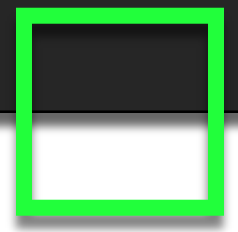
#고급반
#2주차

Lazy Propagation and Plane Sweeping

T. 노주찬



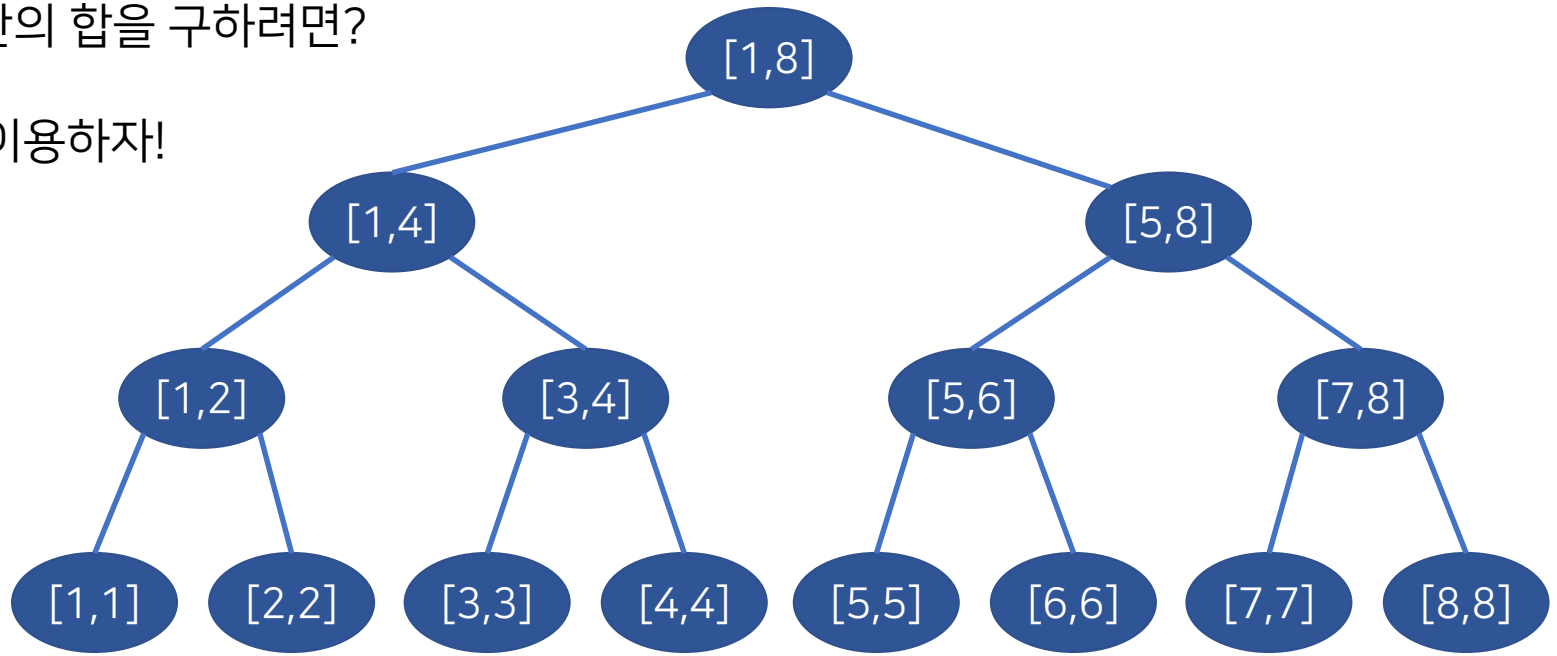
ALOHA
The algorithm club.

저번 주 내용 Review

BOJ 2042 – 구간 합 구하기

수의 변경이 빈번하게 일어나는 구간의 합을 구하려면?

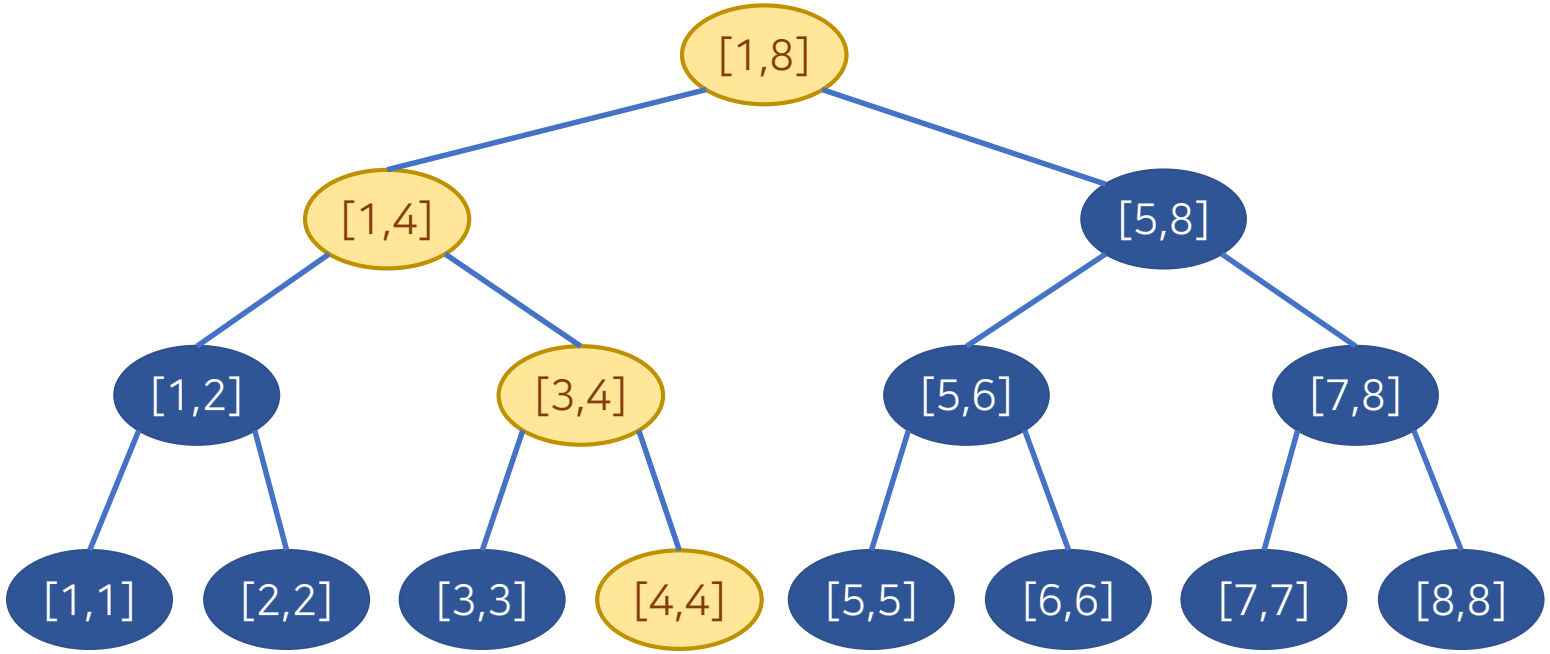
Segment Tree, Fenwick Tree를 이용하자!



저번 주 내용 Review

4번째 수를 변경하려면?

그림과 같이 내려가서 변경



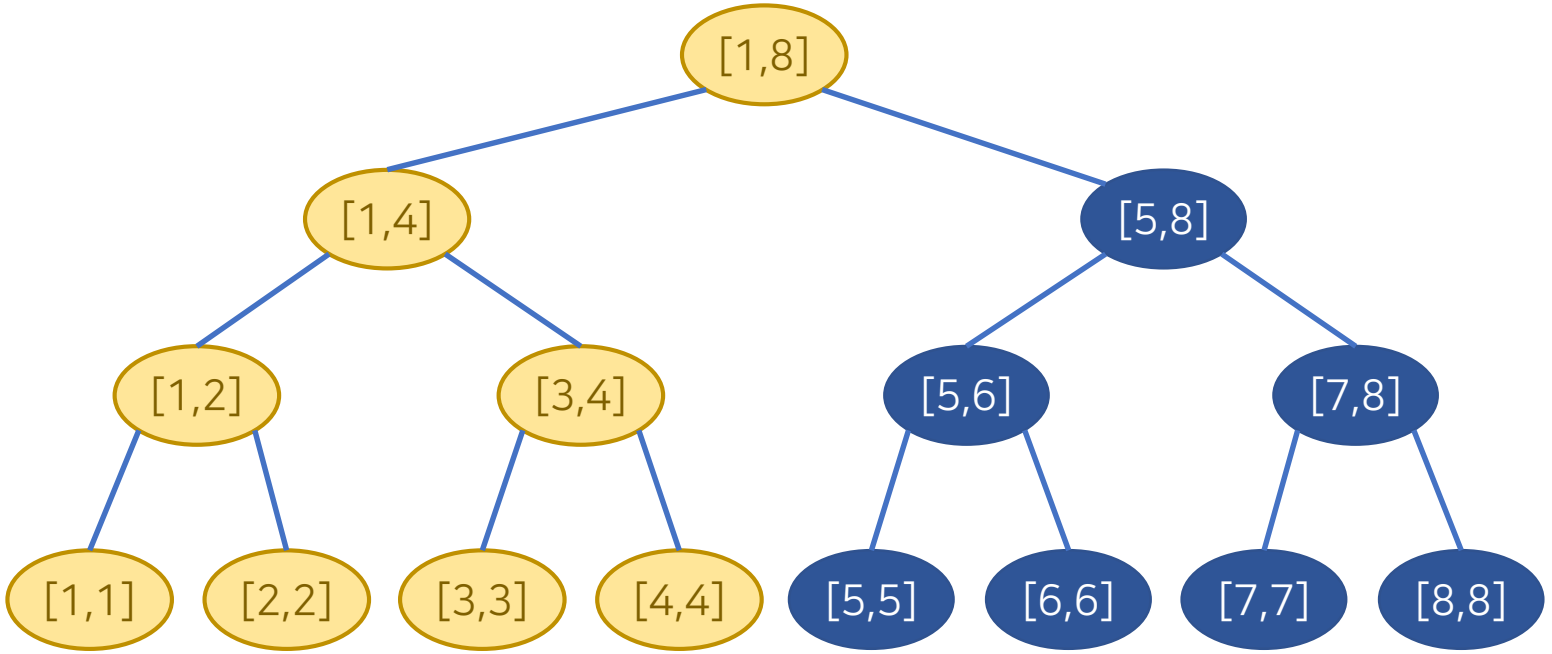
이런 상황을 가정해봅시다.

그런데 만약 x 로 바꿔야 할 수가 1개가 아니라 어느 연속적인 구간이라면?

수를 1개 변경하는데 O(log N)

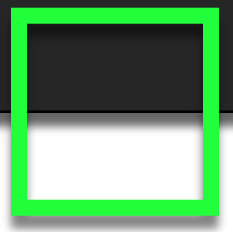
N개를 변경하려면 O(N log N) ?

= TLE!



#1

Lazy Propagation



Lazy... 그게 뭐에요?

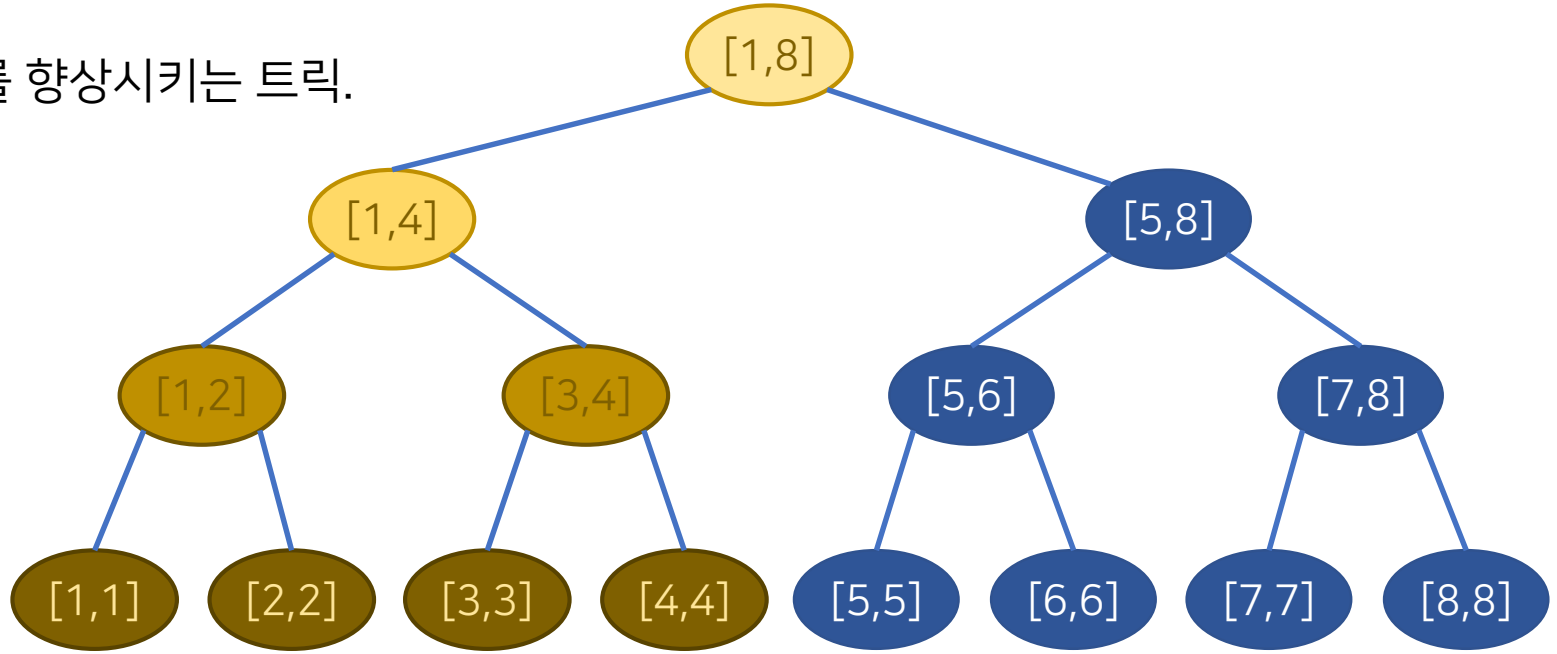
게으른 전파

Lazy Propagation

Lazy... 그게 뭐에요?

전파: 변경 내용을 Root Node 에서 Leaf Node 까지 Child Node 에 값의 변화를 적용시키는 것.

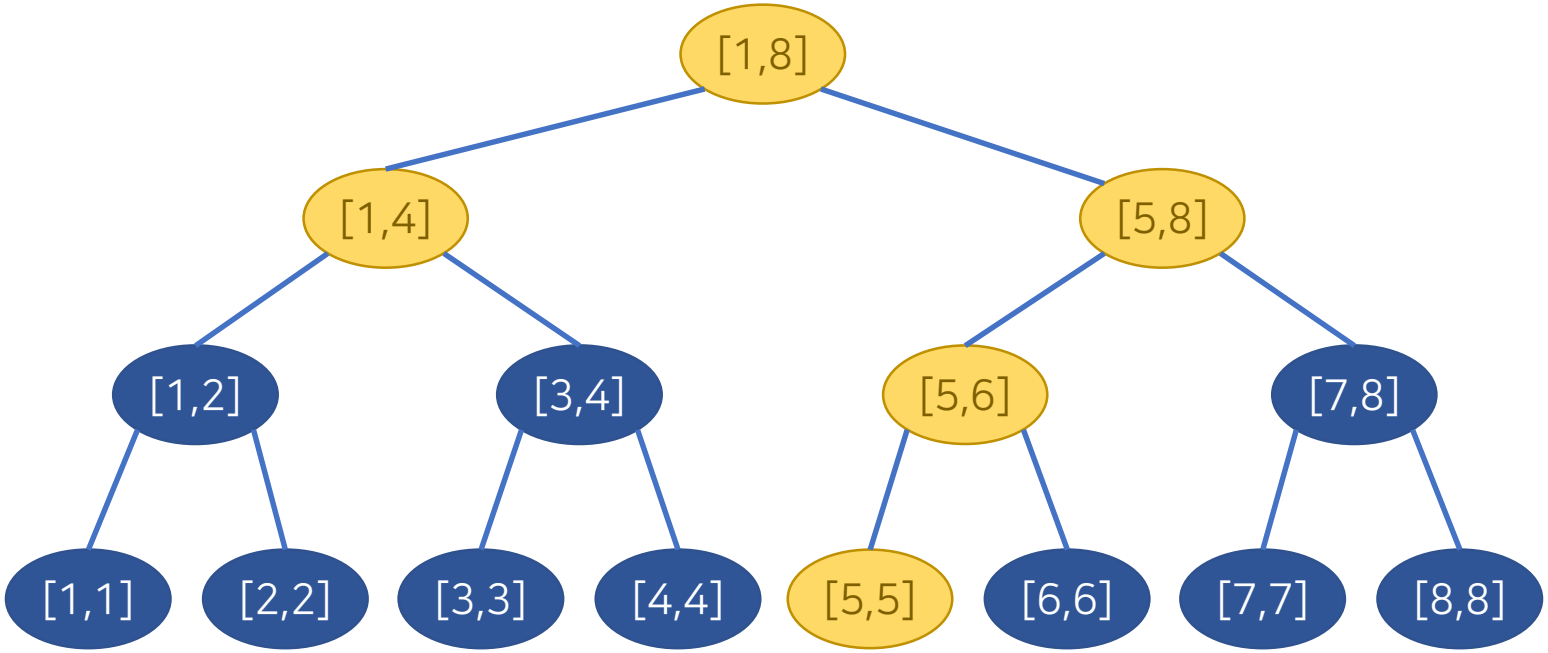
Lazy Propagation:
전파를 최대한 미뤄서 시간 복잡도를 향상시키는 트릭.



[1, 4] 구간에 대해 변경 내용을 전파 시키는 모습

Lazy Propagation

어느 구간의 값을 요청하는 쿼리가 들어왔을 때, 항상 Leaf Node 까지 내려가지 않는다.



Lazy Propagation

그렇다면 항상 Leaf Node 까지 전파할 필요가 있을까?

No! 값의 변화를 가지고 있다가 그 Node 의 값이 필요할 때만 Child Node 로 전파하면 된다!

Lazy Propagation

편의상 값의 변화를 가지고 있는 배열을 Lazy 라고 합시다.

Case 1. Update

1. 값이 변경되는 구간에 포함되는 Node 까지 내려간다.
2. 해당 노드에 Lazy 가 있다면 적용하고, Childe Node 에 Lazy를 전파한 다음 Lazy 를 0으로 바꿔준다.
3. 값의 변화를 Lazy 를 넣어준 뒤, 2번의 과정을 해준다.

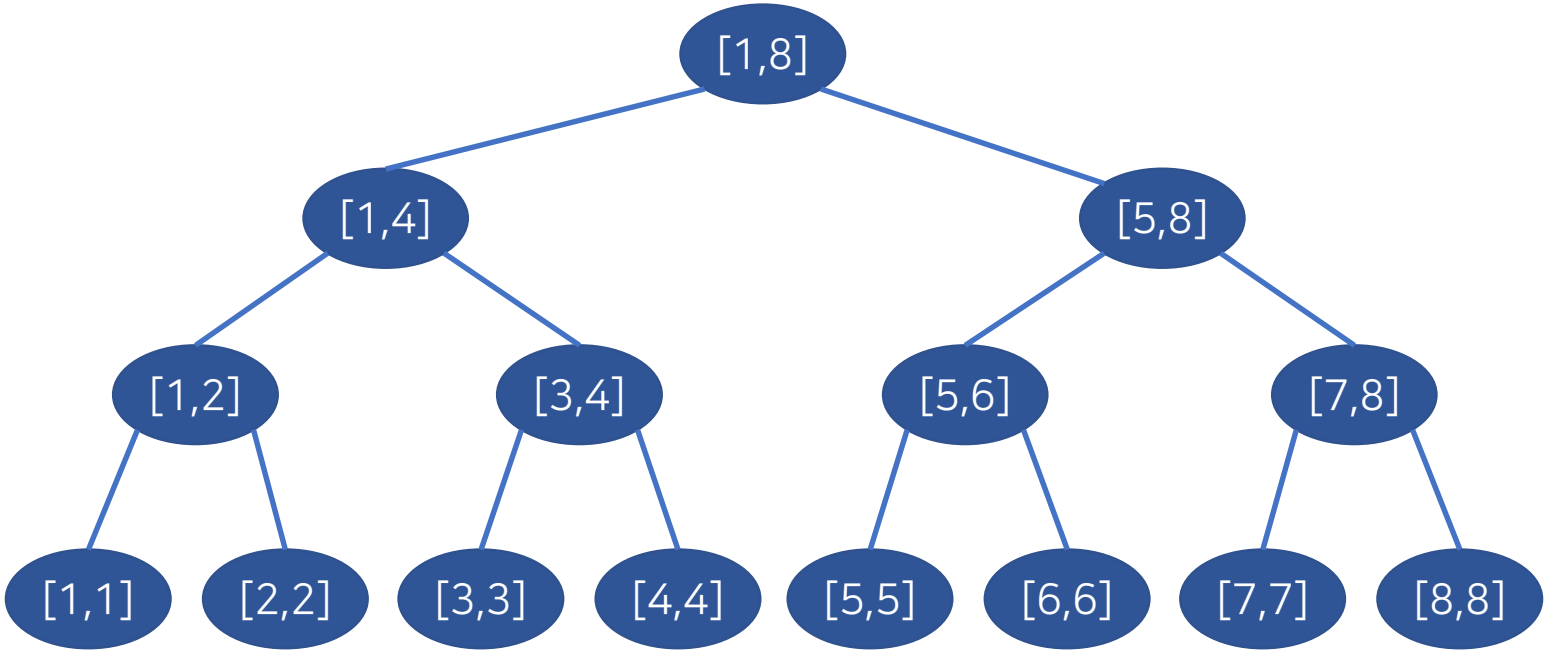
Case 2. Get

1. 값을 구하는 구간에 포함되는 Node 까지 내려간다.
2. 해당 노드에 Lazy가 있다면 적용하고, Childe Node에 Lazy를 전파한 다음 Lazy 를 0으로 바꿔준다.
3. 해당 노드가 가지고 있는 값을 반환한다.

Tip. 각 Case의 2번 과정은 동일하니 함수로 만들어서 분리할 수 있다!

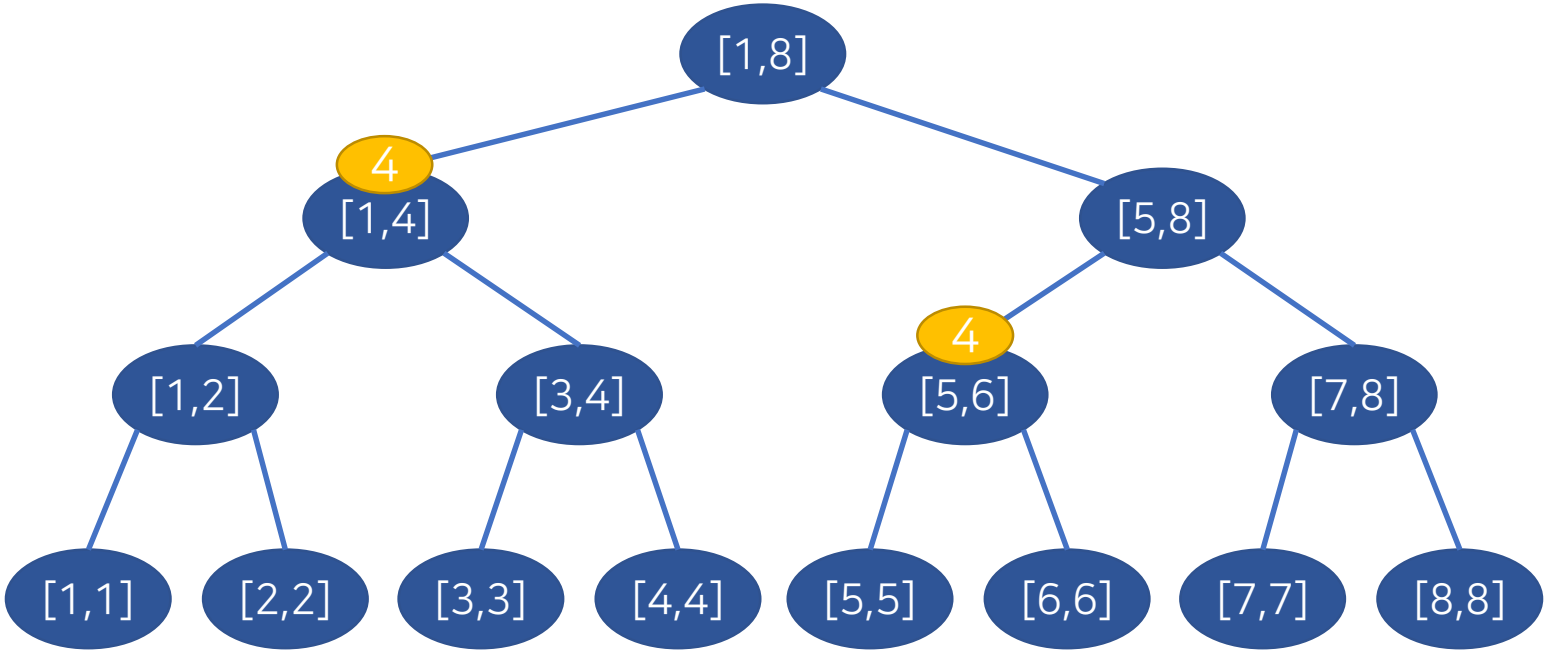
Example – Update

[1, 6] 의 구간에 +4 를 하는 쿼리가 들어왔다면….



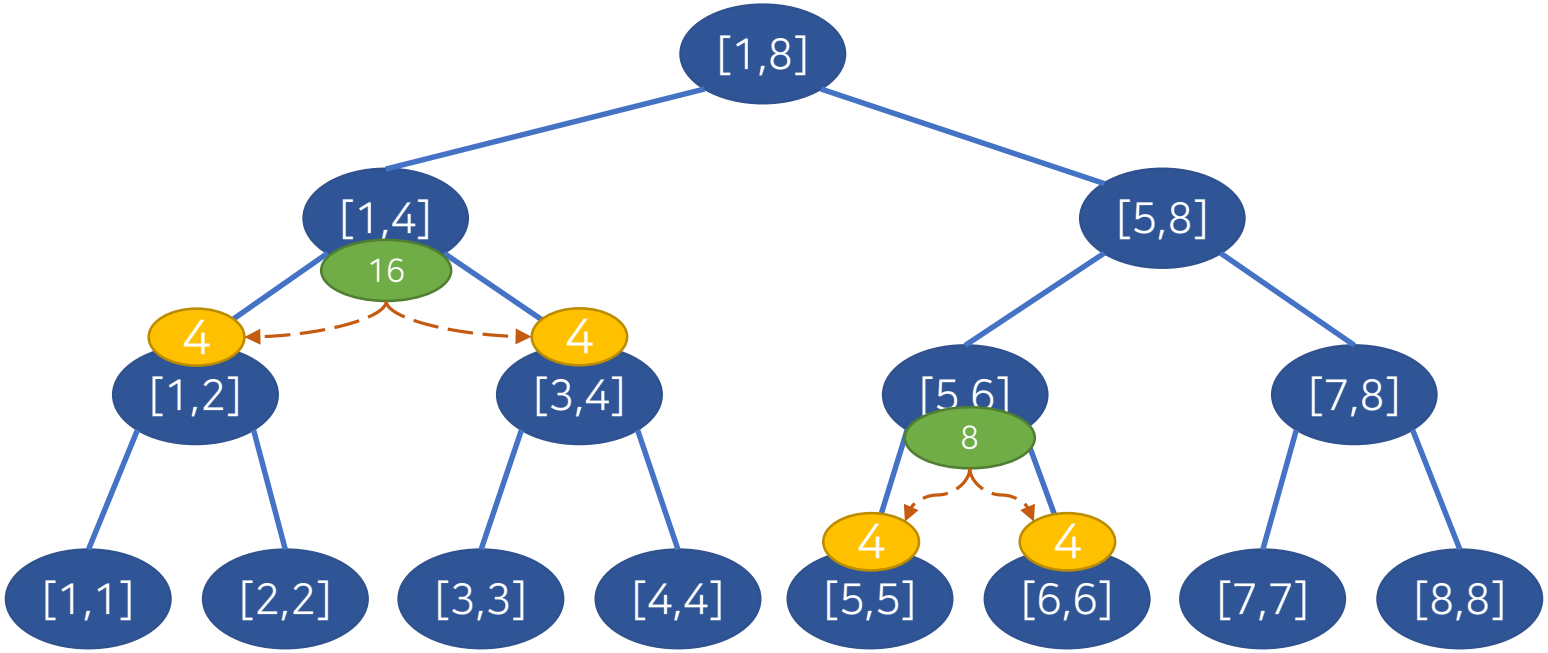
Example – Update

구간에 포함되는 [1,4], [5,6] 노드에 Lazy를 만들고….



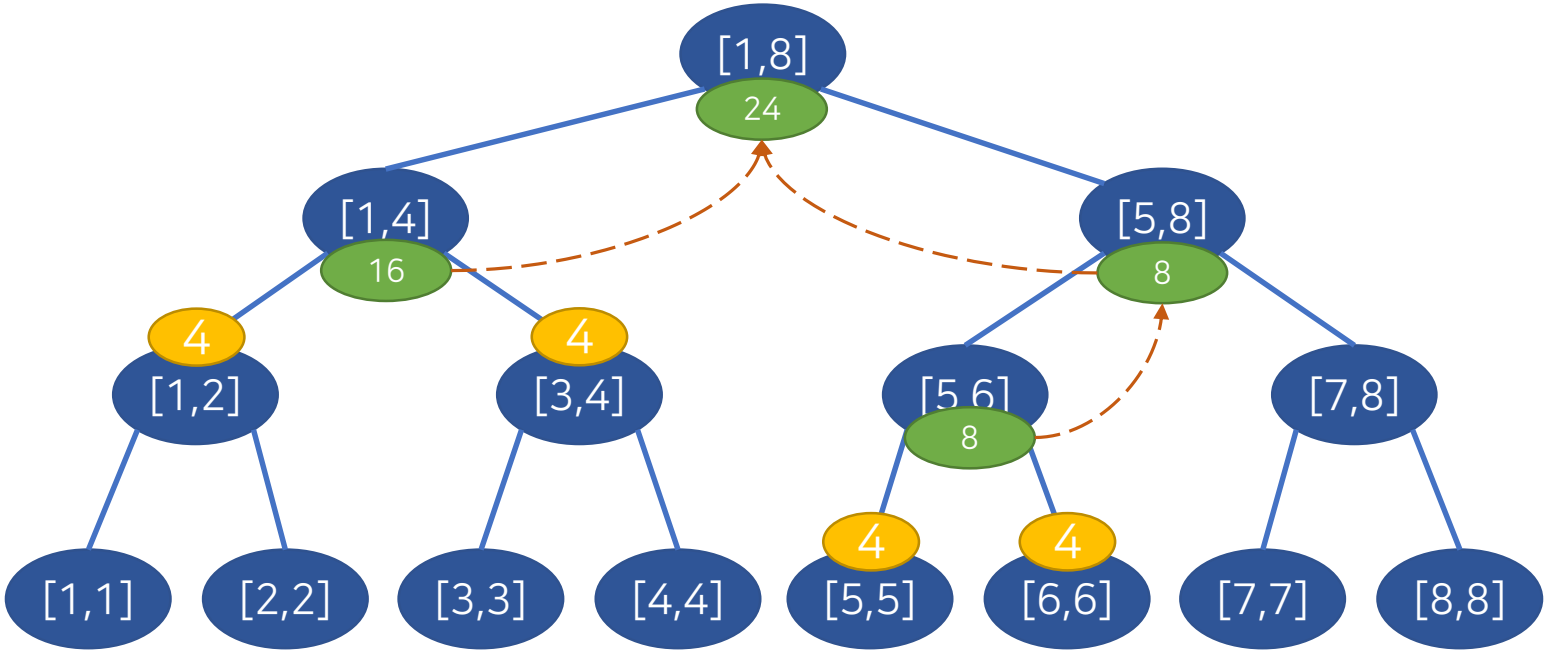
Example – Update

Lazy를 적용 (구간 안의 요소 개수 * Lazy 를 합에 추가) 하고, Child Node에 전파한 뒤….



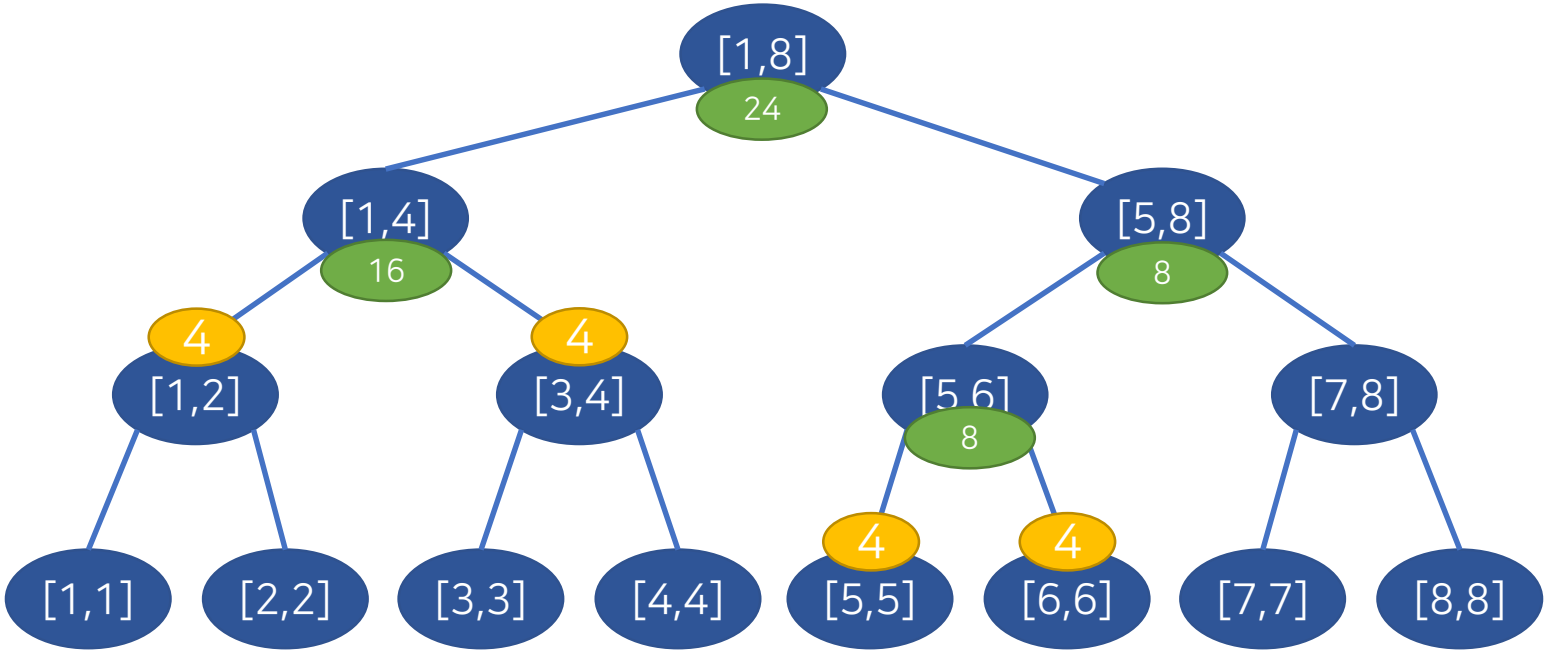
Example – Update

일반 Segment Tree와 마찬가지로 Parent Node 로 올라가면서 값을 구하면 끝!



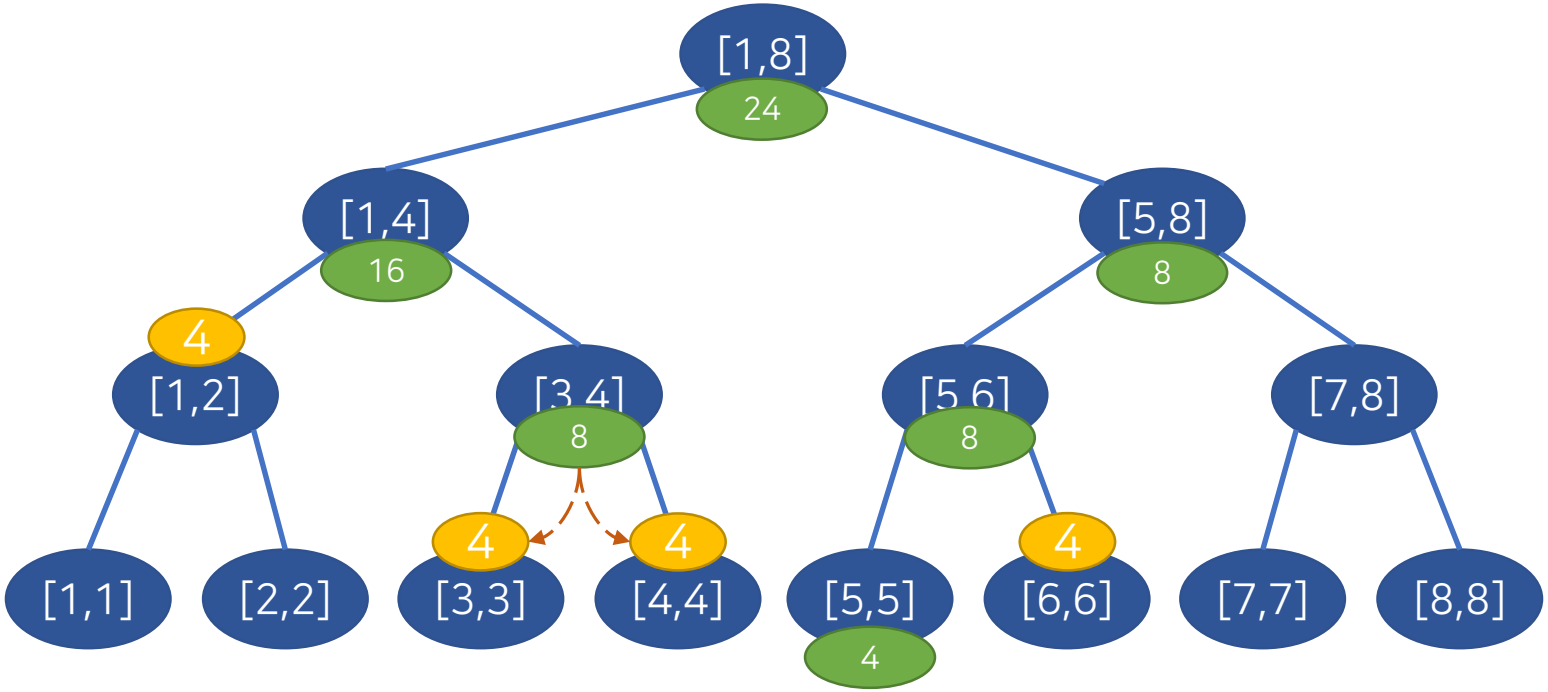
Example – Get

구간 [3,5] 의 합을 구하는 쿼리가 들어왔을 때….



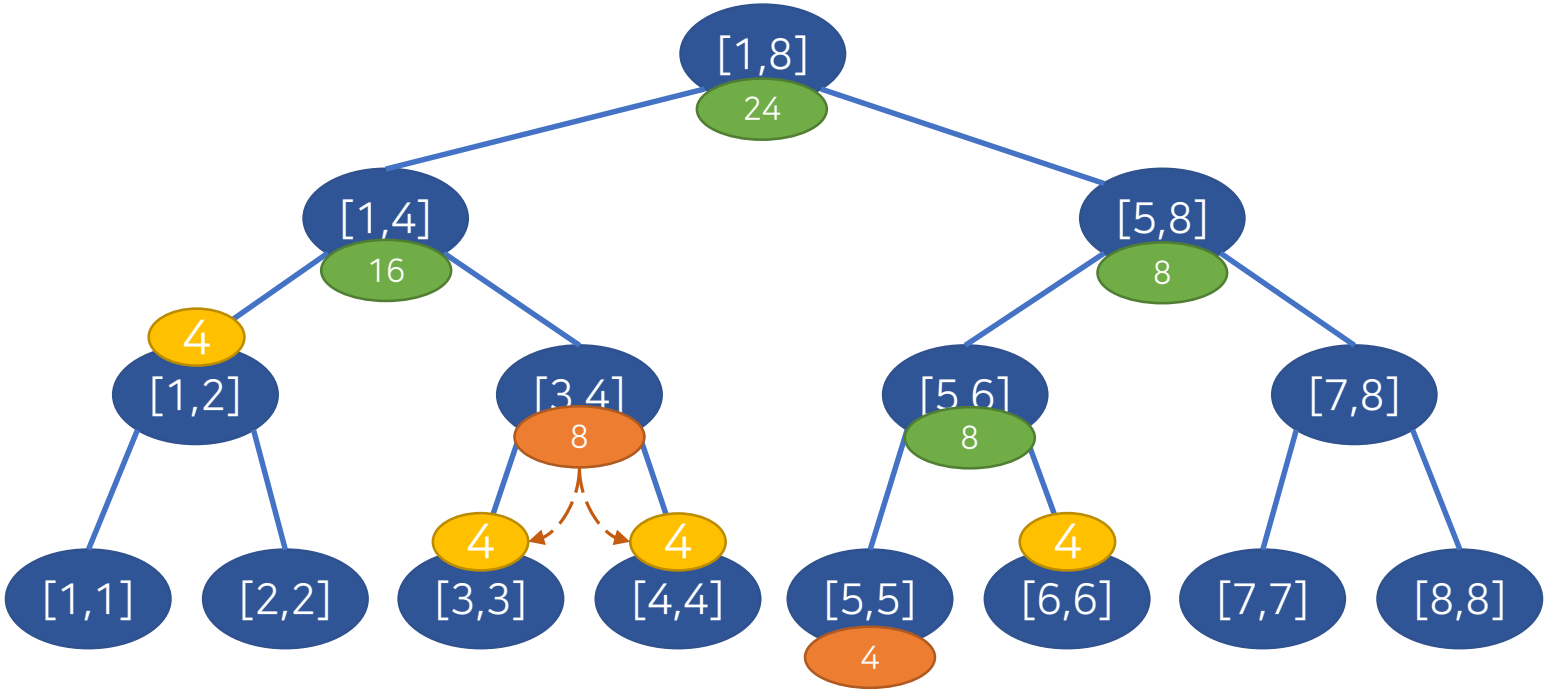
Example – Get

구간 [3,4], [5,5] 가 포함되면서, Lazy가 있으므로 적용하고 전파한 다음….



Example – Get

일반 Segment Tree와 마찬가지로 값을 반환하면서 쿼리의 답을 구하면 12.



Source Code – Update

```
1  void update(int idx, int curL, int curR, int dstL, int dstR, int val)
2  {
3          lazyUpdate(idx, curL, curR); //Lazy가 있는 경우 그것을 적용
4
5          if(curR < dstL || dstR < curL) return ; //update 구간을 벗어나는 경우
6
7          if(dstL <= curL && curR <= dstR){ //현재 구간이 update 구간에 포함되는 경우
8                  lazy[idx] += val;
9                  lazyUpdate(idx, curL, curR); //Lazy 생성 및 적용하고 반환
10                 return ;
11         }
12         int mid = (curL + curR) / 2; // 이 밑으로는 일반 Segment Tree와 동일!
13         update(2*idx, curL, mid, dstL, dstR, val);
14         update(2*idx+1, mid+1, curR, dstL, dstR, val);
15
16         tree[idx] = tree[2*idx] + tree[2*idx+1];
17 }
```

Source Code – LazyUpdate

```
1  void lazyUpdate(int idx, int L, int R)
2  {
3          if(lazy[idx] == 0) return ; //lazy 가 없는 경우
4
5          tree[idx] += (R – L + 1) * lazy[idx]; // lazy 적용
6
7          if(L != R){ //Leaf Node 가 아닌 경우, 자식 노드에 전파
8                  lazy[2*idx] += lazy[idx];
9                  lazy[2*idx+1] += lazy[idx];
10         }
11
12         lazy[idx] = 0; // lazy 초기화
13 }
14
15
16
17
```

Source Code – Get

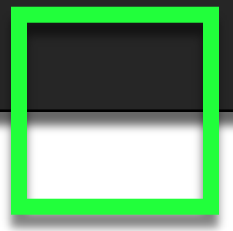
```
1  int Get(int idx, int curL, int curR, int dstL, int dstR)
2  {
3          lazyUpdate(idx, curL, curR); //Lazy가 있는 경우 그것을 적용
4
5          // 이 밑으로는 일반 Segment Tree와 동일!
6
7          if(curR < dstL || dstR < curL) return 0; //Get 구간을 벗어나는 경우
8
9          if(dstL <= curL && curR <= dstR){ //현재 구간이 Get 구간에 포함되는 경우
10                 return tree[idx];
11         }
12
13         int mid = (curL + curR) / 2;
14         return Get(2*idx, curL, mid, dstL, dstR)
15                 + Get(2*idx+1, mid+1, curR, dstL, dstR);
16 }
17
```

연습 문제

BOJ 10999
구간 합 구하기 2

#2

Plane Sweeping



What is Plane Sweeping?

평면 쓸기

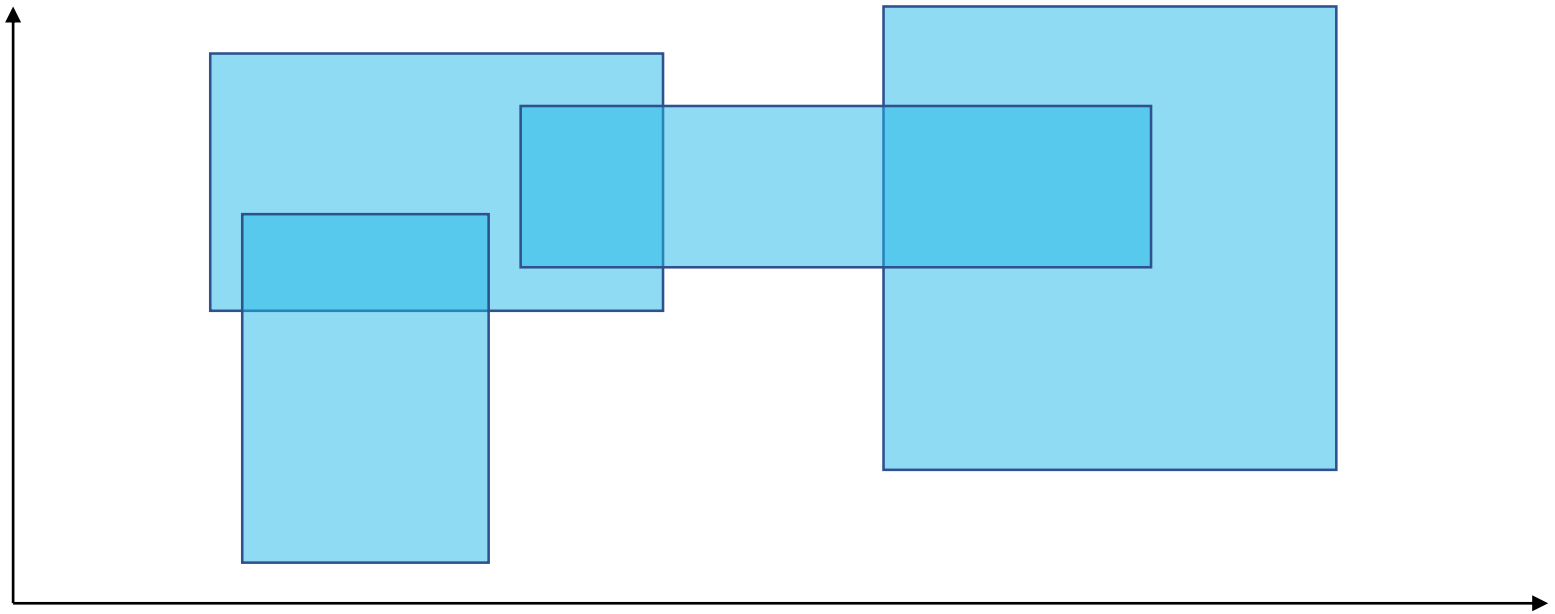
Plane Sweeping

Segment Tree 를 이용하여 2차원 평면에 관한 문제를 푸는 기법.

Plane Sweeping - Example

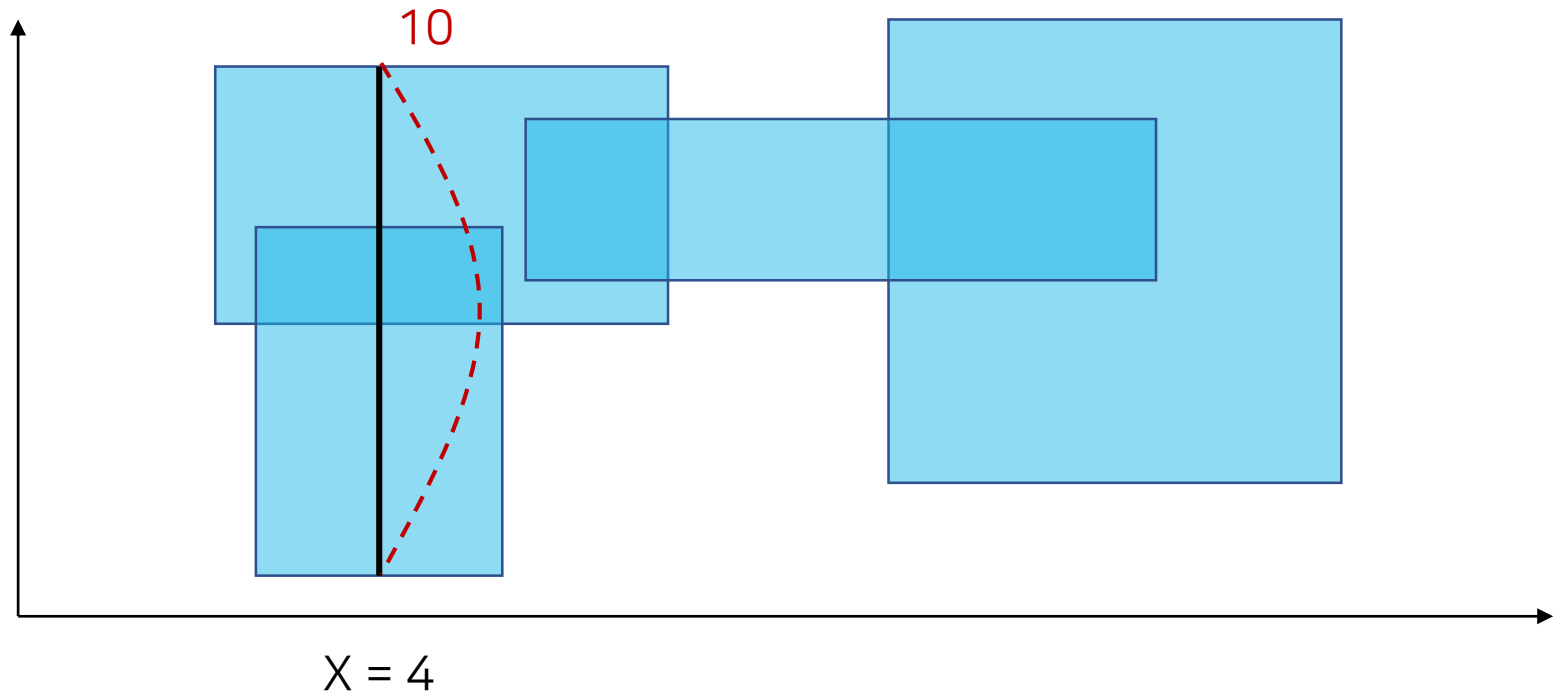
2차원 평면 위에 x, y 축과 평행한 직사각형 여러 개가 있다.

직사각형 끼리 겹치는 면적을 1번만 계산했을 때, 이 직사각형들이 차지하는 면적은 얼마일까?



Plane Sweeping - Example

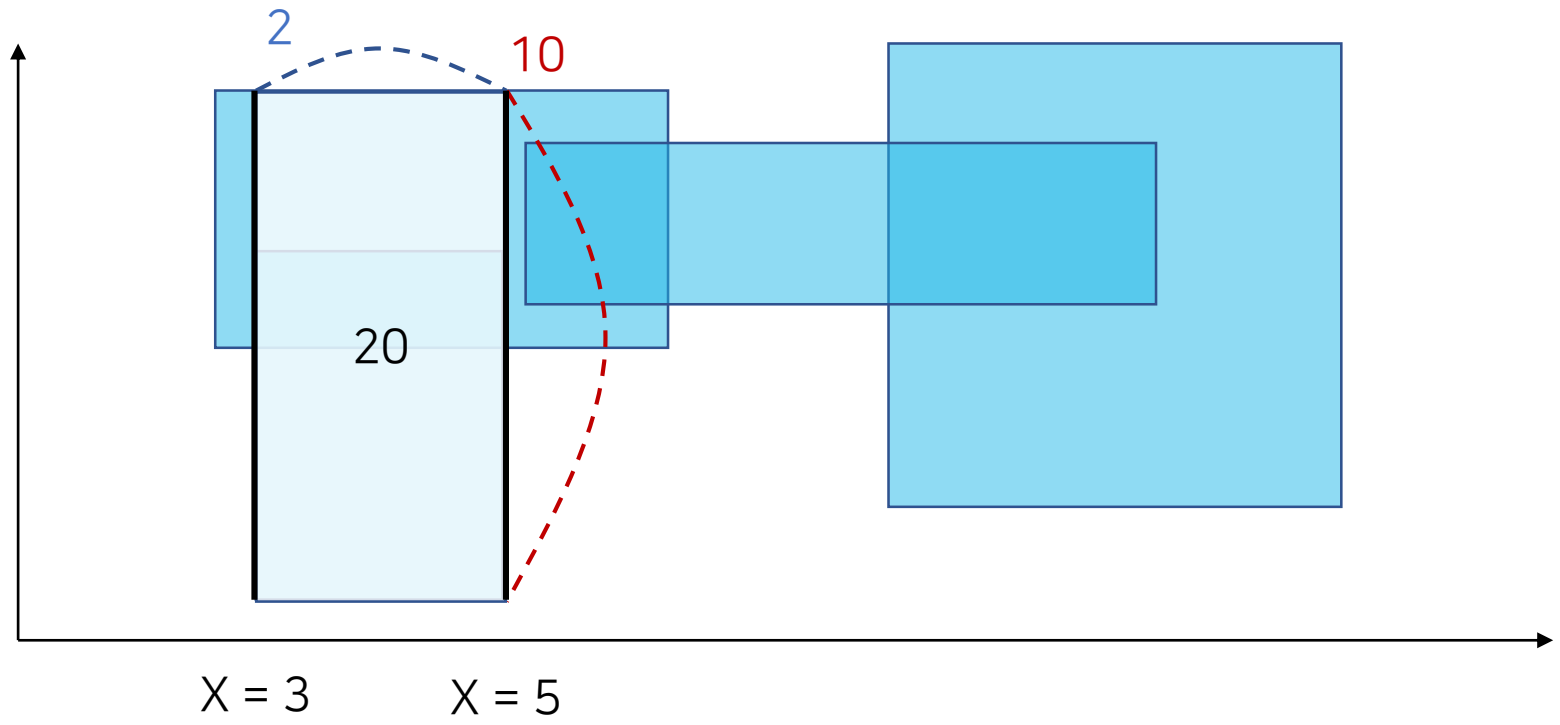
1. 어느 x 좌표에 대해서, 직사각형들이 y축에 평행하게 늘어져 있는 선분의 길이를 알 수 있다고 가정하자.



Plane Sweeping - Example

2. 그렇다면 그 길이가 변할 때마다, X 좌표 변화량 * Y축에 평행한 길이 = 그 구간의 차지하는 면적을 구할 수 있다.

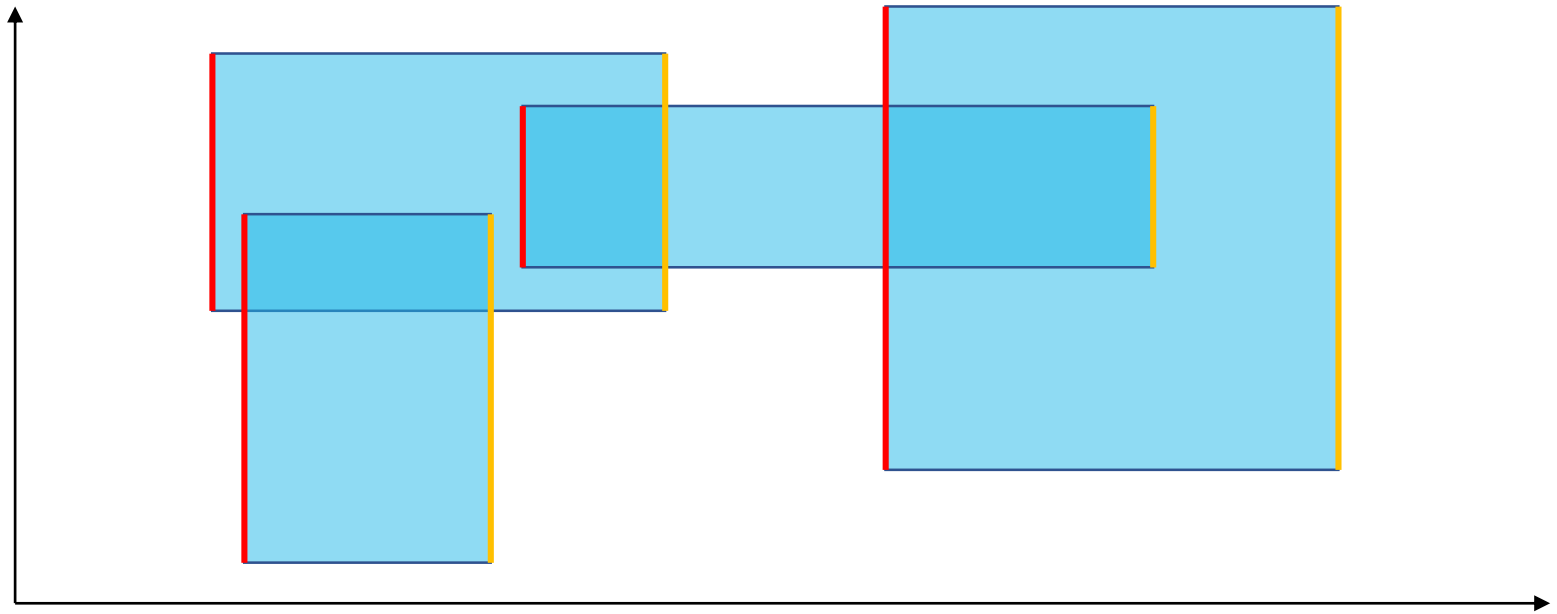
즉, 이 과정을 모든 X 구간에 대해 반복하면 문제를 해결할 수 있다!



Plane Sweeping - Example

직사각형이 Y축에 평행하게 늘어진 길이가 변할 수 있는 것은, 직사각형이 생기거나 사라질 때 뿐이므로

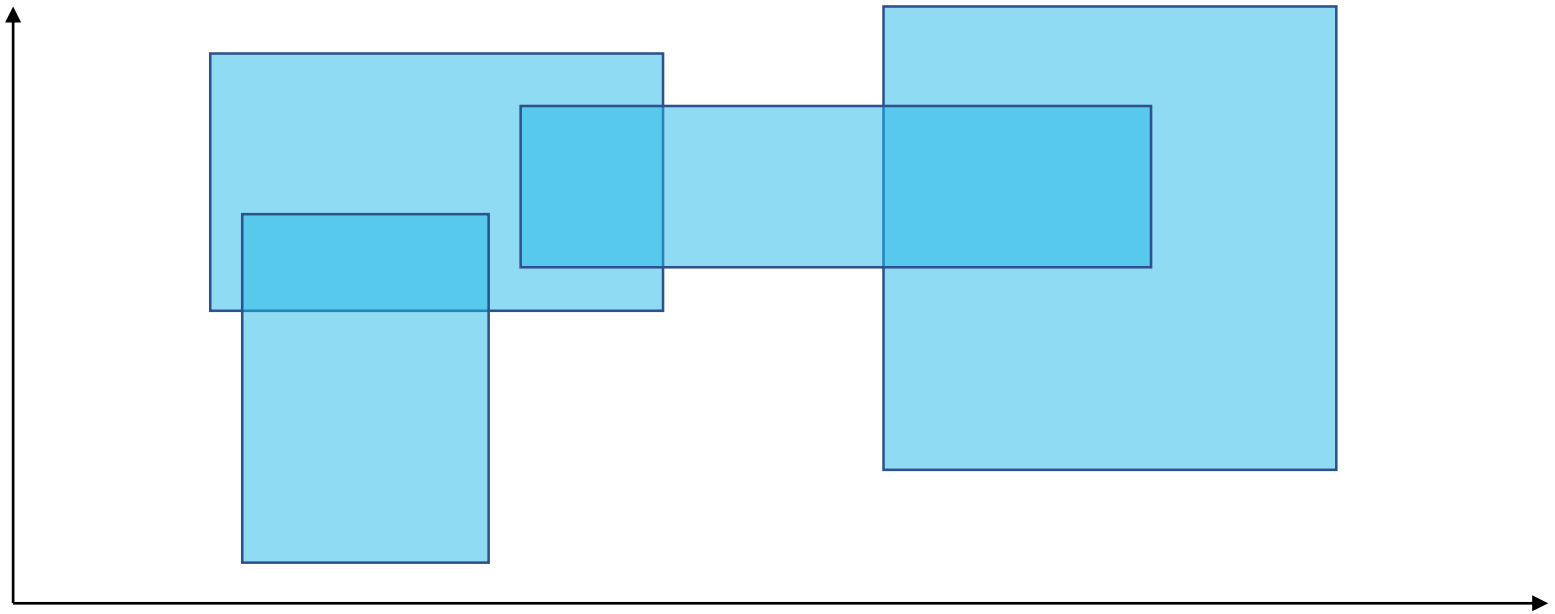
각 직사각형의 X좌표 시작지점과, 끝지점마다 쿼리 형태로 저장해, X 좌표 순으로 정렬하면 Ok.



Plane Sweeping - Example

그렇다면 이제 남은 것은 y축에 평행하게 늘어진 선분의 길이를 구하는 것.

드디어 Segment Tree 의 차례!



Plane Sweeping

평행한 선분은 연속적인 1차원 구간이므로, Segment Tree로 선분에 대한 정보를 저장할 수 있다.
즉, 처리만 잘하면 어느 x 좌표에 대해 y축과 평행한 선분에 관한 정보도 관리할 수 있다!

하지만 어떻게? 심지어 이걸 모든 x 좌표에 대해 할 수 있을까?

앞서서, 직사각형의 x좌표 시작점과 끝점을 쿼리로 만들어 놓았으니
x좌표가 증가하는 순으로 훑으면서 쿼리를 처리하는 것으로 가능하다!

Plane Sweeping

구현의 편의를 위해 하나의 Segment Tree에 cnt, sum 2가지의 정보를 저장할 것이다.
cnt = 현재 구간 위에 올라가있는 직사각형 개수, sum = 현재 구간에서 직사각형이 있는 구간 길이

Case 1. 직사각형의 x좌표 시작점

1. Segment Tree에서 직사각형의 세로 구간에 cnt를 +1 해준다.
2. 올라오면서 현재 Node의 cnt 가 0 보다 크면, sum 을 그 구간의 길이로 바꿔준다.
    I. 0인 경우에는 sum에 자식 노드의 sum 의 합을 저장한다.
    II. 자식 노드가 없는 경우에는 0을 저장한다.

Case 2. 직사각형의 x좌표 끝점

1. Segment Tree에서 직사각형의 세로 구간에 cnt를 -1 해준다.
2. 올라오면서 현재 Node의 cnt 가 0 보다 크면, sum 을 그 구간의 길이로 바꿔준다.
    I. 0인 경우에는 sum에 자식 노드의 sum 의 합을 저장한다.
    II. 자식 노드가 없는 경우에는 0을 저장한다.

Case 3. 넓이 계산

1. Root Node의 sum * 이전 쿼리와 현재 쿼리의 x좌표 차이 를 넓이 합에 더한다.

Source Code – Update

```
1  void update(int idx, int curL, int curR, int dstL, int dstR, int val)
2  {
3          if(curR < dstL || dstR < curL) return ; //update 구간을 벗어나는 경우
4
5          if(dstL <= curL && curR <= dstR) //현재 구간이 update 구간에 포함되는 경우
6  *               cnt[idx] += val;
7          else{
8                  int mid = (curL + curR) / 2;
9                  update(2*idx, curL, mid, dstL, dstR, val);
10                 update(2*idx+1, mid+1, curR, dstL, dstR, val);
11         }
12
13 *       if( cnt[idx] > 0 ) sum[idx] = curR - curL + 1;
14 *       else if( curL != curR ) sum[idx] = sum[2*idx] + sum[2*idx+1];
15 *       else sum[idx] = 0;
16 }
17
```

Where is Lazy Propagation?

Lazy Update 을 사용하지 않고도, 정확한 길이를 구할 수 있다는 것에 의문이 들 수도 있다.

그러나 잘 생각해보면 넓이를 구할 때 항상 Root Node 의 sum을 참조한다.

그리고 cnt의 전파는 Parent 에서 Child 방향으로 일어나지만
sum의 전파는 Child 에서 Parent 방향으로 일어나기 때문에
구간에 딱 맞는 트리의 위치를 발견했다면, 그 밑으로 cnt를 전파시킬 필요가 없는 것이다!

예시로 [3,8] 구간을 업데이트 할 때, [3,4], [5,8] 구간까지만 하면 되고
[3,3], [5, 6] 와 같은 하위 구간까지 업데이트를 할 필요가 없다!

어떻게 보면 전파를 아예 하지 않는 진정한 **Lazy** Propagation….

연습 문제

BOJ 3392
화성 지도

수고했어요!