RMQ

최백준 choi@startlink.io

RMQ

구간의 최소값 구하기

Range Minimum Query

- 배열 A[1], …, A[N]가 있고, 다음과 같은 연산을 수행해야 한다.
  - 최소값: A[i], …, A[j] 중에서 최소값을 찾아 출력한다.
- 이러한 연산이 총 M개 주어진다.

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

다 해보기

다 해보기

구간의 최소값 구하기 (RMQ)

- 최소값: A[i], …, A[j] 중에서 최소값을 찾아 출력한다. O(N)

```
int m = a[i];
for (int k=i; k<=j; k++) {
    if (m > a[k]) {
        m = a[k];
    }
}
```

다 해보기

구간의 최소값 구하기 (RMQ)

- 최소값을 하나 구하는데 O(N) 시간이 걸린다.
- 쿼리의 개수는 총 M개이기 때문에, O(MN) 시간이 필요하다.

저장하기

저장하기

구간의 최소값 구하기 (RMQ)

- D[i][j] = A[i] ~ A[j]중 최소값을 저장한다. (i ≤ j)
- D[i][i] = A[i]
- D[i][j] = min(D[i][j-1], A[i])

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

i \ j	0	1	2	3	4	5	6
0	3	3	2	2	2	1	1
1		6	2	2	2	1	1
2			2	2	2	1	1
3				5	3	1	1
4					3	1	1
5						1	1
6							8

저장하기

구간의 최소값 구하기 (RMQ)

- 최소값을 저장할 N^2 크기의 배열이 하나 필요하다.
- 처음 O(N^2) 시간 동안 배열을 채워놓으면
- 최소값을 구하는데 걸린 시간은 O(1) 이다.

```
int d[N][N];
int a[N];
int n;
for (int i=0; i<n; i++) {
    d[i][i] = a[i];
    for (int j=i+1; j<n; j++) {
        d[i][j] = min(d[i][j-1], a[j]);
    }
}
```

루트N으로 나누기

루트N으로 나누기

구간의 최소값 구하기 (RMQ)

- 영어로는 sqrt decomposition 이라고 한다.
- R= 루트 N이라고 했을 때
- A를 R개의 그룹으로 나눈 다음에, Group[i]에 i번 그룹의 최소값을 저장하는 방식

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

그룹	0	1	2	3

루트N으로 나누기

구간의 최소값 구하기 (RMQ)

- 영어로는 sqrt decomposition 이라고 한다.
- R = 루트 N이라고 했을 때
- A를 R개의 그룹으로 나눈 다음에, Group[i]에 i번 그룹의 최소값을 저장하는 방식

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

Group[0] = 2	Group[1] = 1	Group[2] = 7	Group[3] = 3

루트N으로 나누기

구간의 최소값 구하기 (RMQ)

- 영어로는 sqrt decomposition 이라고 한다.
- R = 루트 N이라고 했을 때
- A를 R개의 그룹으로 나눈 다음에, Group[i]에 i번 그룹의 최소값을 저장하는 방식

```
for (int i=0; i<n; i++) {
    if (i%r == 0) {
        group[i/r] = a[i];
    } else {
        group[i/r] = min(group[i/r], a[i]);
    }
}
```

루트N으로 나누기

구간의 최소값 구하기 (RMQ)

- 최소값을 구하는 쿼리 i, j는 두 가지 경우가 있다. (i ≤ j)

1. i와 j가 같은 그룹인 경우
2. i와 j의 그룹이 다른 경우

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

G[0] = 2	G[1] = 1	G[2] = 7	G[3] = 3

루트N으로 나누기

구간의 최소값 구하기 (RMQ)

- 최소값을 구하는 쿼리 i, j는 두 가지 경우가 있다. (i ≤ j)

1. i와 j가 같은 그룹인 경우

- 이 경우에는 그냥 for loop를 이용해서 최소값을 구하면 된다.
- 그룹에 들어있는 수의 개수는 루트N이기 때문에, 총 걸리는 시간은 O(루트N) 이다.

```
for (int i=start; i<=end; i++) {
    ans = min(ans, a[i]);
}
```

루트N으로 나누기

구간의 최소값 구하기 (RMQ)

- 최소값을 구하는 쿼리 i, j는 두 가지 경우가 있다. ($i \leq j$)

2. i와 j가 다른그룹인 경우

- 이런 경우에는 3가지로 나눌 수 있다.
  - i가 들어있는 그룹
  - j가 들어있는 그룹
  - i와 j 사이에 들어있는 그룹

루트N으로 나누기

구간의 최소값 구하기 (RMQ)

- A[1] ~ A[9]의 최소값을 구하는 경우
- N = 11
- R = 루트N = 3
- 1의 그룹: 1/R = 1/3 = 0
- 9의 그룹: 9/R = 9/3 = 3

- 시작 그룹에 들어있는 수의 개수는 R개
- 끝 그룹에 들어있는 수의 개수는 R개
- 시작과 끝 그룹 사이에 있는 그룹의 수는 R개
- 아래 표시된 값을 비교해야 한다.

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

G[0] = 2	G[1] = 1	G[2] = 7	G[3] = 3

루트N으로 나누기

구간의 최소값 구하기 (RMQ)

```
while (true) {
    ans = min(ans, a[start]);
    start += 1;
    if (start % r == 0) {
        break;
    }
}
```

```
while (true) {
    ans = min(ans, a[end]);
    end -= 1;
    if (end % r == r-1) {
        break;
    }
}
for (int i=start/r; i<=end/r; i++) {
    ans = min(ans, group[i]);
}
```

루트N으로 나누기

구간의 최소값 구하기 (RMQ)

- 총 O(3루트N)의 시간이 걸리게 된다.

다이나믹 프로그래밍

다이나믹 프로그래밍

구간의 최소값 구하기 (RMQ)

- D[i][j] = A[i]부터 2^j개의 최소값

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

다이나믹 프로그래밍

구간의 최소값 구하기 (RMQ)

- D[i][j] = A[i]부터 2^j개의 최소값
- A[i]부터 2^j개의 최소값은
- A[i]부터 2^(j-1)개의 최소값과 A[i+2^(j-1)]부터 2^(j-1)개의 최소값과 같다.

<table>
<tr><td colspan="2">D[i][j]</td></tr>
<tr><td>D[i][j-1]</td><td>D[i+(1<<(j-1))][j-1]</td></tr>
</table>

- D[i][j] = min(D[i][j-1], D[i+(1<<(j-1))][j-1]

다이나믹 프로그래밍

구간의 최소값 구하기 (RMQ)

- 1부터 6까지 최소값 구하기

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

1~6까지 최소값

D[1][2]

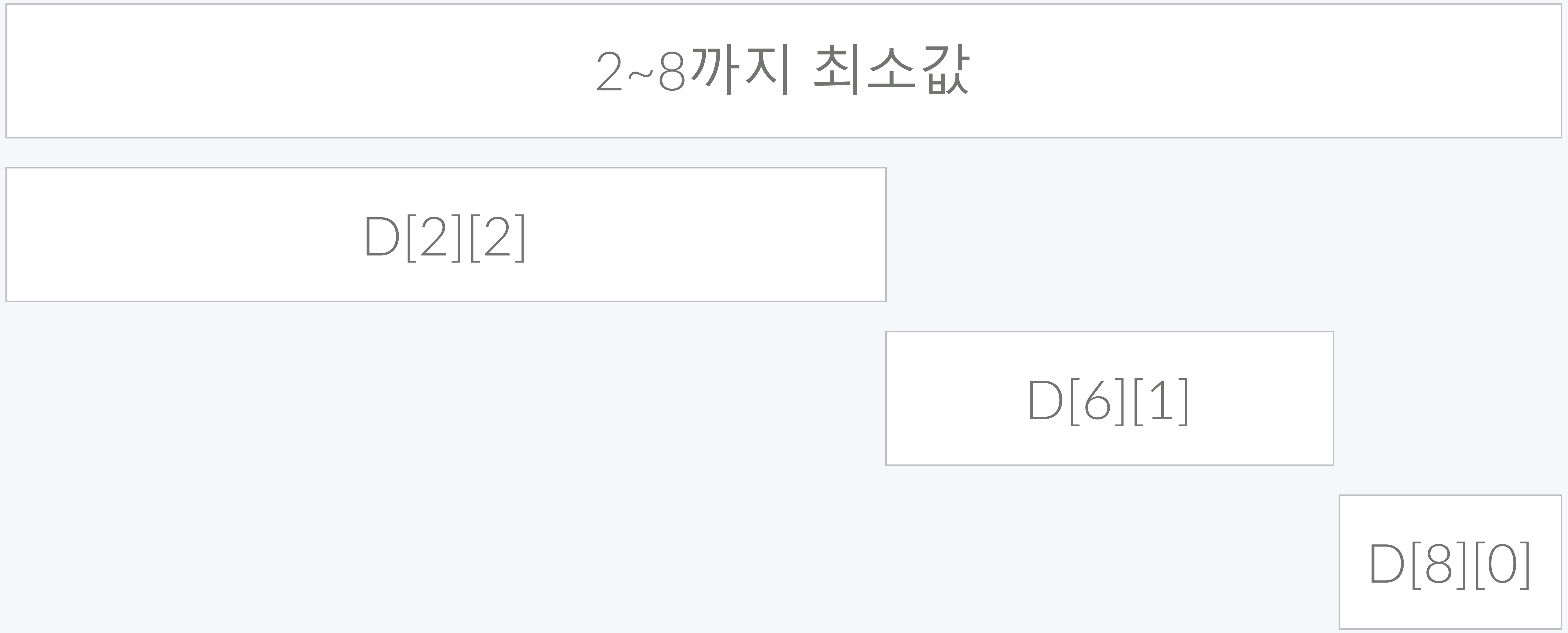
D[5][1]

다이나믹 프로그래밍

구간의 최소값 구하기 (RMQ)

- 2부터 8까지 최소값 구하기

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



다이나믹 프로그래밍

구간의 최소값 구하기 (RMQ)

- 공간: O(NlgN)
- 선처리: O(NlgN)
- 최소값 구하는 시간: O(lgN)

다이나믹 프로그래밍

구간의 최소값 구하기 (RMQ)

```
for (int i=0; i<n; i++) {
    d[i][0] = a[i];
}
for (int j=1; j<17; j++) {
    for (int i=0; i<n; i++) {
        if (i+(1<<j)-1 < n) {
            d[i][j] = min(d[i][j-1], d[i+(1<<(j-1))][j-1]);
        } else {
            break;
        }
    }
}
```

다이나믹 프로그래밍

구간의 최소값 구하기 (RMQ)

```
int ans = a[start];
int k = 16;
while (start <= end && k >= 0) {
    if (start + (1<<k) - 1 <= end) {
        ans = min(ans, d[start][k]);
        start += (1<<k);
    }
    k -= 1;
}
```

- 위의 소스에서 16은 문제의 N 제한이 10만이기 때문에, 정한 값이다.
- 2^16 = 65536 이라, 2^17 크기를 가지는 경우는 없기 때문

다이나믹 프로그래밍

구간의 최소값 구하기 (RMQ)

- 쿼리는 O(1)만에 구할 수 있다.
- k = lg길이 = lg(j-i+1) 로 한다면
- 정답은 min(D[i][k], D[j-(1<<k)+1][k]) 으로 구할 수 있다. (최소값은 겹쳐도 되기 때문)
- 아래 그림은 1~6까지 최소값을 한 번에 구하는 경우
- lg(6-1+1) = 2

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

1~6까지 최소값

D[1][2]

D[3][2]

다이나믹 프로그래밍

구간의 최소값 구하기 (RMQ)

- 쿼리는 O(1)만에 구할 수 있다.
- k = lg길이 = lg(j-i+1) 로 한다면
- 정답은 min(D[i][k], D[j-(1<<k)+1][k]) 으로 구할 수 있다. (최소값은 겹쳐도 되기 때문)
- 아래 그림은 2~8까지 최소값을 한 번에 구하는 경우
- lg(8-2+1) = 2

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

2~8까지 최소값

D[2][2]

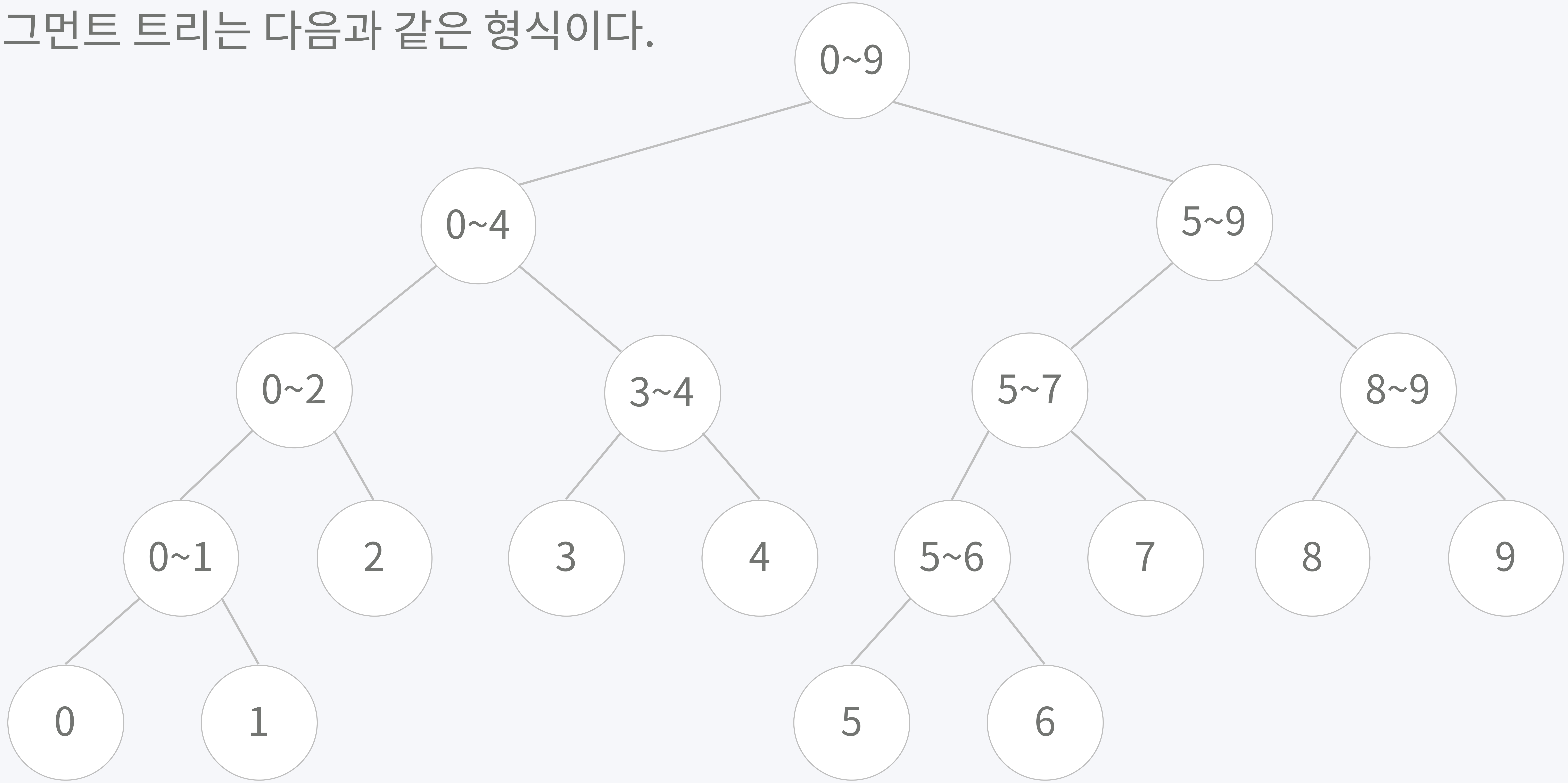
D[5][2]

세그먼트 트리

세그먼트 트리

구간의 최소값 구하기 (RMQ)

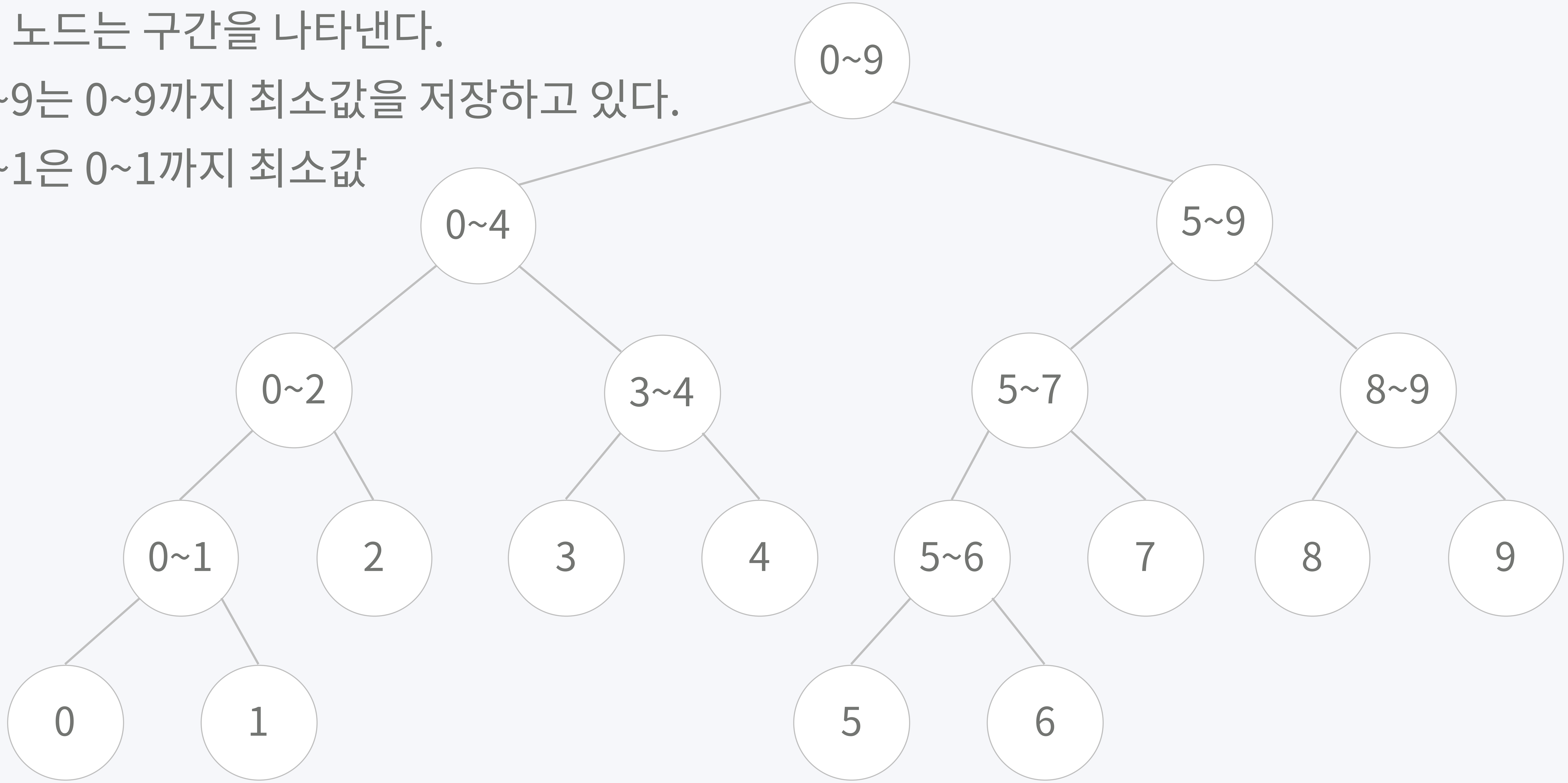
- 세그먼트 트리는 다음과 같은 형식이다.



세그먼트 트리

구간의 최소값 구하기 (RMQ)

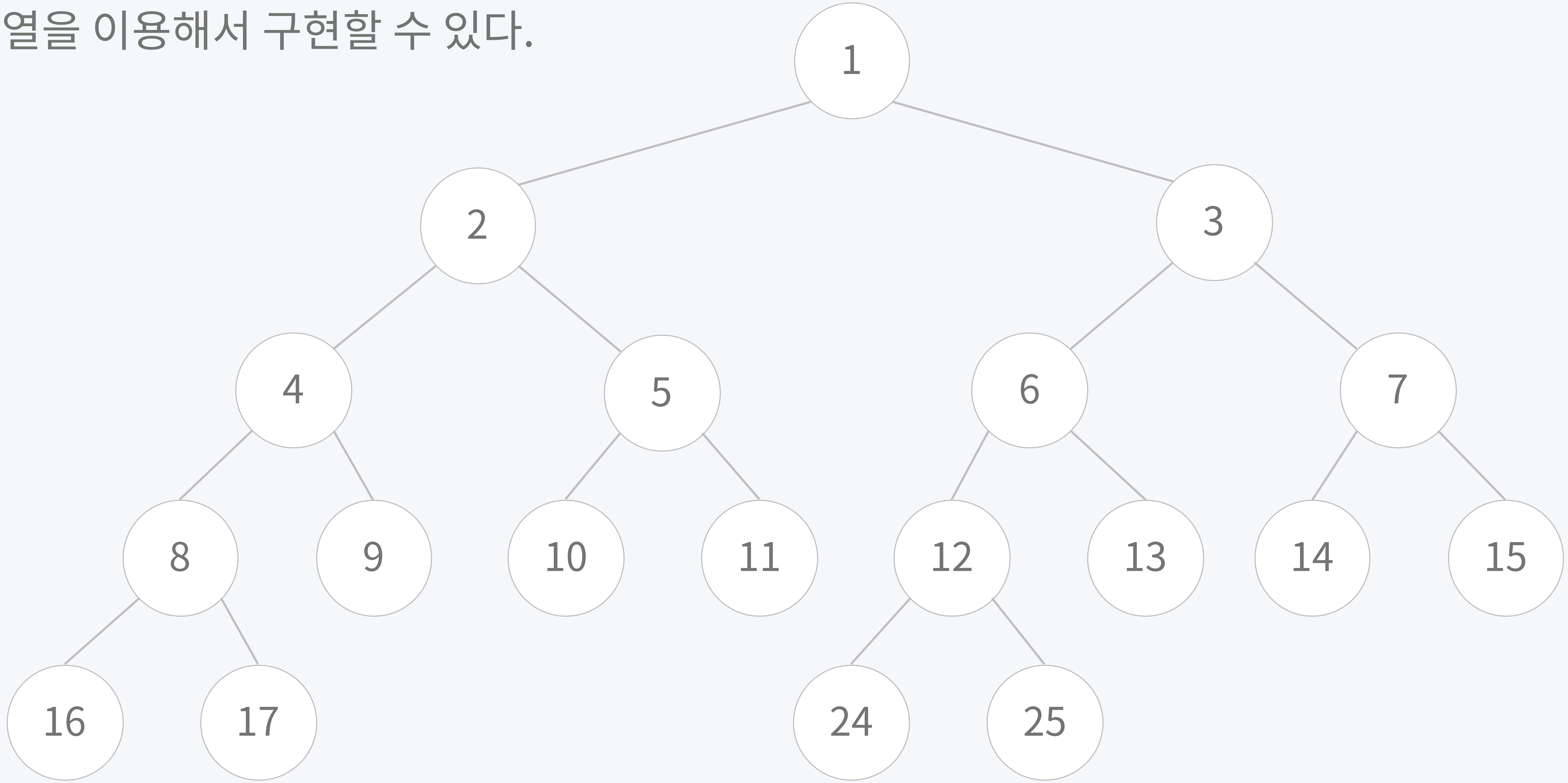
- 각 노드는 구간을 나타낸다.
- 0~9는 0~9까지 최소값을 저장하고 있다.
- 0~1은 0~1까지 최소값



세그먼트 트리

구간의 최소값 구하기 (RMQ)

- 배열을 이용해서 구현할 수 있다.



세그먼트 트리

구간의 최소값 구하기 (RMQ)

- N이 2의 제곱꼴인 경우에는 Full Binary Tree
- 리프 노드가 N개인 Full Binary Tree: 필요한 노드의 개수: 2N-1
- 아닌 경우에는 높이 H = [lgN] 이다.
- 필요한 배열의 크기: 2^(H+1)

세그먼트 트리

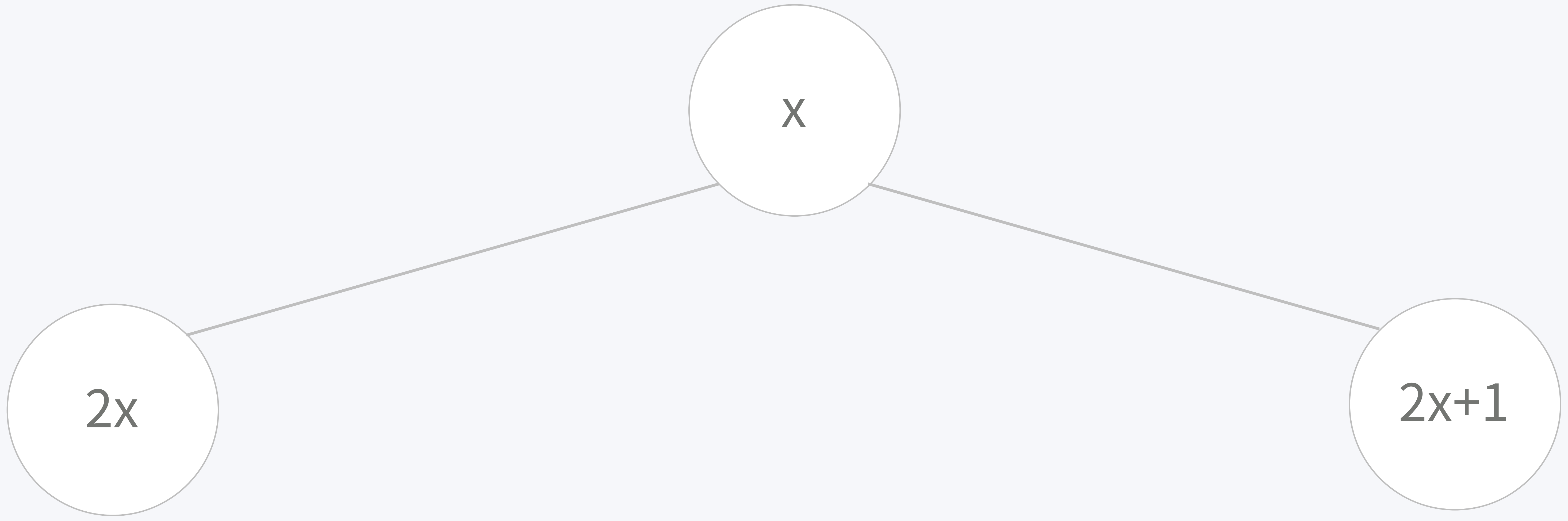
구간의 최소값 구하기 (RMQ)

```
void init(int node, int start, int end) {
    if (start == end) {
        tree[node] = a[start];
    } else {
        init(node*2, start, (start+end)/2);
        init(node*2+1, (start+end)/2+1, end);
        tree[node] = min(tree[node*2],tree[node*2+1]);
    }
}
```

세그먼트 트리

구간의 최소값 구하기 (RMQ)

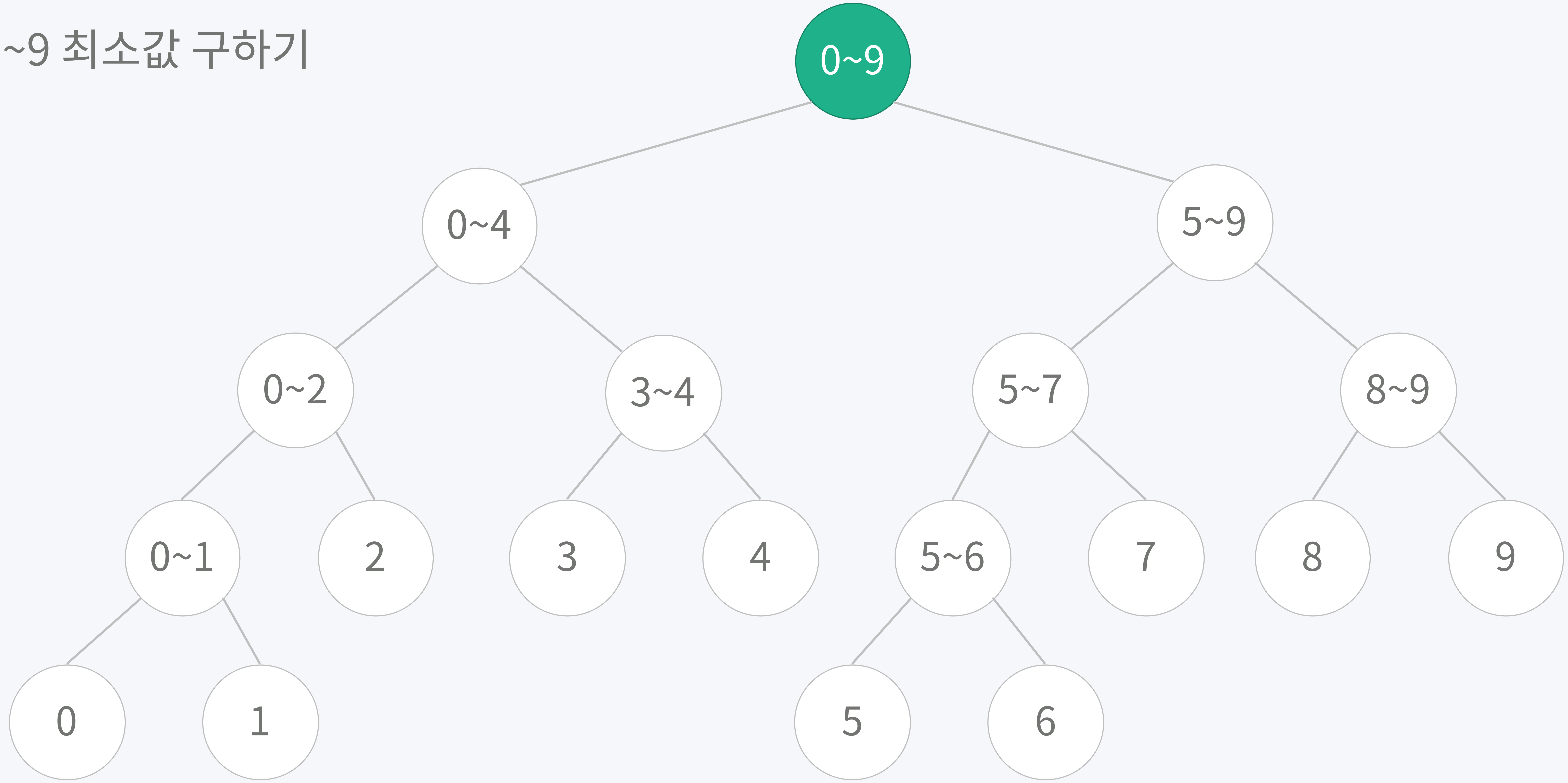
- start == end인 경우에는 리프 노드
- 노드 node의 왼쪽 자식: 2*node, 오른쪽 자식: 2*node+1
- 어떤 노드가 [start, end]를 담당한다면
- 왼쪽 자식: [start, (start+end)/2], 오른쪽 자식: [(start+end)/2+1, end] 를 담당해야 한다



세그먼트 트리

구간의 최소값 구하기 (RMQ)

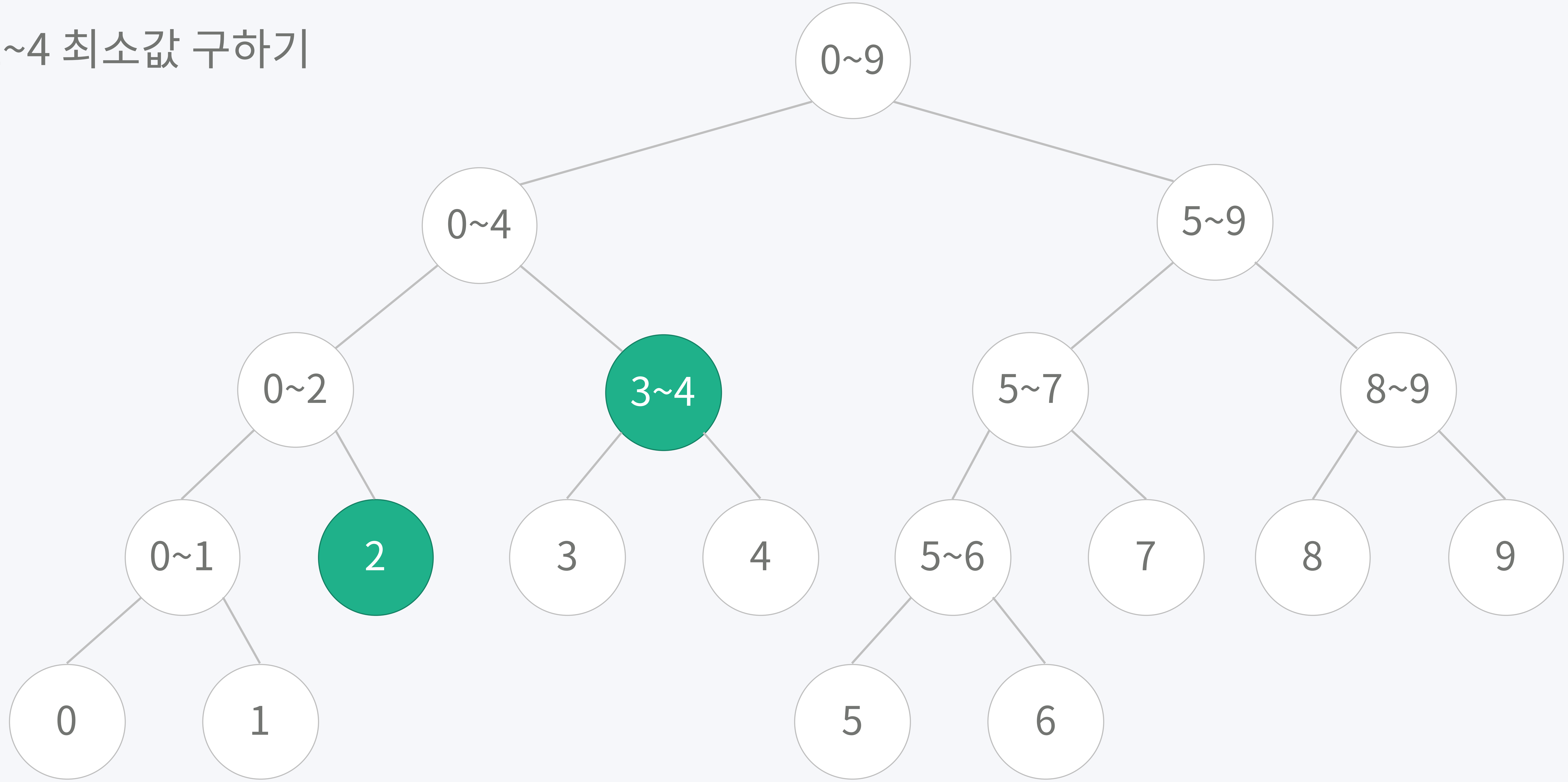
- 0~9 최소값 구하기



세그먼트 트리

구간의 최소값 구하기 (RMQ)

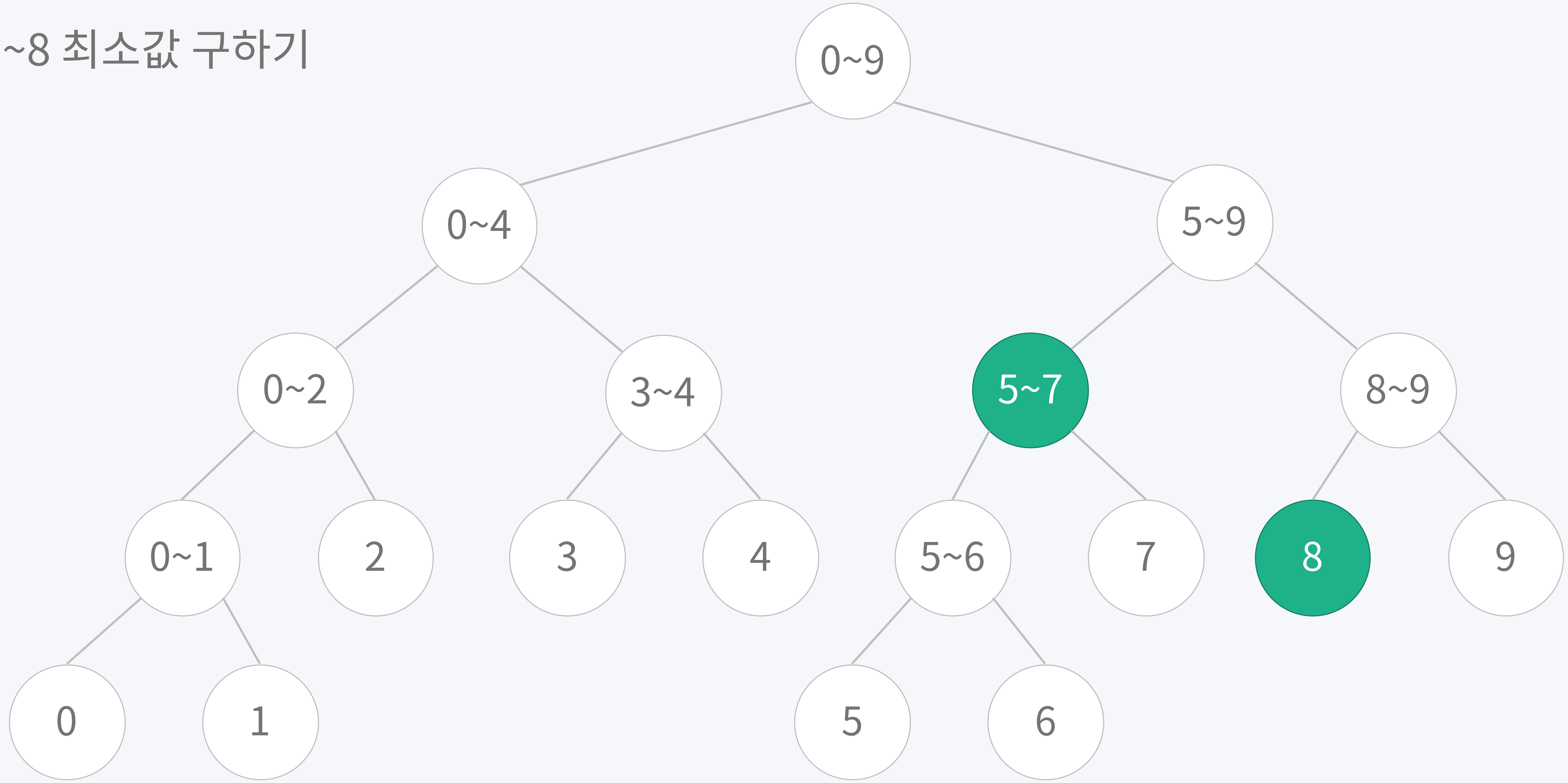
- 2~4 최소값 구하기



세그먼트 트리

구간의 최소값 구하기 (RMQ)

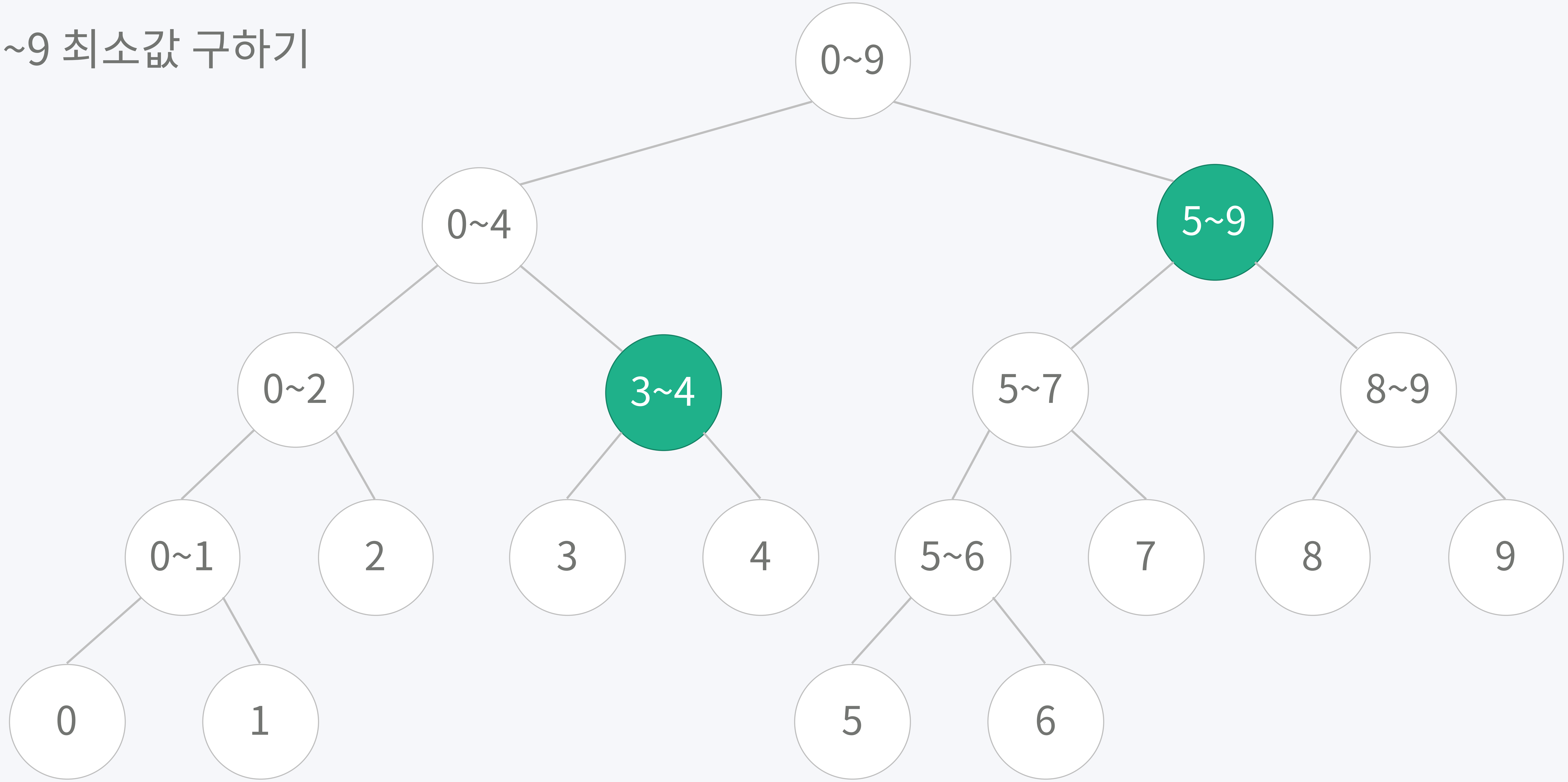
- 5~8 최소값 구하기



세그먼트 트리

구간의 최소값 구하기 (RMQ)

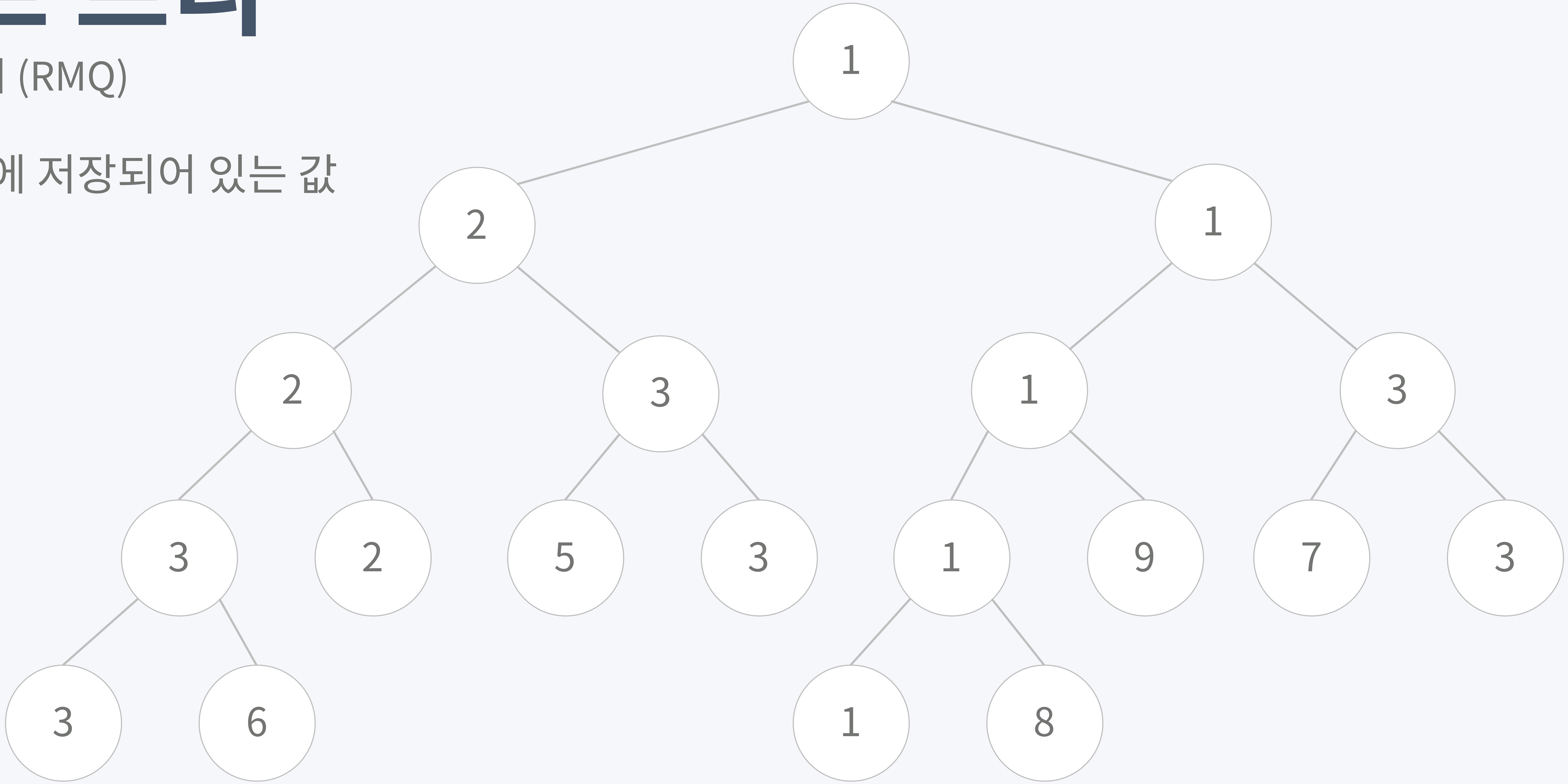
- 3~9 최소값 구하기



세그먼트 트리

구간의 최소값 구하기 (RMQ)

- 세그먼트 트리에 저장되어 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

세그먼트 트리

구간의 최소값 구하기 (RMQ)

```
int query(int node, int start, int end, int i, int j) {
    if (i > end || j < start)
        return -1;
    if (i <= start && end <= j)
        return d[node];
    int m1 = query(2*node, start, (start+end)/2, i, j);
    int m2 = query(2*node+1, (start+end)/2+1, end, i, j);
    if (m1 == -1) return m2;
    else if (m2 == -1) return m1;
    else return min(m1, m2);
}
```

최소값

https://www.acmicpc.net/problem/10868

- 세그먼트 트리를 이용해서 RMQ를 구현하는 문제

최소값과 최대값

https://www.acmicpc.net/problem/2357

- 세그먼트 트리를 이용해서 RMQ를 구현하는 문제

Sliding Window

최소값 찾기

https://www.acmicpc.net/problem/11003

- N개의 수 A[1], A[2], …, A[N]이 주어졌을 때
- D[i] = A[i-L+1] ~ A[i] 중의 최소값이라고 할 때, D를 구하는 문제
- $1 \leq L \leq N \leq 5{,}000{,}000$

A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]	A[11]	A[12]
1	5	2	3	6	2	3	7	3	5	2	6

D[1] = 1	D[4] = 2	D[7] = 2	D[10] = 3
D[2] = 1	D[5] = 2	D[8] = 2	D[11] = 2
D[3] = 1	D[6] = 2	D[9] = 3	D[12] = 2

최소값 찾기

https://www.acmicpc.net/problem/11003

- 덱을 이용해서 푼다.
- 덱에는 값과 인덱스를 저장한다.
- 덱에는 최대 L개의 값이 저장 된다.
- 덱에 들어있는 값은 항상 증가하는 순서대로 저장되어 있다.

최소값 찾기

https://www.acmicpc.net/problem/11003

- 덱은 값이 증가하는 순서대로 저장되어 있기 때문에, 가장 앞에 있는 값이 최소값이 된다.
- 먼저, 가장 앞에 있는 값의 인덱스와 현재 값의 인덱스가 L보다 많이 차이나는지 검사해야 한다.
- 그 다음에는, 가장 뒤에 있는 값이 현재 값보다 큰지 검사해서 크면 덱에서 뺀다. (반복)
- 이제 현재 값을 덱에 넣는다.

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

최소값 찾기

https://www.acmicpc.net/problem/11003

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

값	1
인덱스	0

값	1	5
인덱스	0	1

값	1	2
인덱스	0	2

값	2	3
인덱스	2	3

값	2	3	6
인덱스	2	3	4

값	2
인덱스	5

값	2	3
인덱스	5	6

값	2	3	7
인덱스	5	6	7

값	3	3
인덱스	6	8

값	3	5
인덱스	8	9

값	2
인덱스	10

값	2	6
인덱스	10	11

최소값 찾기

https://www.acmicpc.net/problem/11003

```
deque<pair<int,int>> d;
vector<int> ans(n);
for (int i=0; i<n; i++) {
    int cur = a[i];
    if (!d.empty() && d.front().second <= i-l) {
        d.pop_front();
    }
    while (!d.empty() && d.back().first > cur) {
        d.pop_back();
    }
    d.push_back(make_pair(cur, i));
    ans[i] = d.front().first;
}
```