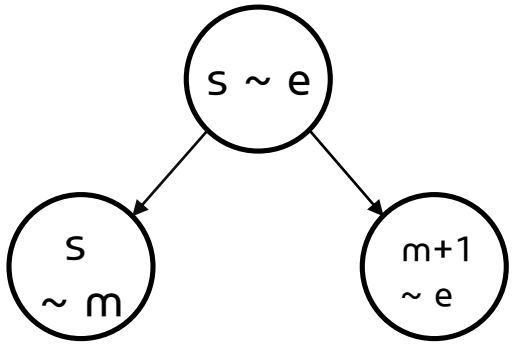
세그먼트 트리(Segment Tree)

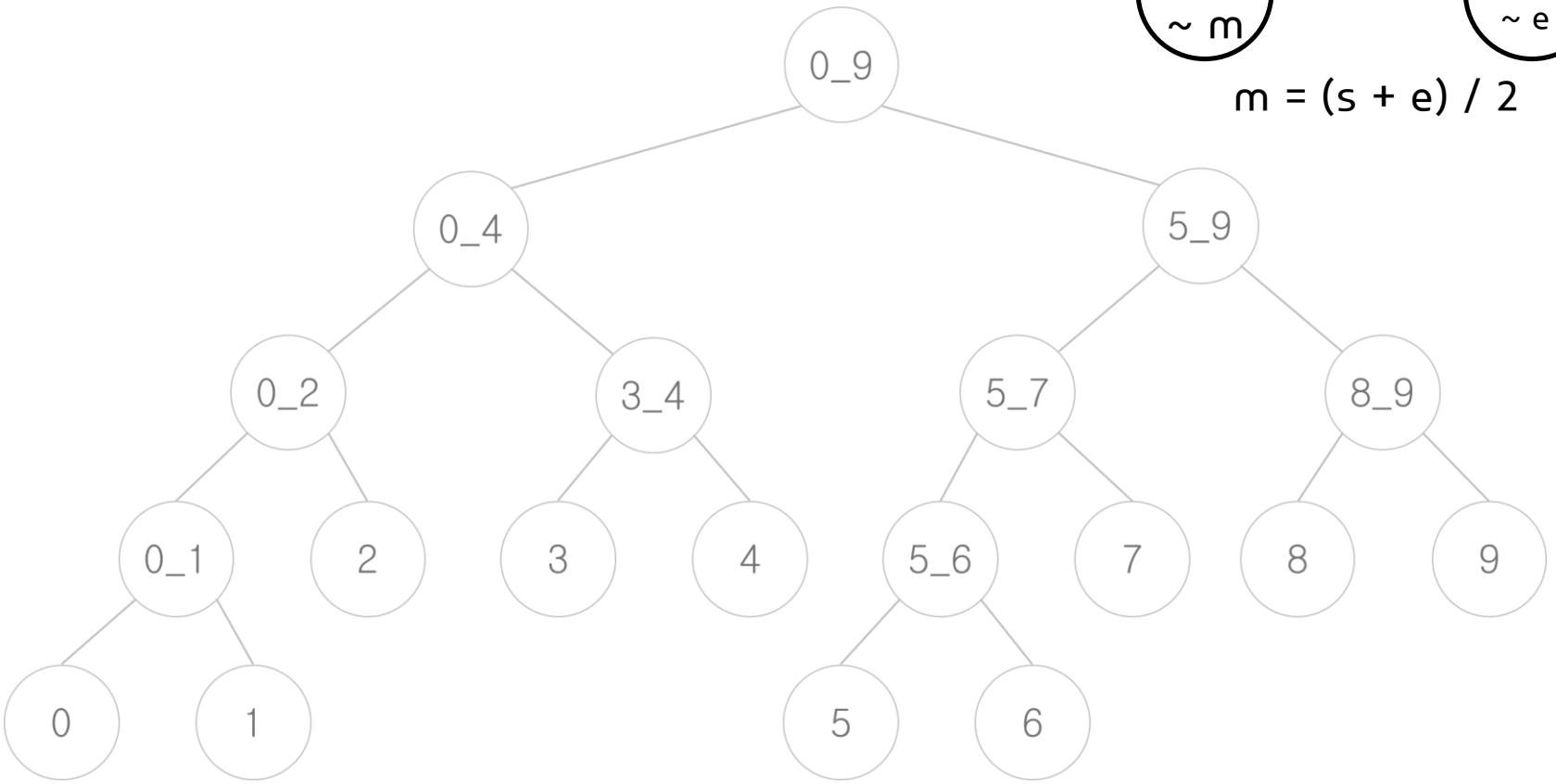
세그먼트 트리

Range Minimum Query

- 세그먼트 트리는 다음과 같은 형식입니다.



m = (s + e) / 2

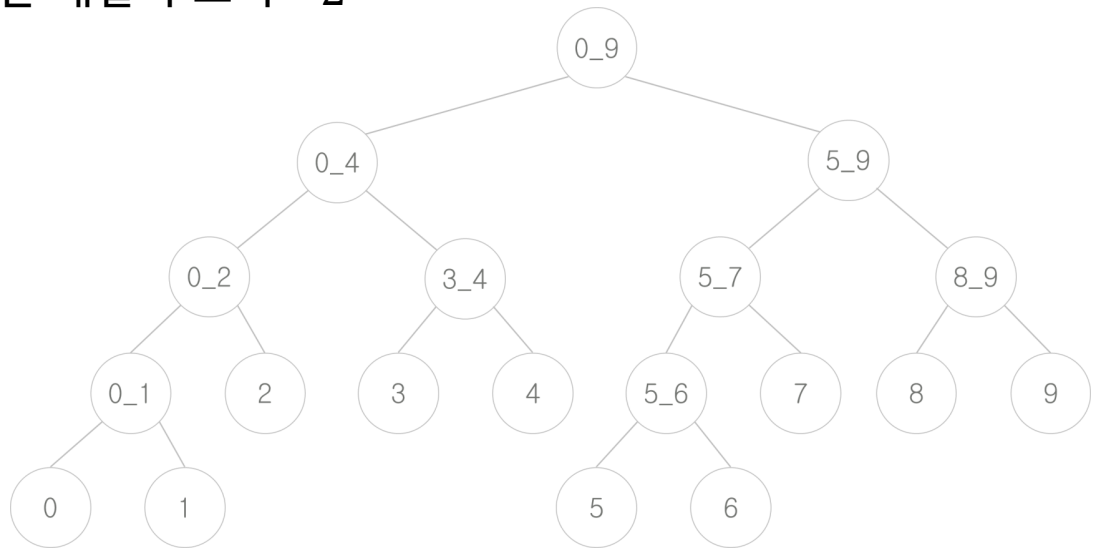


Gaussian

세그먼트 트리

Range Minimum Query

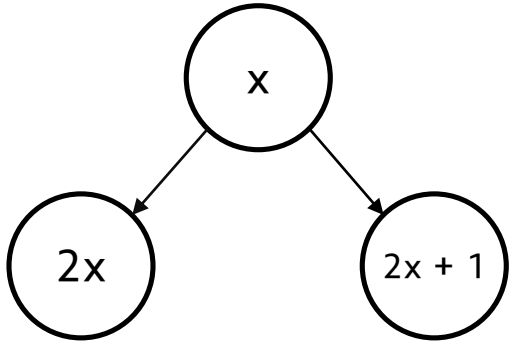
- 각 노드는 구간을 나타냅니다.
- 0 ~ 9는 0 ~9 까지의 최소값을 저장하고 있습니다. 0 ~ 1은 0 ~ 1까지의 최솟값
- 노드는 3가지 인수를 통하여 정보를 저장합니다.
  ① 저장된 칸의 번호
  ② 구간 : 시작
  ③ 구간 : 끝
- N이 2의 제곱 꼴인 경우 Full Binary Tree 이므로 높이 H = $\lceil lgN \rceil$ 입니다.
  이 때 필요한 배열의 크기 : $2^{(H+1)}$



Gaussian

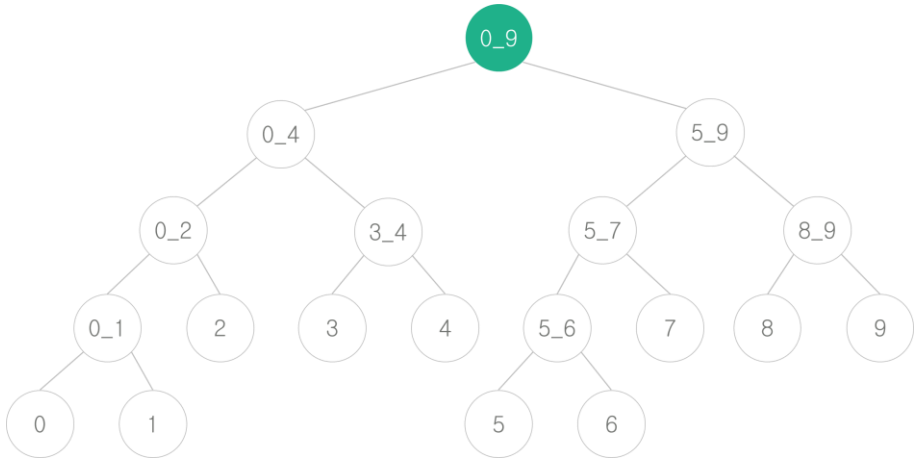
세그먼트 트리

Range Minimum Query



- start == end 인 경우에는 리프 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]
- ※ 노드의 범위가 쿼리의 범위에 완전히 포함? 완전히 미포함 되는가?
  ① 노드의 범위[start, end]가 쿼리의 [i, j] 범위의 부분집합 이라면 구간의 최솟값을 리턴
  ② 노드의 범위[start, end]가 쿼리의 [i, j] 범위와 배타적 이라면 불가능한 값 리턴
  ③ 그 외의 경우 왼쪽/오른쪽 노드를 나눈 후 분할하여 구간의 최솟값 확인

- 0 ~ 9 최솟값 구하기 : root node 구간에 완전히 포함되어 최솟값 리턴

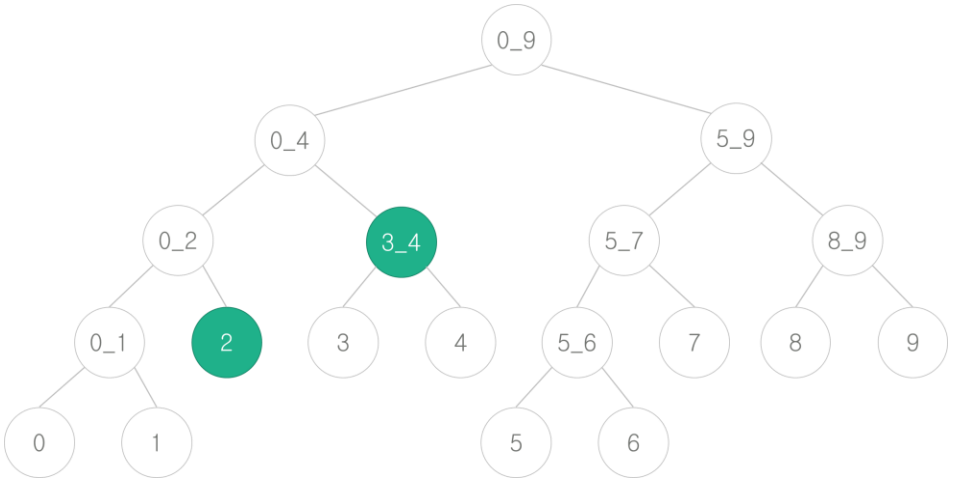


Gaussian

세그먼트 트리

Range Minimum Query

- 구간 [2, 4] 범위의 최솟값 구하기
  ① 노드 [0, 9]가 쿼리 [2, 4]에 부분집합/배타적 경우가 아니므로 왼쪽/오른쪽 자식으로 분할 한다.
  ② 노드 [5, 9]는 쿼리 [2, 4]에 배타적이므로 불가능한 값 리턴
  ③ 노드 [0, 4]는 쿼리 [2, 4]에 부분집합/배타적 경우가 아니므로 왼쪽/오른쪽 자식으로 분할 한다.
  ④ 노드 [3, 4]는 쿼리 [2, 4]에 부분집합 이므로 구간 [3, 4]의 최솟값을 리턴 한다.
  ⑤ 노드 [2]는 쿼리 [2, 4]에 부분집합 이므로 구간 [2]의 최솟값을 리턴 한다.
  ⑥ 노드 [0, 1]은 쿼리 [2, 4]에 배타적이므로 불가능한 값 리턴



Gaussian

세그먼트 트리

Range Minimum Query

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

세그먼트 트리 초기화

```
int query(int node, int start, int end, int i, int j) {
        if (j < start || end < i) return -1;
        if (i <= start && end <= j) return tree[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);
}
```

Minimum Query 함수

Gaussian

세그먼트 트리

Range Minimum Query

```
void update(int node, int start, int end, int i, int new_val) {
        if (i < start || end < i) return;
        tree[node] = min(tree[node], new_val);
        if (start != end) {
                update(2 * node, start, (start + end) / 2, i, new_val);
                update(2 * node + 1, (start + end) / 2 + 1, end, i, new_val);
        }
}
```

세그먼트 트리 업데이트

최소값

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

Gaussian