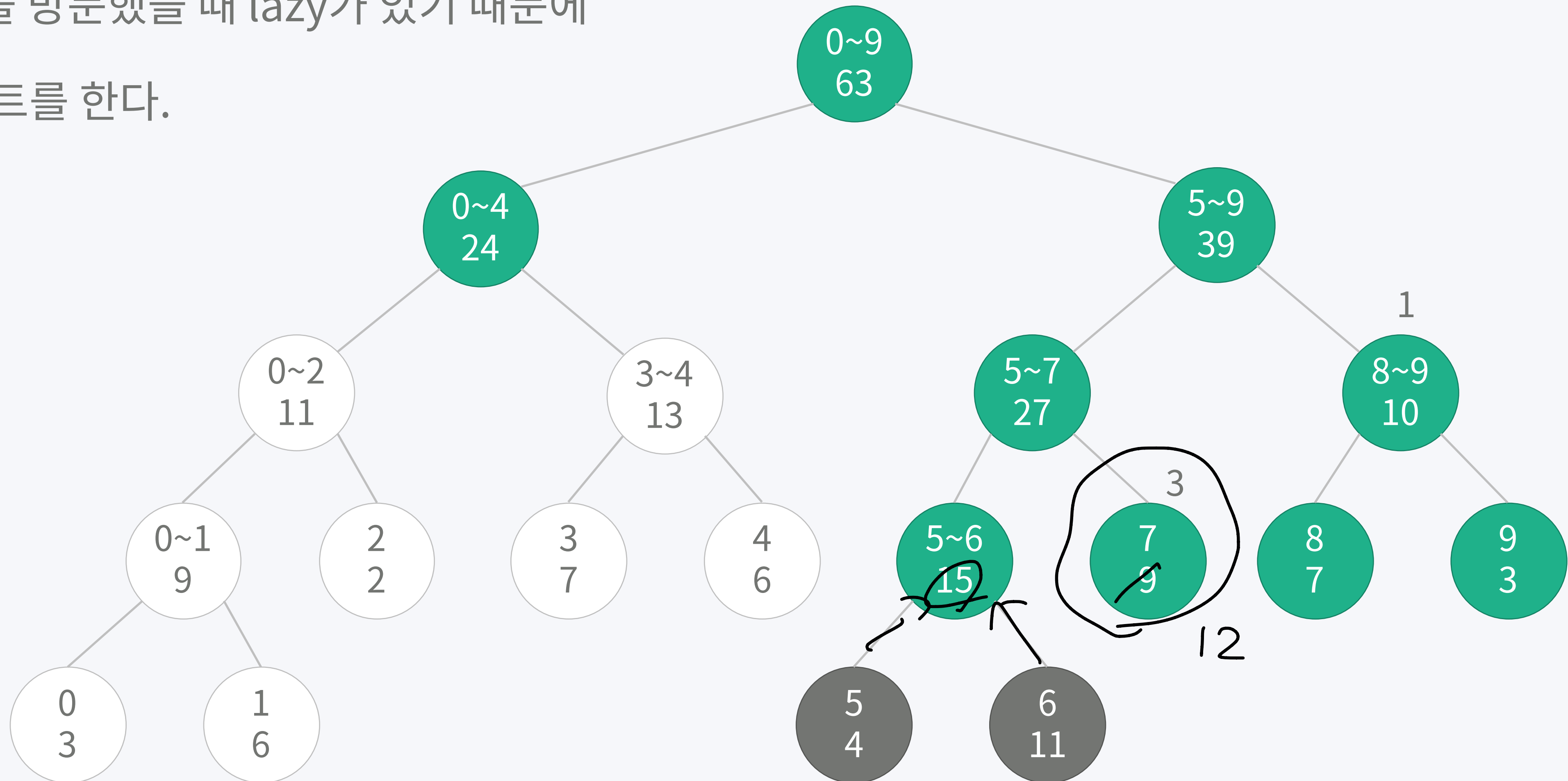


Lazy Propagation

50

구간의 합 구하기

- 5와 6 을 방문했을 때 lazy가 있기 때문에
- 업데이트를 한다.

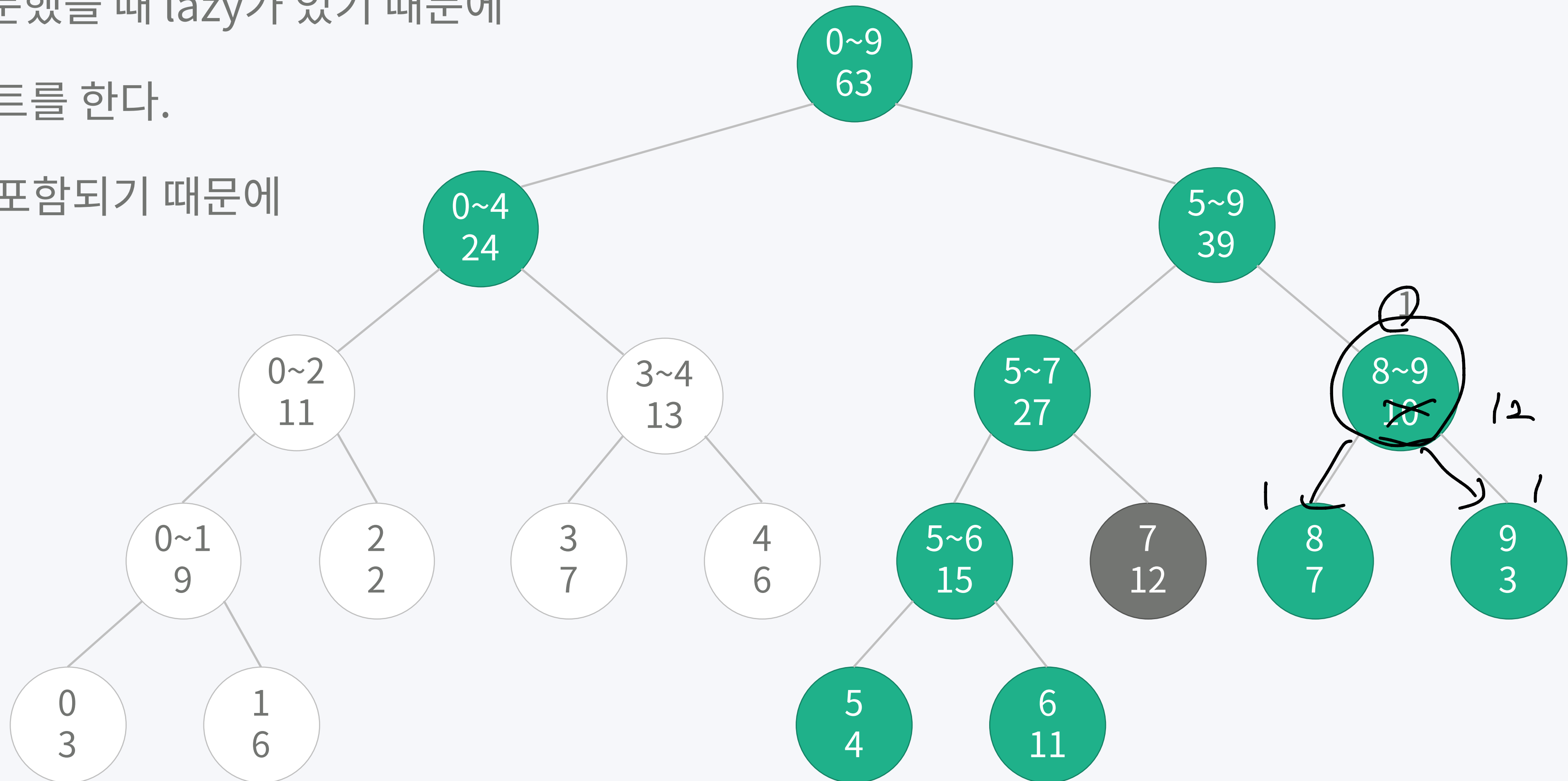


Lazy Propagation

51

구간의 합 구하기

- 7을 방문했을 때 lazy가 있기 때문에
- 업데이트를 한다.
- 6~8에 포함되기 때문에
- 리턴

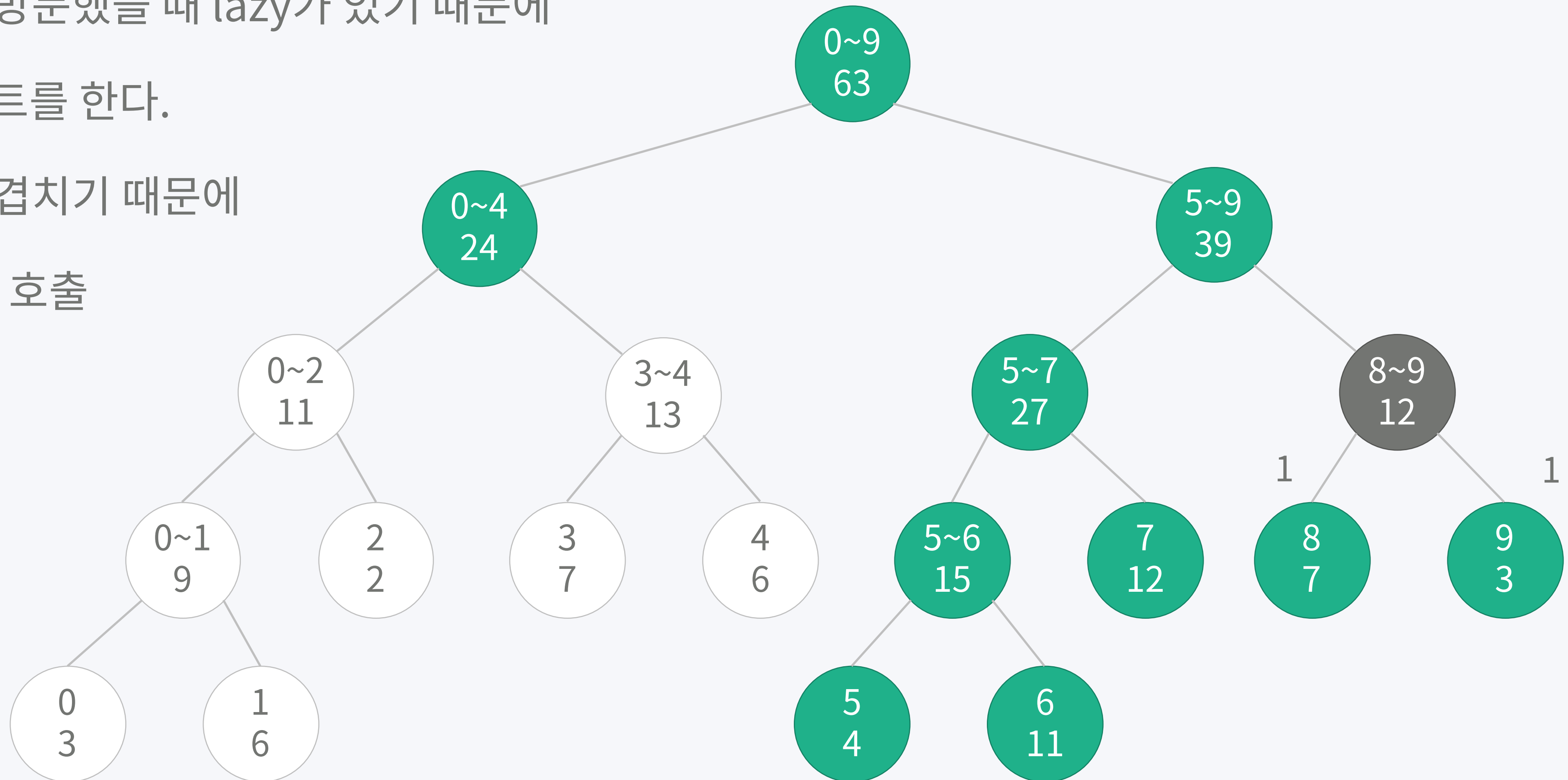


Lazy Propagation

52

구간의 합 구하기

- 8~9을 방문했을 때 lazy가 있기 때문에
- 업데이트를 한다.
- 6~8과 겹치기 때문에
- 두 자식 호출

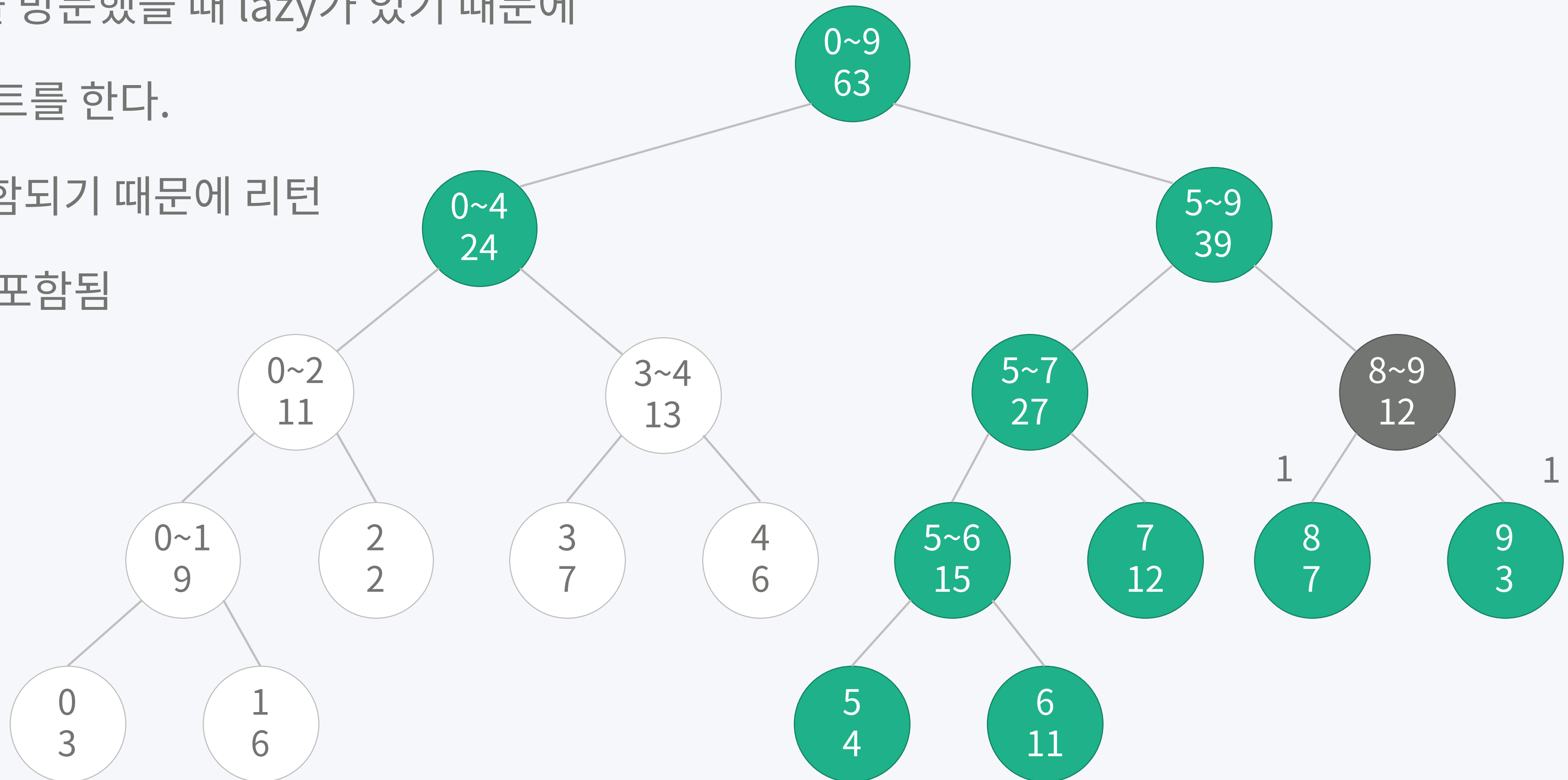


Lazy Propagation

53

구간의 합 구하기

- 8과 9를 방문했을 때 lazy가 있기 때문에
- 업데이트를 한다.
- 8은 포함되기 때문에 리턴
- 9는 안 포함됨



Lazy Propagation

구간의 합 구하기

```
void update_lazy(vector<long long> &tree, vector<long long> &lazy,
int node, int start, int end) {
    if (lazy[node] != 0) {
        tree[node] += (end-start+1) * lazy[node];
        // leaf가 아니면
        if (start != end) {
            lazy[node*2] += lazy[node];
            lazy[node*2+1] += lazy[node];
        }
        lazy[node] = 0;
    }
}
```

Handwritten notes:

- Under $(end-start+1)$: $\frac{end-start+1}{2}$
- Under $lazy[node]$: $\frac{1}{2}$
- Under $lazy[node*2]$: $\frac{1}{2}$
- Under $lazy[node*2+1]$: $\frac{1}{2}$

구간 합 구하기 2

55

<https://www.acmicpc.net/problem/10999>

- C/C++: <https://gist.github.com/Baekjoon/4175019338306283a180>

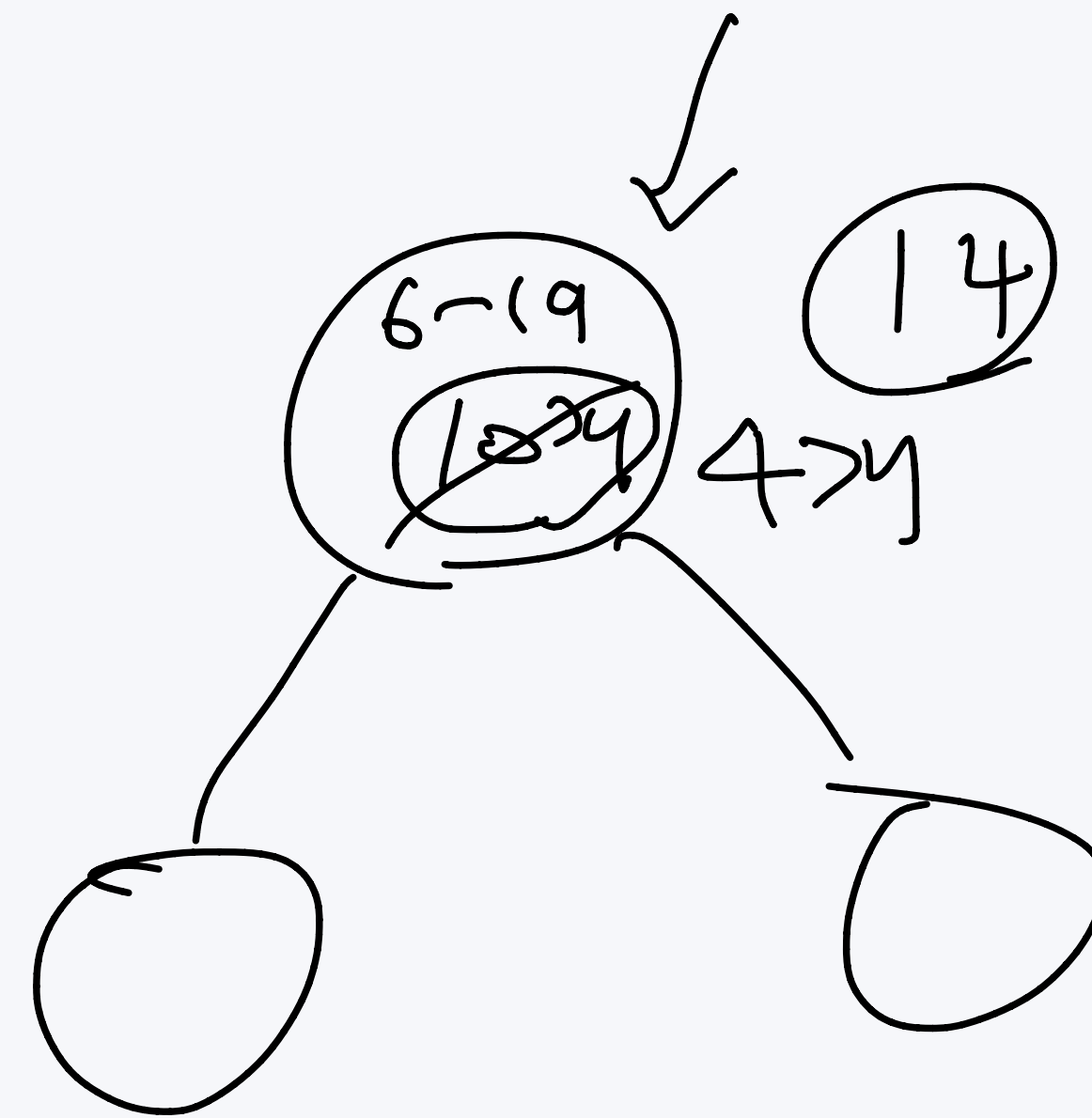
스위치

56

<https://www.acmicpc.net/problem/1395>

- C/C++: <https://gist.github.com/Baekjoon/247c11d7da24f577d74d>

~~10~~
~~2~~
243 ;



Fenwick Tree

Fenwick Tree

2진수 2진수

마지막 1

58

Fenwick Tree (BIT)

- i 의 마지막 자리: i 를 2진수로 나타냈을 때, 가장 마지막 1이 나타내는 값

- $3 = 11_2$ 1
- $5 = 101_2$ 1
- $6 = 110_2$ 2
- $8 = 1000_2$ 8
- $9 = 1001_2$ 1
- $10 = 1010_2$ 2
- $11 = 1011_2$
- $12 = 1100_2$ 4
- $16 = 10000_2$

Fenwick Tree

Fenwick Tree (BIT)

$\sim num + 1 = -num$

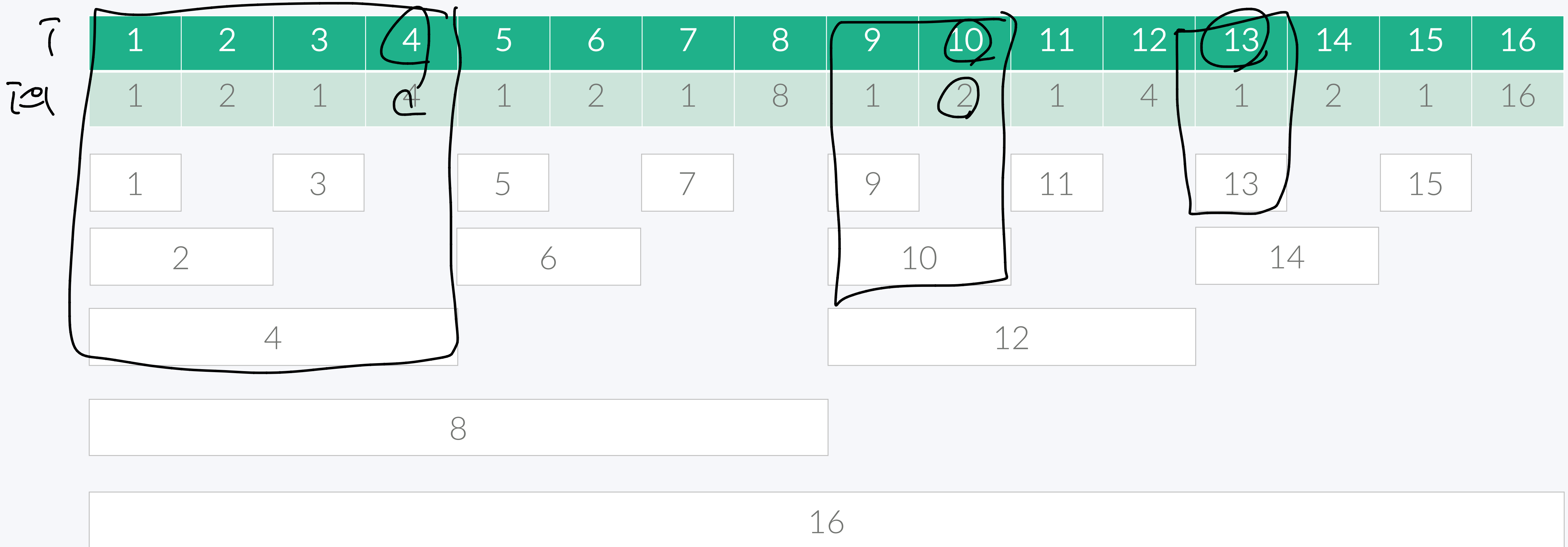
- $-num = \sim num + 1$
- $num = 100110101110101100000000000000$
- $\sim num = 0110010100010100111111111111$
- $-num = 011001010001010010000000000000$
- $num \& -num = 000000000000000000000000000000$

Fenwick Tree

Fenwick Tree (BIT)

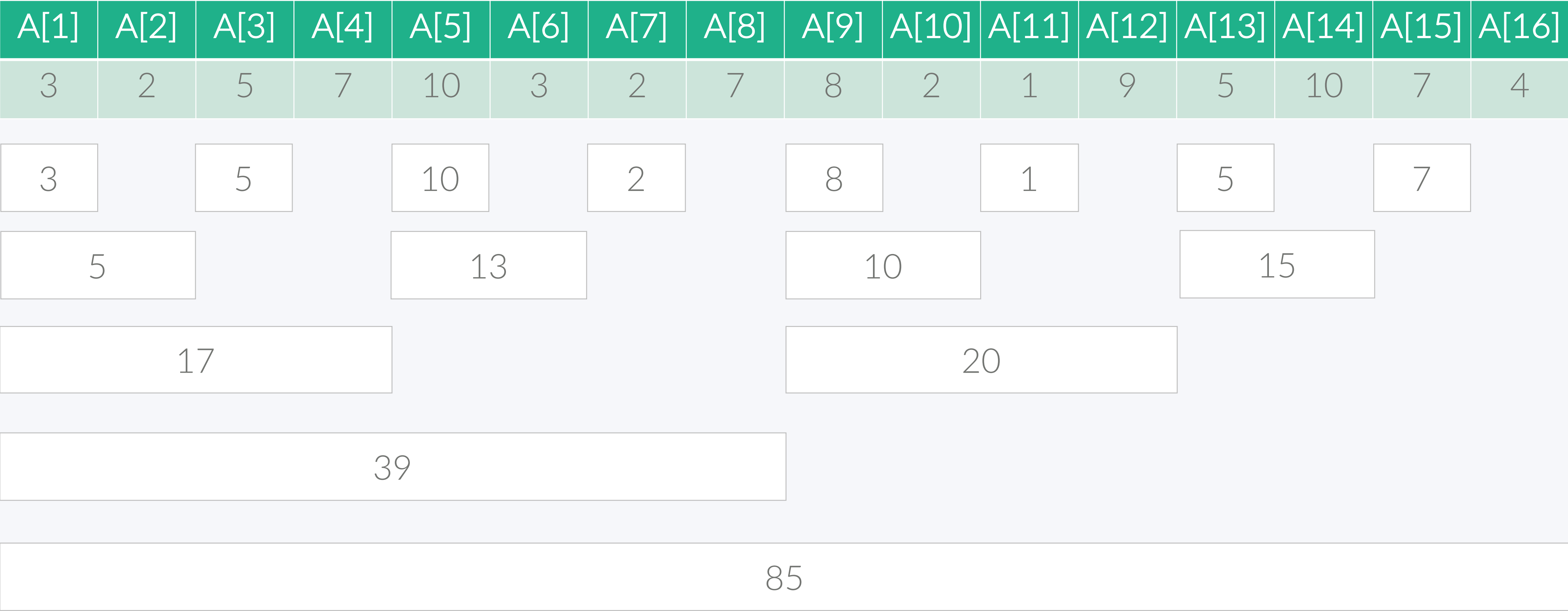
$tree[i] =$ [부터] 앞으로 다시 가게
 하는

60



Fenwick Tree

Fenwick Tree (BIT)



Fenwick Tree

Fenwick Tree (BIT)

- $A[1] + \dots + A[13]$ 을 구하려면
- $13 = 1101_2$ $1 \sim 12$
- $D[1101_2] + D[1100_2] + D[1000_2]$

$$13 = 1101_2$$

$$\text{tree}[13] = \underline{A[13] - A[12]}$$

$$12 = 1100_2$$

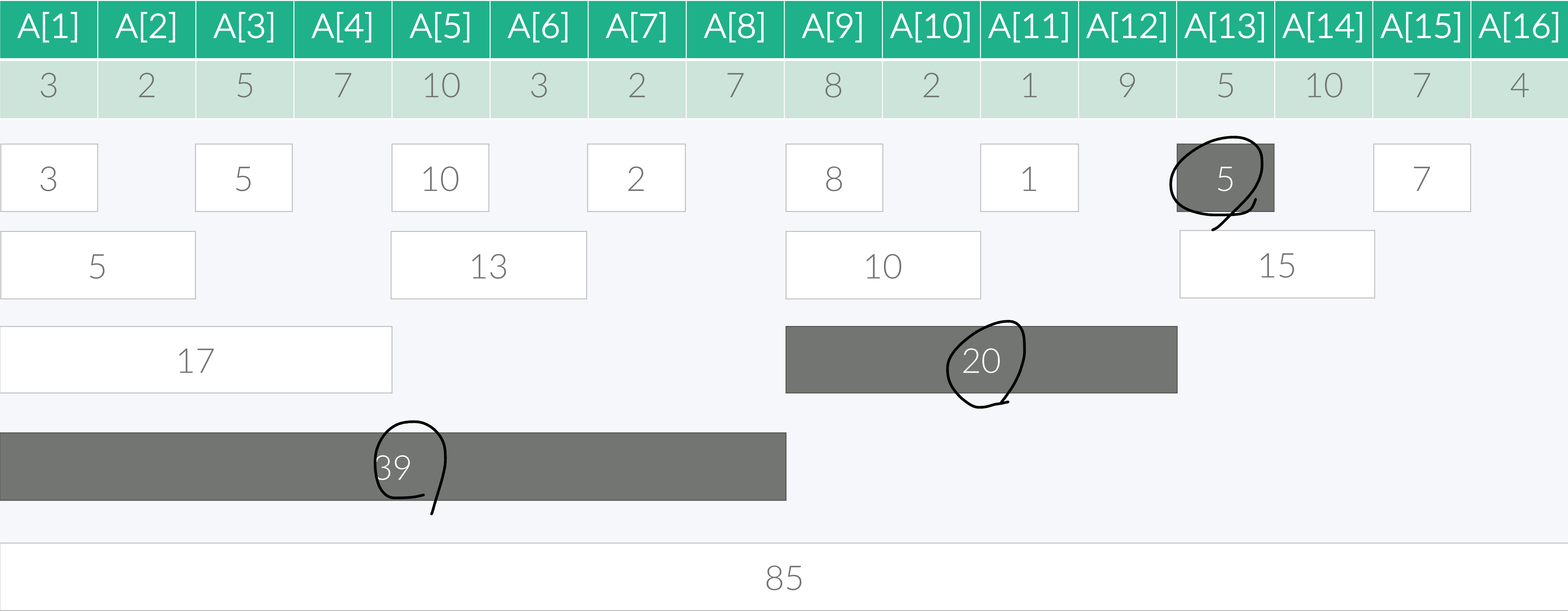
$$\text{tree}[12] = \underline{A[12] - A[8]}$$

$$8 = 1000_2$$

$$\text{tree}[8] = \underline{A[8] - A[4]}$$

Fenwick Tree

Fenwick Tree (BIT)



Fenwick Tree

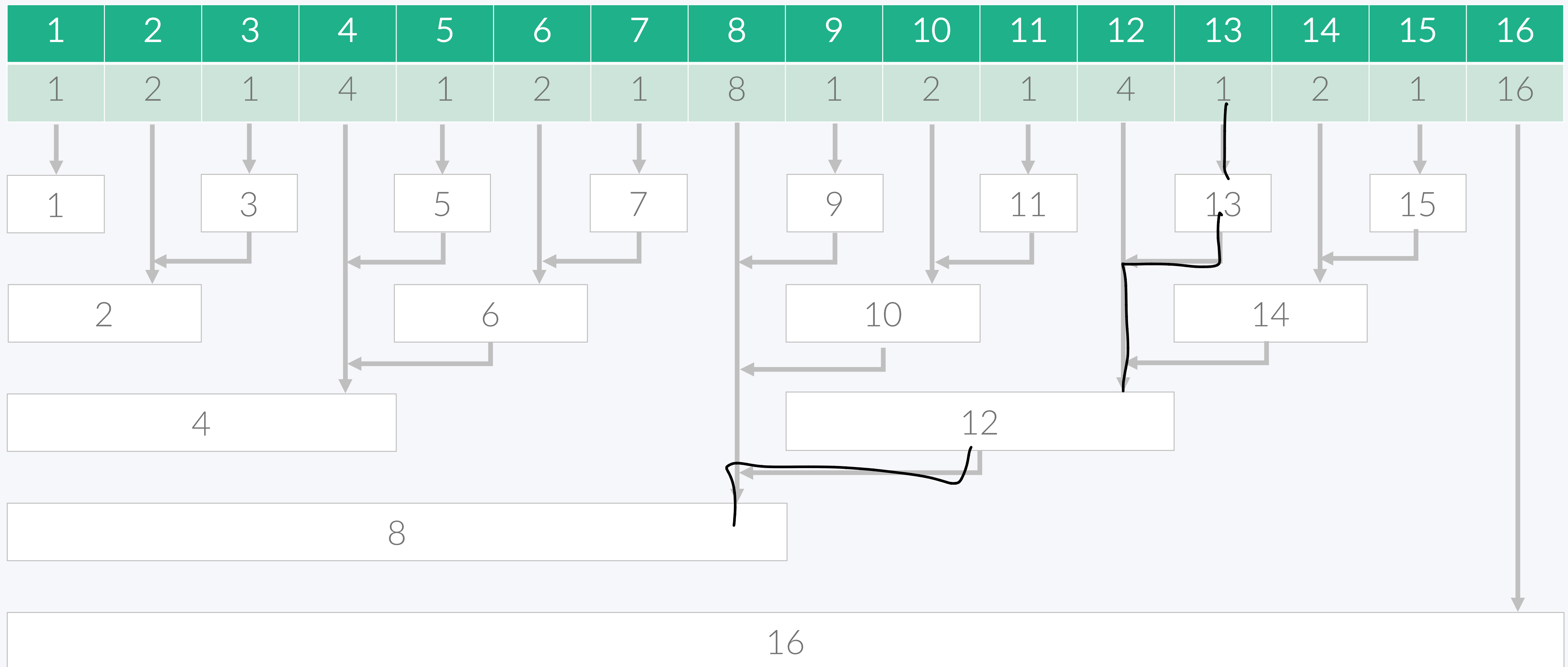
Fenwick Tree (BIT)

```
int sum(int i) {  
    int ans = 0;  
    while (i > 0) {  
        ans += d[i];  
        i -= (i & -i);  
    }  
    return ans;  
}
```


Fenwick Tree

65

Fenwick Tree (BIT)



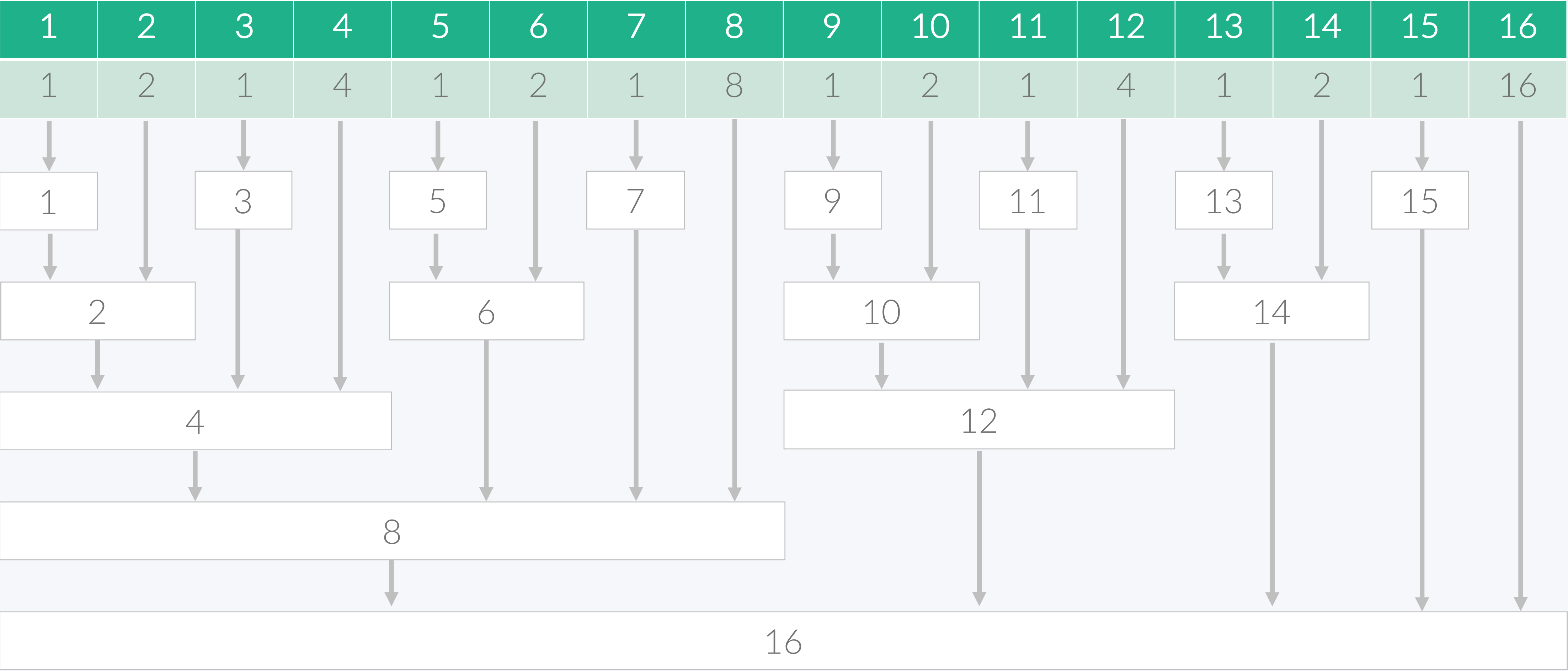
Fenwick Tree

Fenwick Tree (BIT)

```
int update(int i, int num) {  
    while (i <= n) {  
        d[i] += num;  
        i += (i & -i);  
    }  
}
```

Fenwick Tree

Fenwick Tree (BIT)



구간 합 구하기

<https://www.acmicpc.net/problem/2042>

- C/C++: <https://gist.github.com/Baekjoon/20b134a15afb54eddb79>
- C/C++: <https://gist.github.com/Baekjoon/fcf34370c7c51c436647b8266f8dab95>

2D Fenwick Tree

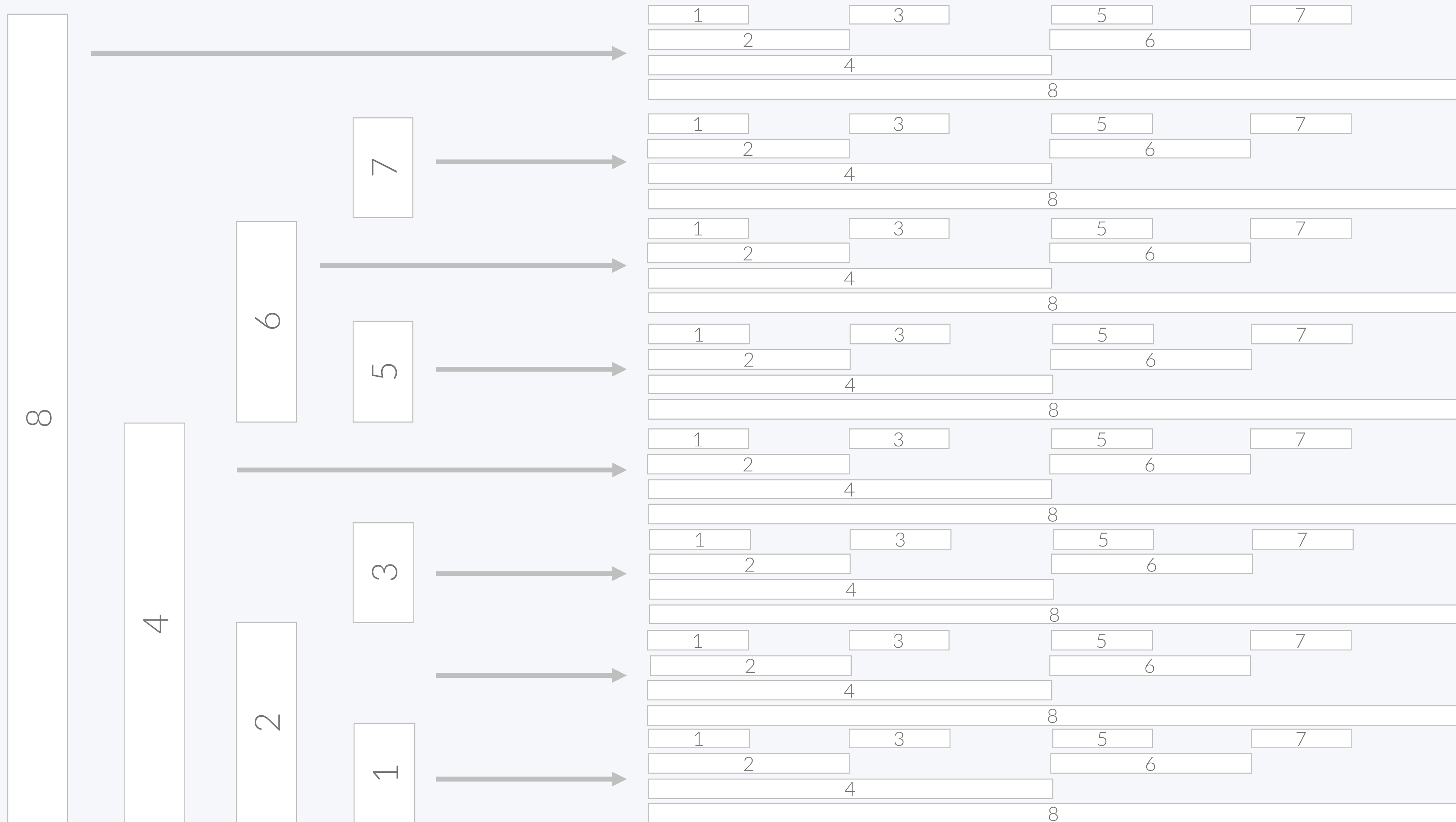
2D Fenwick Tree (BIT)

- 1차원을 2차원으로 확장해서 풀면 된다.
- x 에 대해서 그리고 y 에 대해서 트리를 만들면 된다

2D Fenwick Tree

70

2D Fenwick Tree (BIT)



2D Fenwick Tree

2D Fenwick Tree (BIT)

```
void update(int x, int y, int val) {  
    for (int i=x; i<=n; i+=i&-i) {  
        for (int j=y; j<=n; j+=j&-j) {  
            tree[i][j] += val;  
        }  
    }  
}
```

$1 \sim N$

$\text{for } (i=0; i < N; i++)$
 $\text{for } (j=0; j < M; j++)$
 $\text{Sum} += A[i][j]$

2D Fenwick Tree

2D Fenwick Tree (BIT)

```
int sum(int x, int y) {  
    int ans = 0;  
    for (int i=x; i>0; i-=i&-i) {  
        for (int j=y; j>0; j-=j&-j) {  
            ans += tree[i][j];  
        }  
    }  
    return ans;  
}
```


구간 합 구하기 3

73

<https://www.acmicpc.net/problem/11658>

- C/C++: <https://gist.github.com/Baekjoon/14778526b16b31a329e6>