Persistent Segment Tree

Sogang ICPC Team
2020 Spring 20141574 임지환

- 임지환 (서강대학교 컴퓨터공학과)
- BOJ : raararaara
- 2020 President of ‘Sogang ICPC Team’
- 2018 Sogang Programming Contest Master 부문 6등상
- 2019 Sogang Programming Contest Champion 부문 4등상

Contents

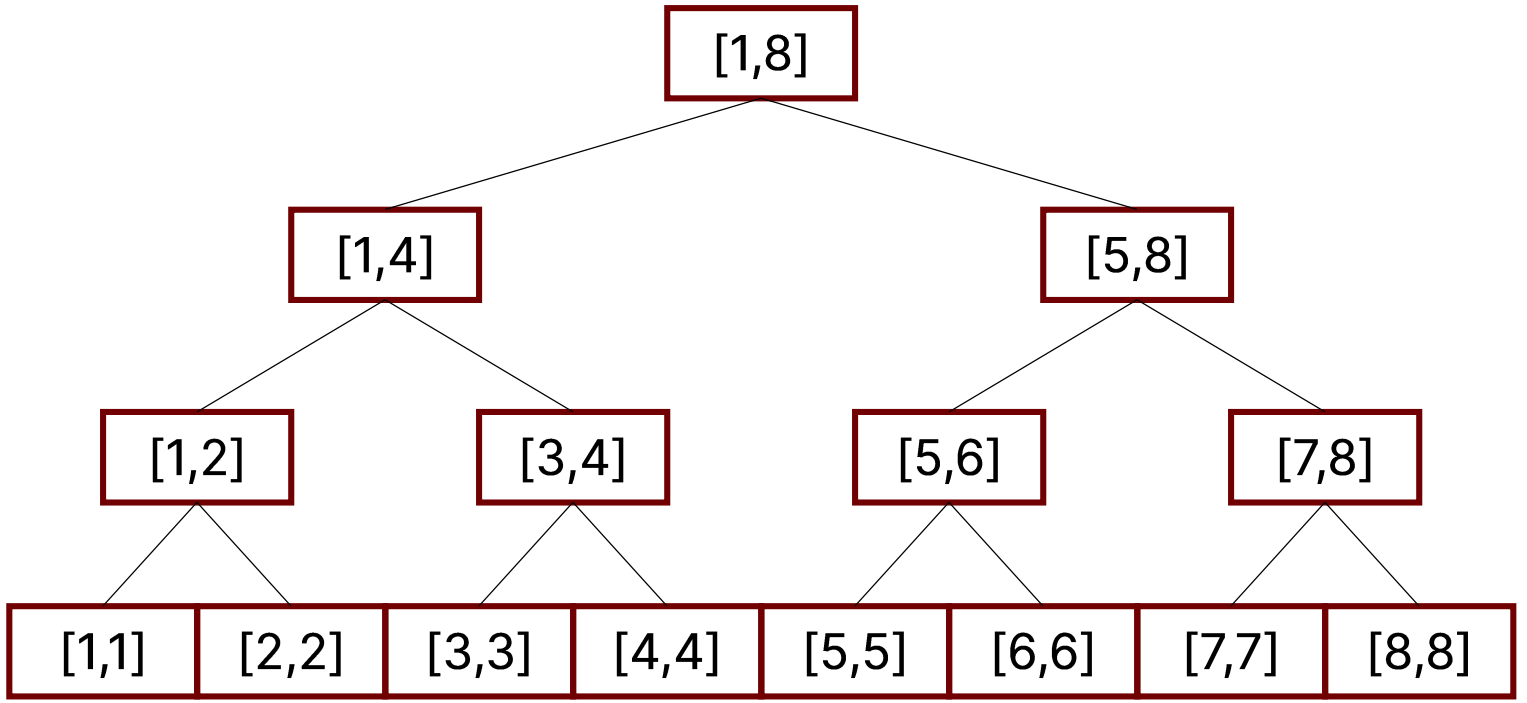
Persistent Data Structure

https://en.wikipedia.org/wiki/Persistent_data_structure

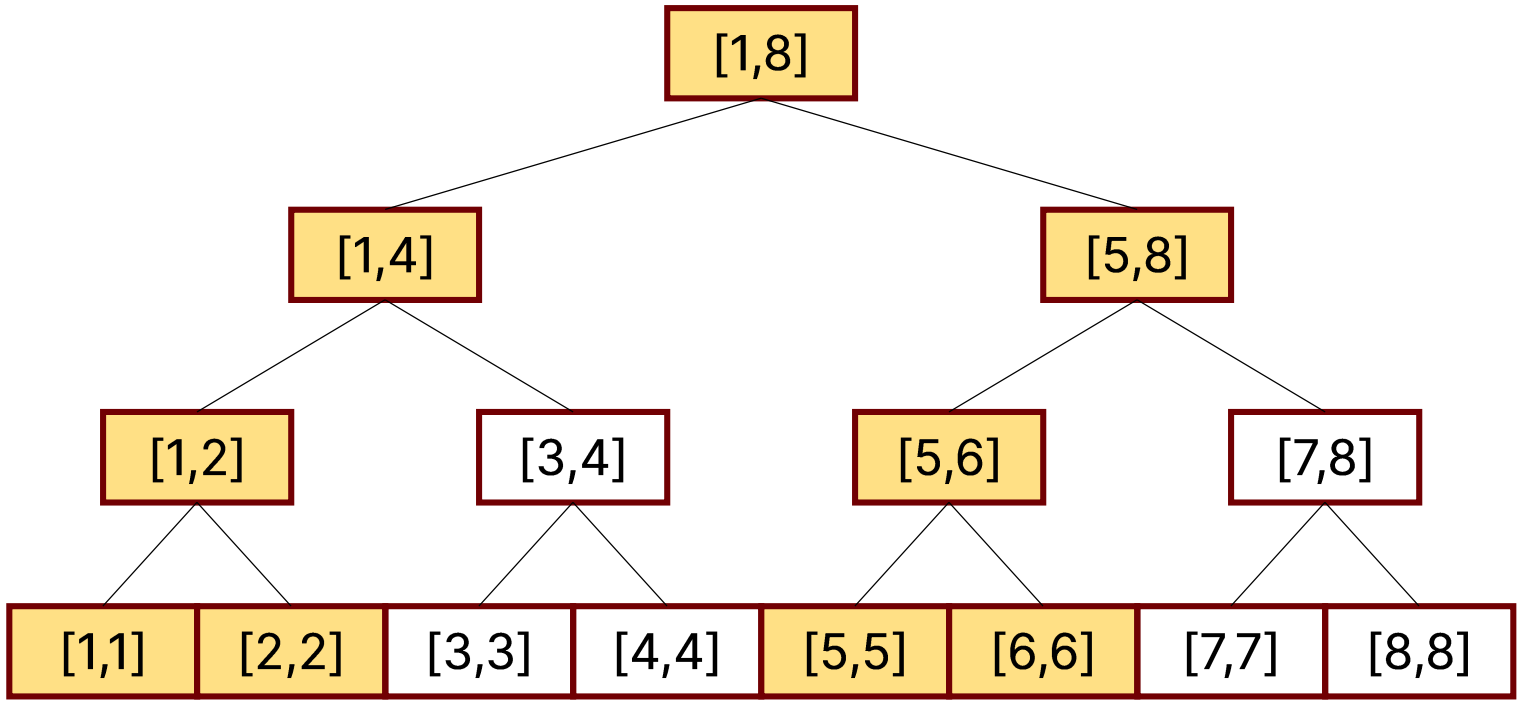
- 이전 버전의 정보를 함께 저장하고 있는 자료구조

- 최신 버전에서 이전 버전의 데이터가 필요한 경우 사용

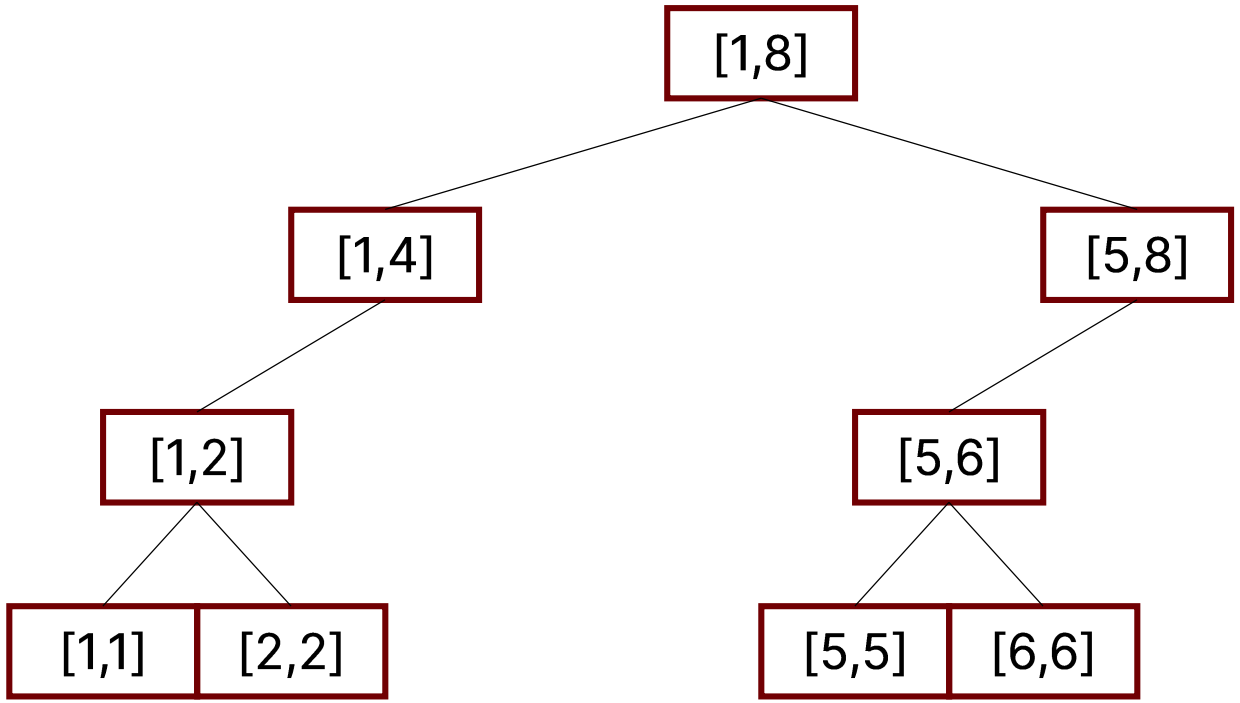
- Ordinary (static) Segment Tree

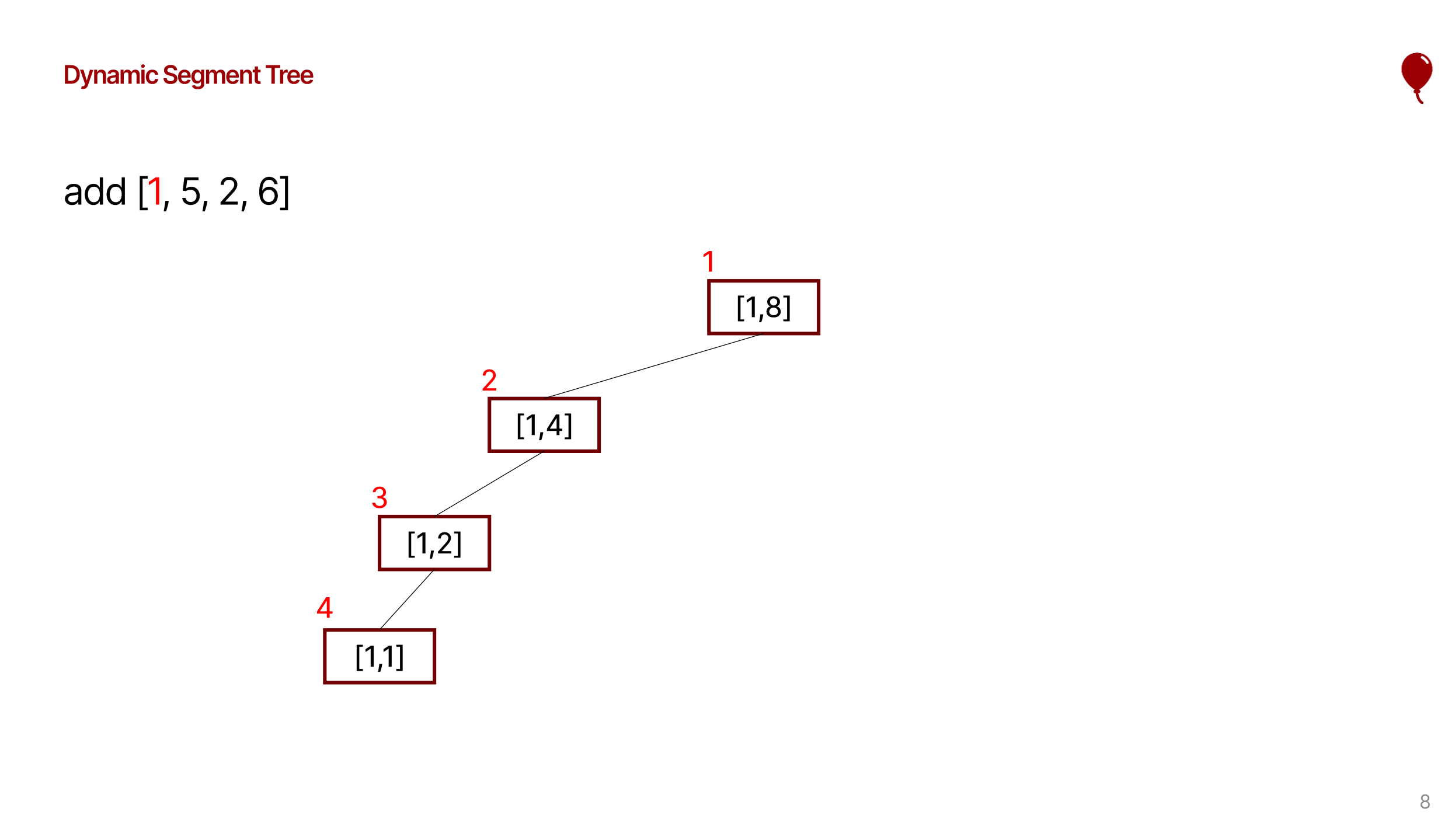


- Ordinary (static) Segment Tree



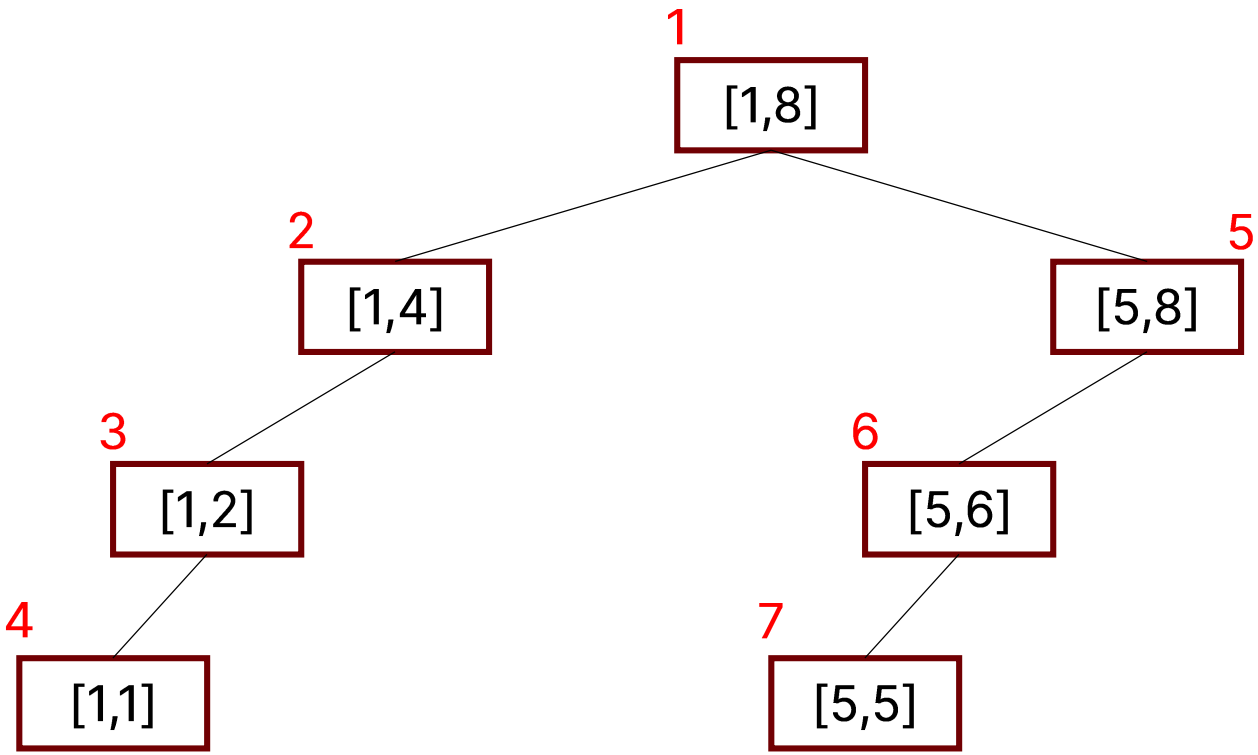
add [1, 5, 2, 6]



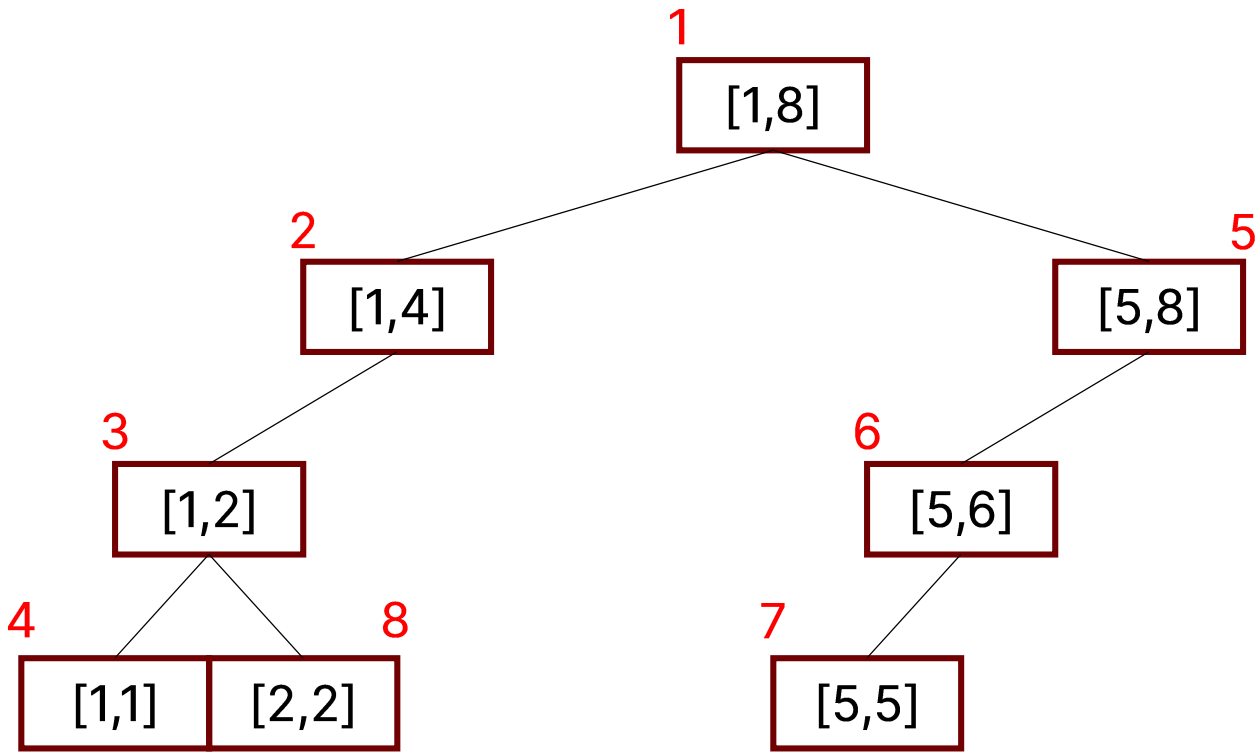


add [1, 5, 2, 6]

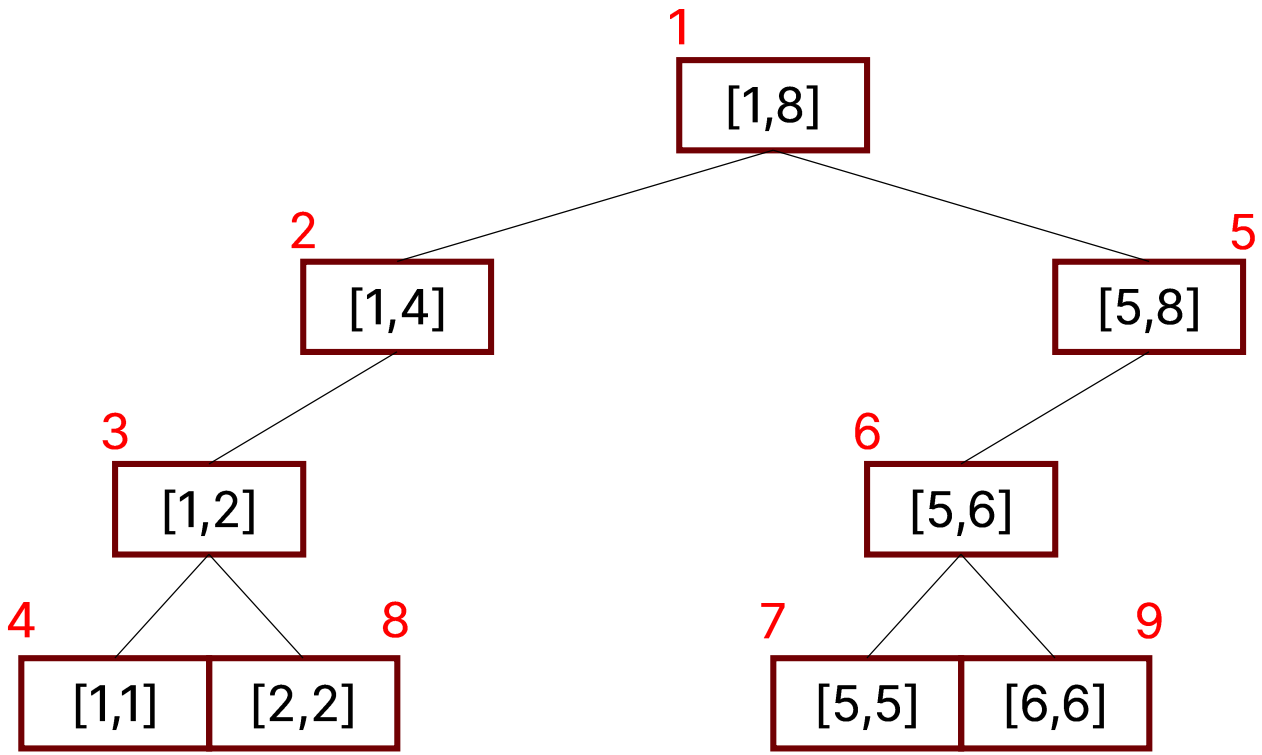
add [1, 5, 2, 6]



add [1, 5, 2, 6]

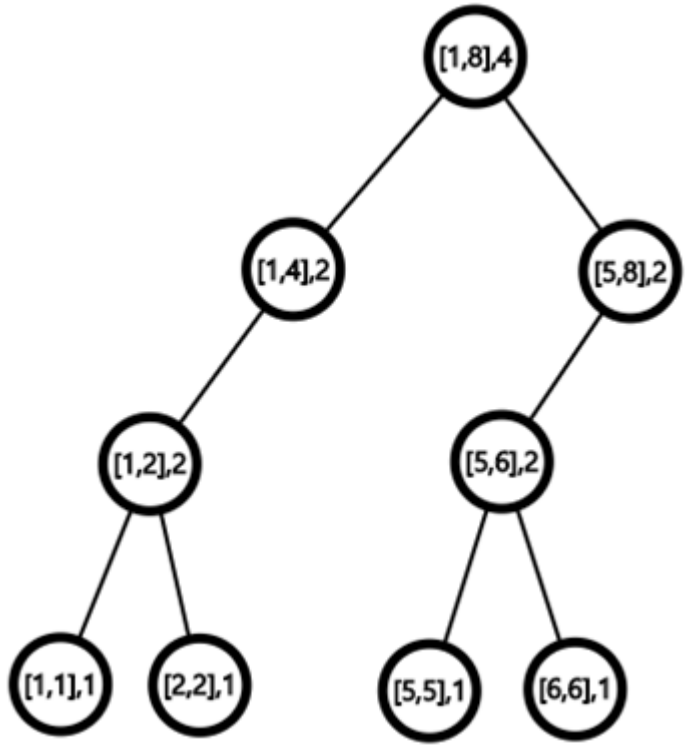


add [1, 5, 2, 6]



- Implementation

```
8   struct Node {
9       int l, r, val;
10  };
11  vector<Node> tree(2);
12
13  void update(int idx, int x, int n = 1, in
14      if (idx < nl || nr < idx) return;
15      tree[n].val += x;
16      if (nl != nr) {
17          lint mid = (nl + nr) / 2;
18          if (idx <= mid) {
19              if (tree[n].l == 0) {
20                  tree.push_back({ 0,0,0 })
21                  tree[n].l = (int)tree.siz
22              }
23              update(idx, x, tree[n].l, nl,
24          }
25          else {
26              if (tree[n].r == 0) {
27                  tree.push_back({ 0,0,0 })
28                  tree[n].r = (int)tree.siz
29              }
30              update(idx, x, tree[n].r, mid + 1, nr);
31          }
32      }
33  }
```




```
35  int sum(int l, int r, int n = 1, int nl = -INF, int nr = INF) {
        if (r < nl || nr < l) return 0;
        (l <= nl && nr <= r) return tree[n].val;
        t mid = (nl + nr) / 2;
        ret = 0;

        (l <= mid) {
         if (tree[n].l == 0) {
             tree.push_back({ 0,0,0 });
             tree[n].l = (int)tree.size() - 1;
         }
         ret += sum(l, r, tree[n].l, nl, mid);

        (mid + 1 <= r) {
         if (tree[n].r == 0) {
             tree.push_back({ 0,0,0 });
             tree[n].r = (int)tree.size() - 1;
         }
         ret += sum(l, r, tree[n].r, mid + 1, nr);

        urn ret;
56  }
```

- Implementation

```cpp
struct Node {
    int l, r, val;
};
vector<Node> tree(2);

void update(int idx, int x, int n = 1, int nl = -INF, int nr = INF) {
    if (idx < nl || nr < idx) return;
    tree[n].val += x;
    if (nl != nr) {
        lint mid = (nl + nr) / 2;
        if (idx <= mid) {
            if (tree[n].l == 0) {
                tree.push_back({ 0,0,0 });
                tree[n].l = (int)tree.size() - 1;
            }
            update(idx, x, tree[n].l, nl, mid);
        }
        else {
            if (tree[n].r == 0) {
                tree.push_back({ 0,0,0 });
                tree[n].r = (int)tree.size() - 1;
            }
            update(idx, x, tree[n].r, mid + 1, nr);
        }
    }
}
```

```cpp
int sum(int l, int r, int n = 1, int nl = -INF, int nr = INF) {
    if (r < nl || nr < l) return 0;
    if (l <= nl && nr <= r) return tree[n].val;
    lint mid = (nl + nr) / 2;
    int ret = 0;

    if (l <= mid) {
        if (tree[n].l == 0) {
            tree.push_back({ 0,0,0 });
            tree[n].l = (int)tree.size() - 1;
        }
        ret += sum(l, r, tree[n].l, nl, mid);
    }
    if (mid + 1 <= r) {
        if (tree[n].r == 0) {
            tree.push_back({ 0,0,0 });
            tree[n].r = (int)tree.size() - 1;
        }
        ret += sum(l, r, tree[n].r, mid + 1, nr);
    }
    return ret;
}
```

Dynamic Segment Tree

- Time Complexity
  - ✓ Update : $O(logN)$
  - ✓ Range query : $O(logN)$

- Space Complexity

  let L : range of data, N : number of queries
  - ✓ Update(add) : $O(logL)$, total : $O(NlogL)$

#15816 퀘스트 중인 모험가

첫째 줄에 지금까지 달성한 퀘스트의 개수 N이 주어진다. ($1 \leq N \leq 1,000,000$)

둘째 줄에 지금까지 달성한 퀘스트들의 번호 Q_1 ... Q_N 까지의 N개의 수가 주어진다. ($-1,000,000,000 \leq Q[i] \leq 1,000,000,000$, $Q[i] < Q[i+1]$)

셋째 줄에 애드온 요청의 개수 M이 주어진다. ($1 \leq M \leq 1,000,000$)

넷째 줄부터 M개의 줄에 걸쳐서 애드온에 요청할 명령이 주어진다.

1. 1 X: 퀘스트 번호 X를 달성했다. 애드온에 이를 반영해야 한다. ($-1,000,000,000 \leq X \leq 1,000,000,000$)
2. 2 L R: 퀘스트 번호 L 이상 R 이하인 퀘스트 중, 모험가가 달성하지 못한 퀘스트의 개수를 출력한다. ($-1,000,000,000 \leq L \leq R \leq 1,000,000,000$)

출력

애드온 기능2에 해당하는 출력을 요청당 한 줄씩 출력한다.

예제 입력 1 복사

```
3
1 10 20
4
2 1 20
1 5
2 1 20
2 1 1
```

예제 출력 1 복사

```
17
16
0
```

#15816 퀘스트 중인 모험가

- By dynamic segment tree:
  - 12byte per Node
  - 10^6update queries
  - 2×10^9 range of data

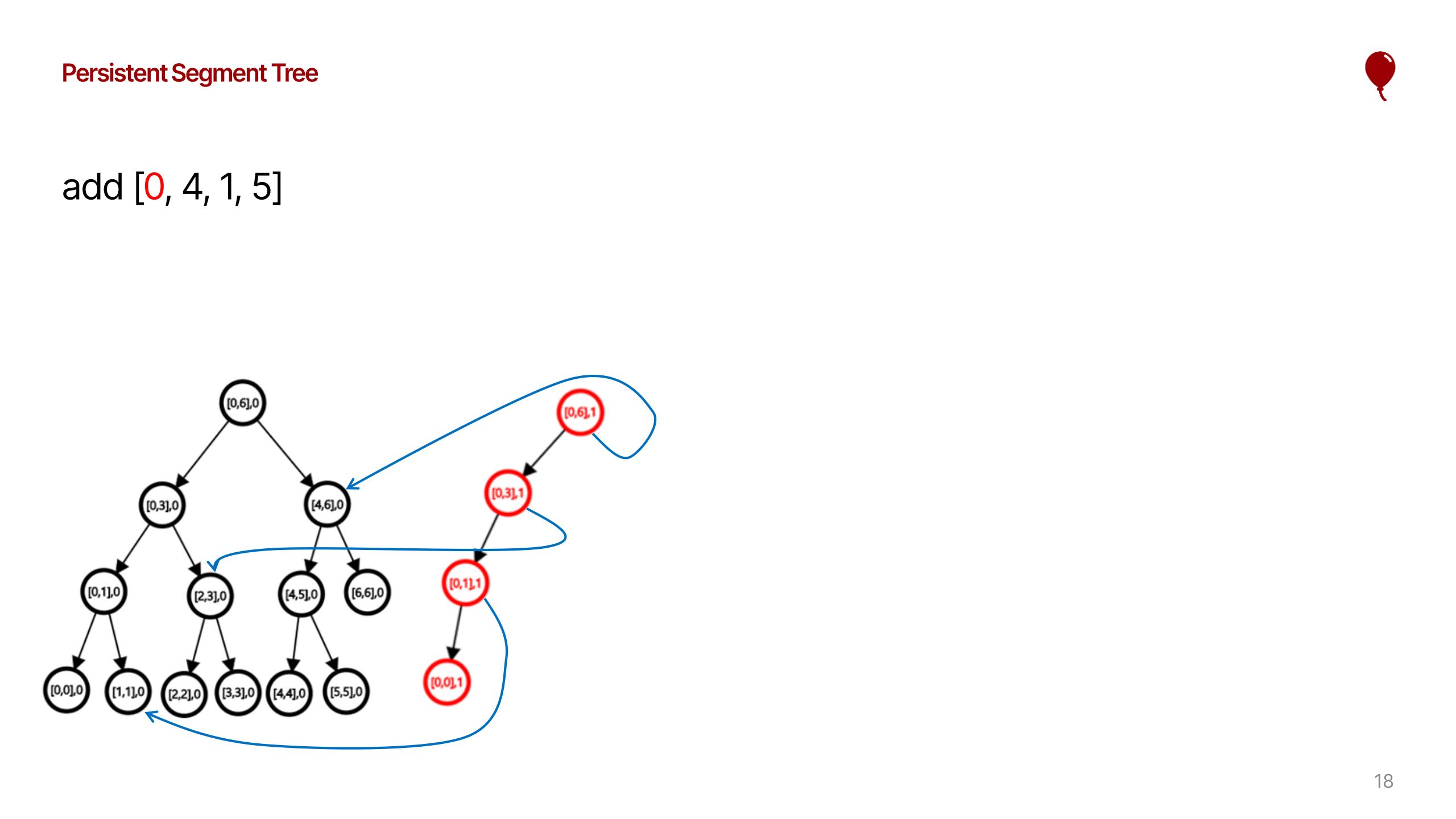
채점 번호	아이디	문제 번호	결과	메모리	시간	언어	코드 길이
19031967	raararaara	#4 15816	맞았습니다!!	66892 KB	1316 ms	C++14 / 수정	1538 B
19031429	raararaara	#4 15816	메모리 초과			C++14 / 수정	1593 B
19031396	raararaara	#4 15816	메모리 초과			C++14 / 수정	1593 B
19031346	raararaara	#4 15816	메모리 초과			C++14 / 수정	1592 B
19031314	raararaara	#4 15816	메모리 초과			C++14 / 수정	1585 B
19031179	raararaara	#4 15816	메모리 초과			C++14 / 수정	1573 B

Persistent Segment Tree

- 세그먼트 트리의 갱신 과정, or 생성 과정을 모두 저장하고 있는 자료구조

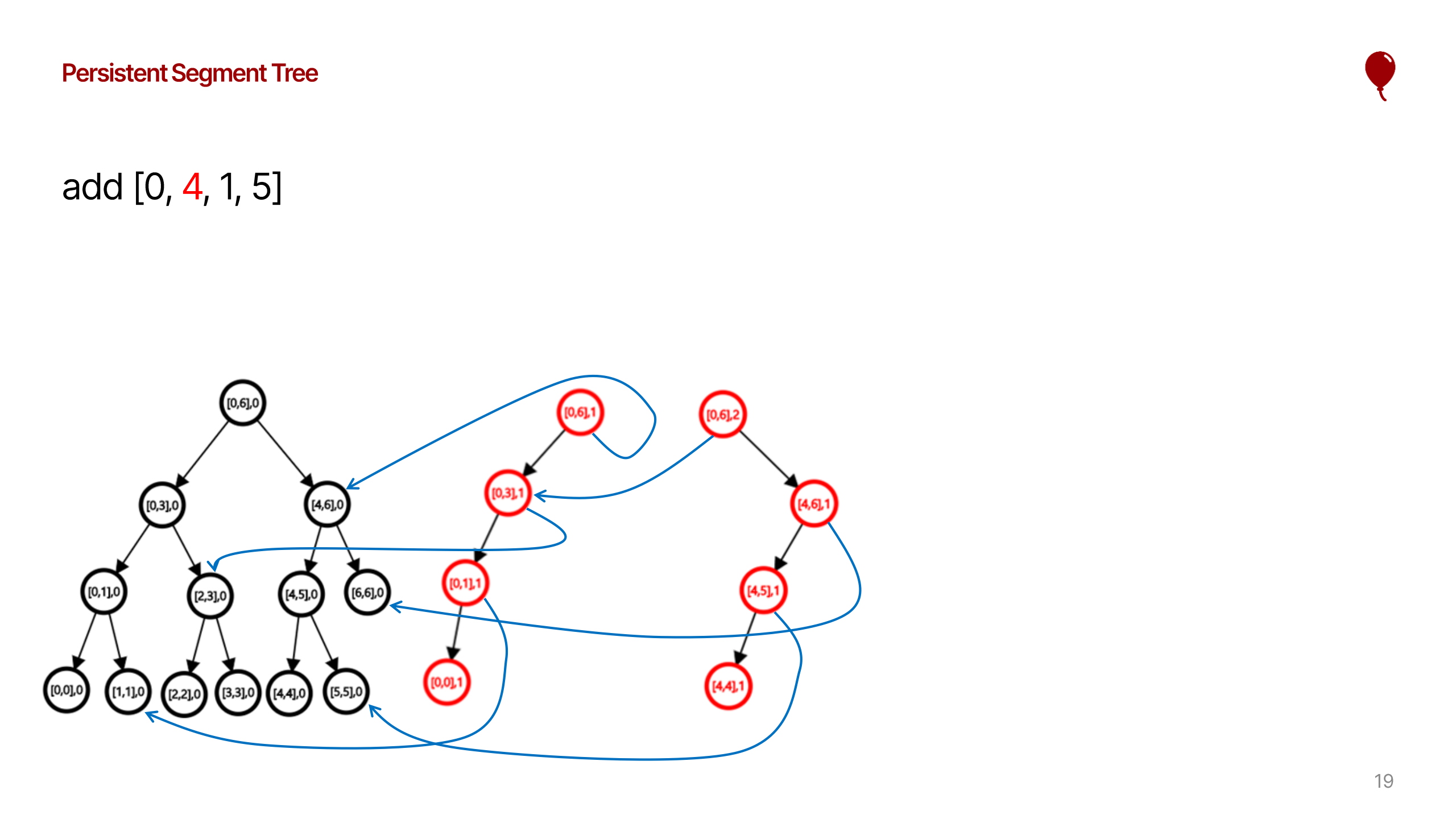
- Space Complexity : $O(NlogN)$ / $O(logN)$ per one ver.

- Usage:
  - ✓ 특정 버전의 정보가 필요한 경우
  - ✓ update query가 없는 2D range query



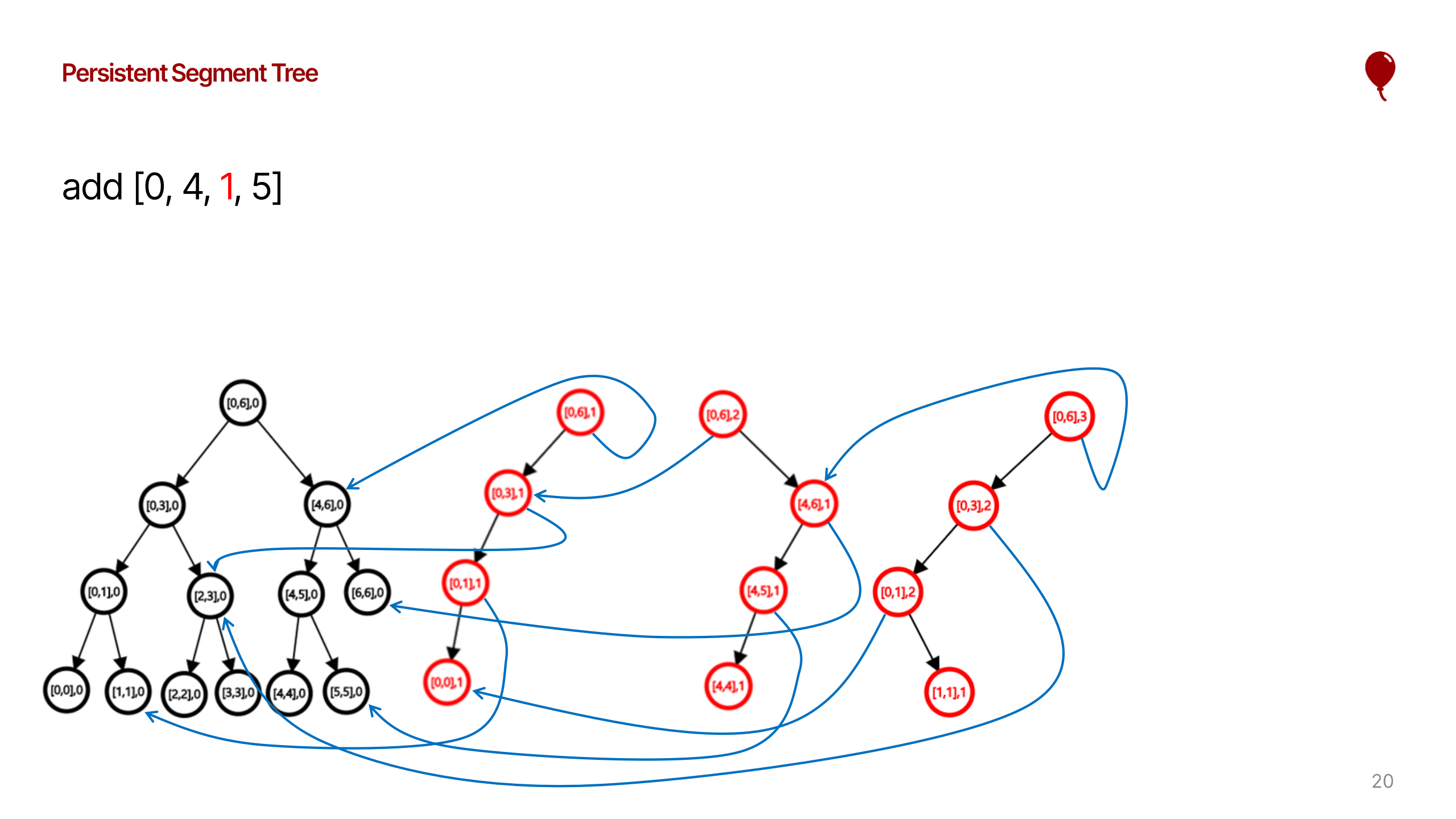
add [0, 4, 1, 5]

[0,6],0
[0,3],0
[4,6],0
[0,1],0
[2,3],0
[4,5],0
[6,6],0
[0,0],0
[1,1],0
[2,2],0
[3,3],0
[4,4],0
[5,5],0
[0,6],1
[0,3],1
[0,1],1
[0,0],1



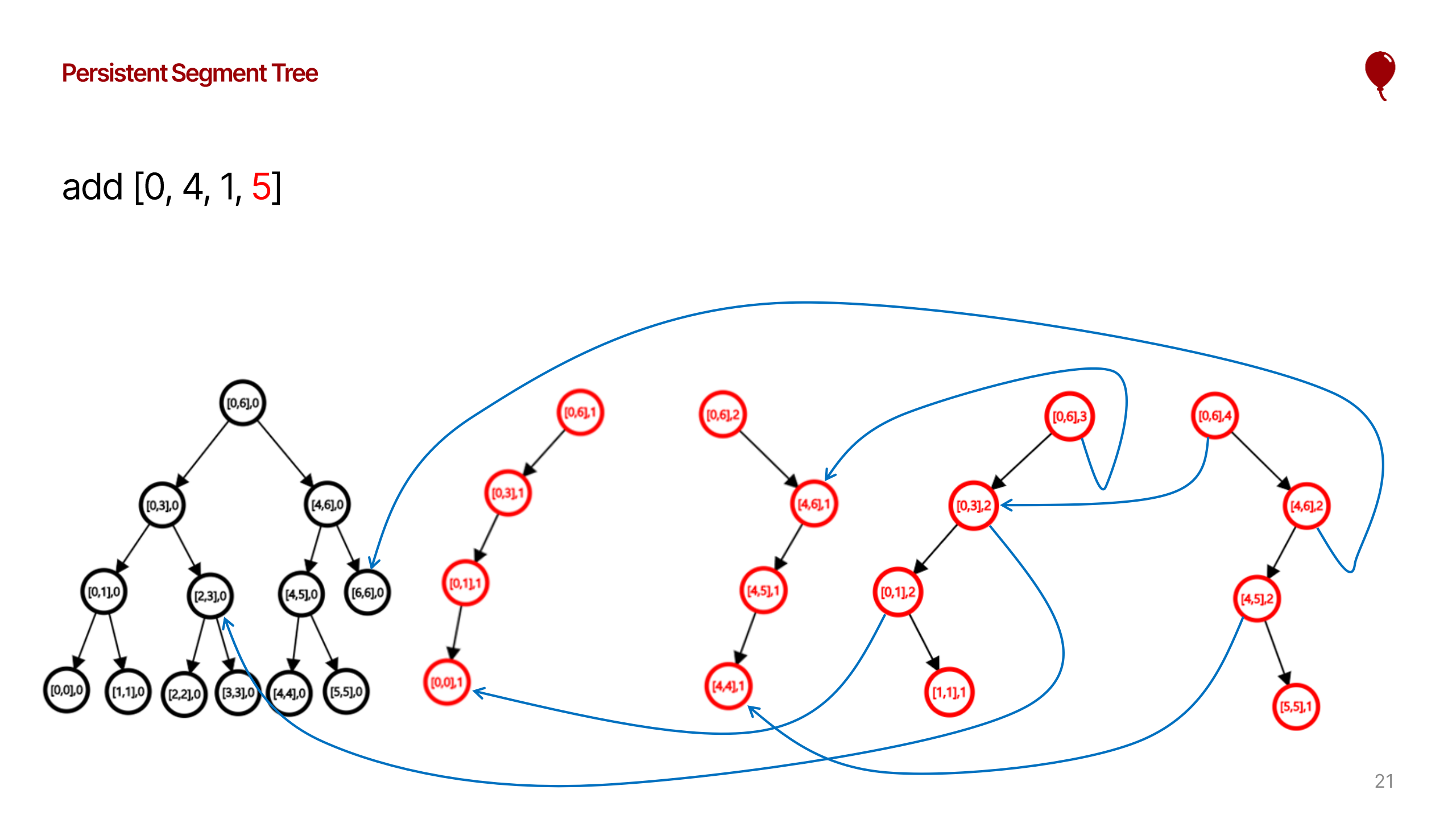
add [0, 4, 1, 5]

[0,6],0
[0,3],0
[4,6],0
[0,1],0
[2,3],0
[4,5],0
[6,6],0
[0,0],0
[1,1],0
[2,2],0
[3,3],0
[4,4],0
[5,5],0
[0,6],1
[0,3],1
[0,1],1
[0,0],1
[0,6],2
[4,6],1
[4,5],1
[4,4],1



add [0, 4, 1, 5]

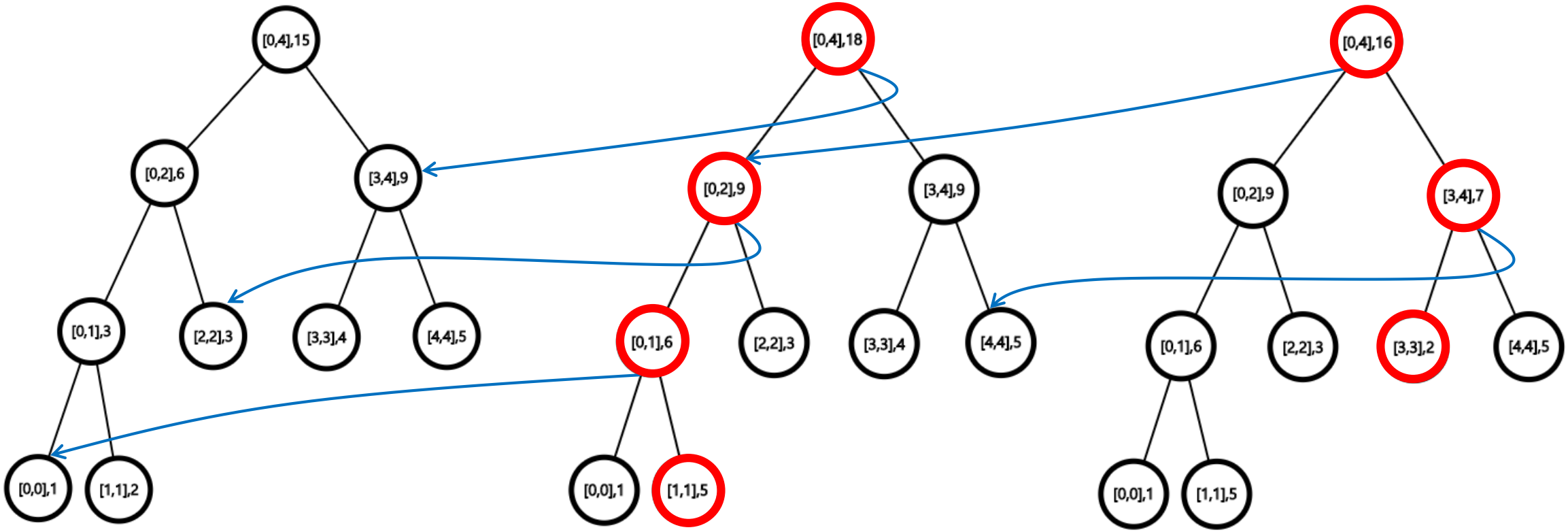
[0,6],0
[0,3],0
[4,6],0
[0,1],0
[2,3],0
[4,5],0
[6,6],0
[0,0],0
[1,1],0
[2,2],0
[3,3],0
[4,4],0
[5,5],0
[0,6],1
[0,3],1
[0,1],1
[0,0],1
[0,6],2
[4,6],1
[4,5],1
[4,4],1
[0,6],3
[0,3],2
[0,1],2
[1,1],1



add [0, 4, 1, 5]

[0,6],0
[0,3],0
[4,6],0
[0,1],0
[2,3],0
[4,5],0
[6,6],0
[0,0],0
[1,1],0
[2,2],0
[3,3],0
[4,4],0
[5,5],0
[0,6],1
[0,3],1
[0,1],1
[0,0],1
[0,6],2
[4,6],1
[4,5],1
[4,4],1
[0,6],3
[0,3],2
[0,1],2
[1,1],1
[0,6],4
[4,6],2
[4,5],2
[5,5],1

- 쿼리($1 \leq M \leq 100{,}000$)
  - ✓단일 원소 update
  - ✓range sum
- k'th version required



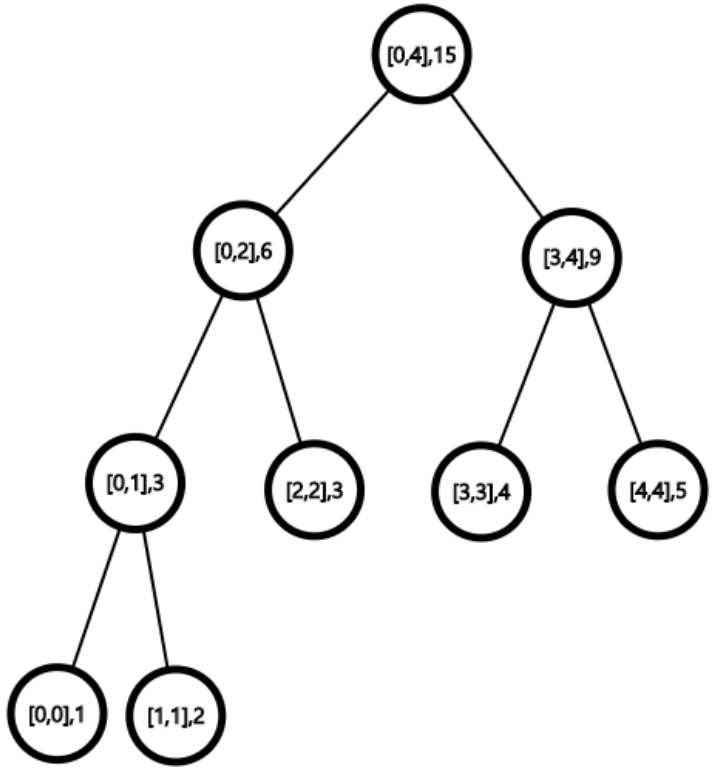
Persistent Segment Tree - Implementation

- Tree definition

```
11    lint a[mxn];
12    vector<int> xlist;
13
14    int rt[mxn];
15    struct Node {
16        int l, r;
17        lint val;
18    };
19    vector<Node> tree;
```

- 1st Version

```
21    void init(int n, int nl = 0, int nr = N-1) {
22        if (nl == nr) {
23            tree[n].val = a[nl];
24            return;
25        }
26        int mid = (nl + nr) / 2;
27        tree.push_back({ 0,0,0 }), tree[n].l = (int)tree.size() - 1;
28        init(tree[n].l, nl, mid);
29        tree.push_back({ 0,0,0 }), tree[n].r = (int)tree.size() - 1;
30        init(tree[n].r, mid + 1, nr);
31        tree[n].val = tree[tree[n].l].val + tree[tree[n].r].val;
32    }
```



Persistent Segment Tree - Implementation

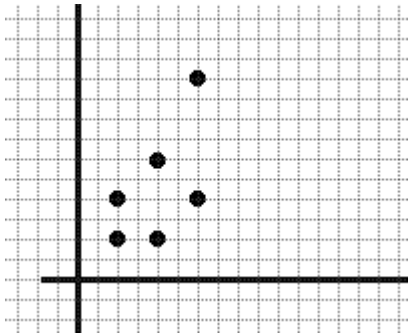
- point update

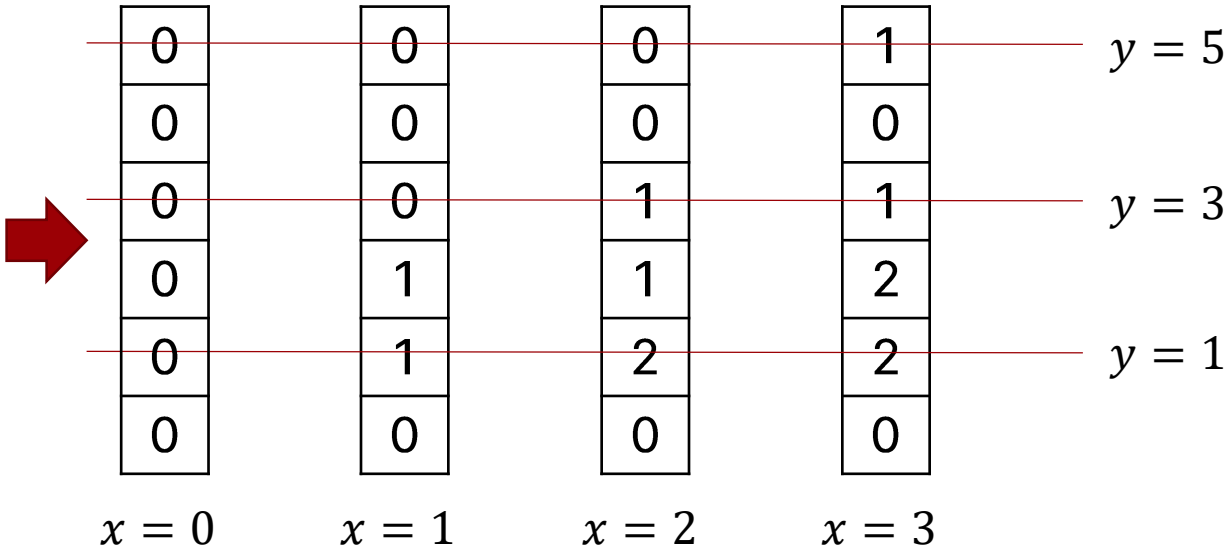
```
34  void update(int idx, int x, int n = 1, int nl = 0, int nr = N-1) {
35      if (nl == nr) return;
36      int mid = (nl + nr) / 2;
37      if (idx <= mid) {
38          int lidx = tree[n].l;
39          tree.push_back({ tree[lidx].l, tree[lidx].r, tree[lidx].val + x });
40          tree[n].l = (int)tree.size() - 1;
41          update(idx, x, tree[n].l, nl, mid);
42      }
43      else {
44          int ridx = tree[n].r;
45          tree.push_back({ tree[ridx].l, tree[ridx].r, tree[ridx].val + x });
46          tree[n].r = (int)tree.size() - 1;
47          update(idx, x, tree[n].r, mid + 1, nr);
48      }
49  }
```

- range sum

```
51  int sum(int l, int r, int n, int nl = 0, int nr = N-1) {
52      if (r < nl || nr < l) return 0;
53      if (l <= nl && nr <= r) return tree[n].val;
54      int mid = (nl + nr) / 2;
55      return sum(l, r, tree[n].l, nl, mid) + sum(l, r, tree[n].r, mid + 1, nr);
56  }
```

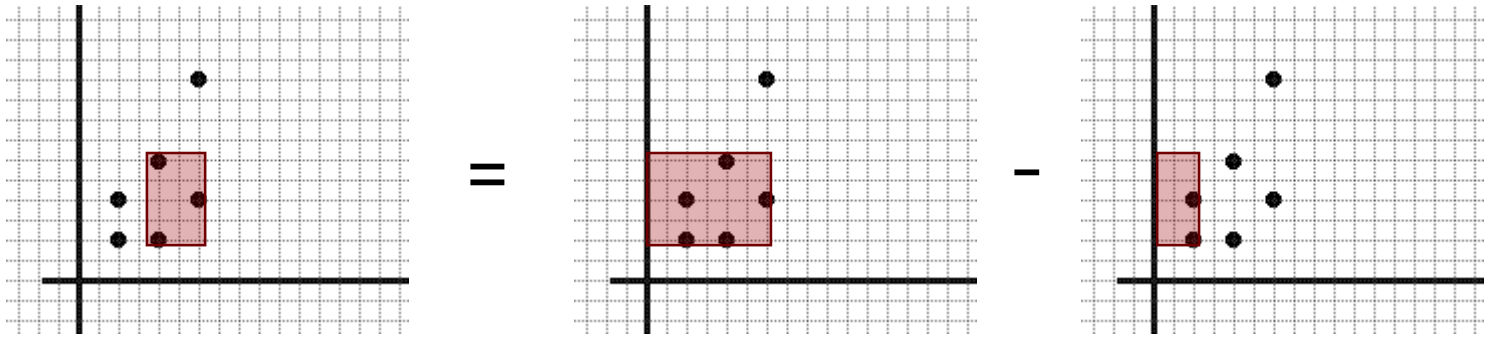
- 2D range query ($1 \leq M \leq 50{,}000$)
- grid range : $0 \leq x, y \leq 10^5$
- kth version of PST : $x = [0{:}k]$에 위치한 모든 점들의 정보





- 2D range query

query(y_l,y_r,x^{th} ver)



Summary

- 2D range query, 또는 이전 정보를 참조해야 하는 쿼리 문제를 해결하기 위해 알아야 하는 자료구조.
- https://solved.ac/problems/algorithms/55