Sweeping

2019-2020 Winter

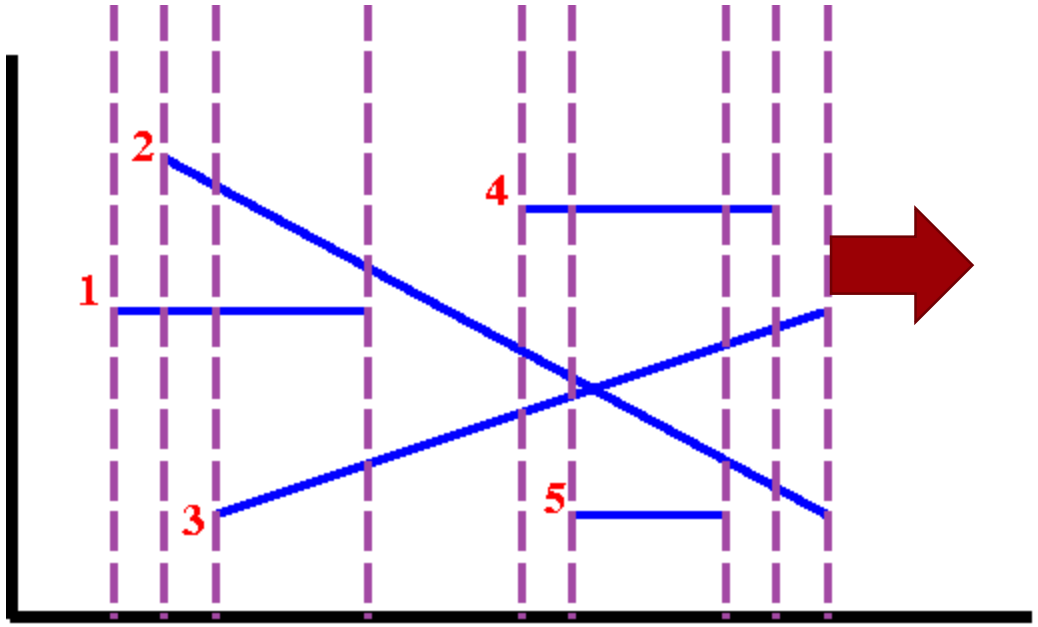
20141574 임지환 (Sogang University)

Index

- 개요
- 수직선 상에서의 sweeping problem
  - Line sweeping examples
- 평면 상에서의 sweeping problem
  - Closest Pair
  - Plane sweeping examples
  - Convex hull – Monotone Chain algorithm

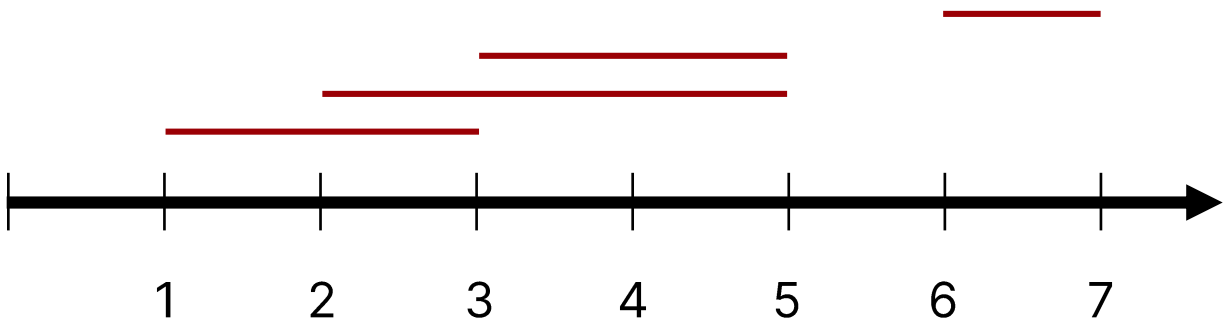
- 평면, 혹은 수직선 상에서의 기하 문제에서 사용
- 평면 상의 한 지점에 대응하는 어떤 **이벤트**들의 집합으로서 문제를 표현

  event : x값에 따라, y값에 따라, 혹은 어떠한 기준에 의거한 점에 따라 **순차적**으로 실행



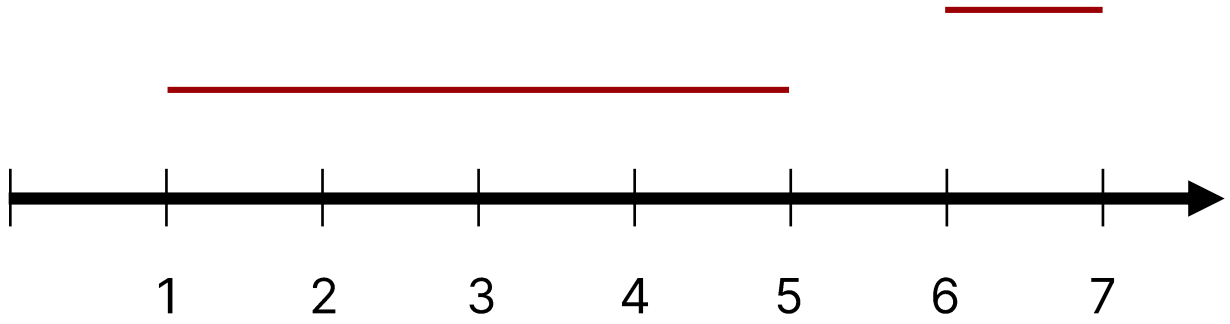
#2170 선 긋기

- 선분 $N(1 \leq N \leq 1{,}000{,}000)$개를 그을 때 그려진 선(들)의 총 길이?
- 선분 (l, r)의 범위 : $-10^9 \leq l, r \leq 10^9$



- **문제 해결 과정 1**

1. 구간이 겹치는 segment를 하나로 보자.
   - 다른 segment조각이 등장하는 경우 : 기존 segment의 right < 현재 segment의 left
2. segment의 정렬 기준 : left가 작은 순
3. 합쳐진 segment의 left와 right 값 갱신



#2170 선 긋기

```
17  sort(arr.begin(), arr.end());
18
19  int ans = 0, l = -1'000'000'001, r = -1'000'000'001;
20  for (int i = 0; i < n; i++) {
21      if (r < arr[i].first) {
22          ans += r - l;
23          l = arr[i].first;
24          r = arr[i].second;
25      }
26      else r = max(r, arr[i].second);
27  }
28  ans += r - l;
```

#17 : left가 작은 순, right가 작은 순

#21~25 : 분리된 segment가 등장한 경우

#21~26 : 합쳐진 segment의 left와 right값 갱신

#22, 28 : total length = 합쳐진 segment들의 길이의 합

#2170 선 긋기

```
17  sort(arr.begin(), arr.end());
18
19  int ans = 0, l = -1'000'000'001, r = -1'000'000'001;
20  for (int i = 0; i < n; i++) {
21      if (r < arr[i].first) {
22          ans += r - l;
23          l = arr[i].first;
24          r = arr[i].second;
25      }
26      else r = max(r, arr[i].second);
27  }
28  ans += r - l;
```

#17 : left가 작은 순, right가 작은 순

#21~25 : 분리된 segment가 등장한 경우

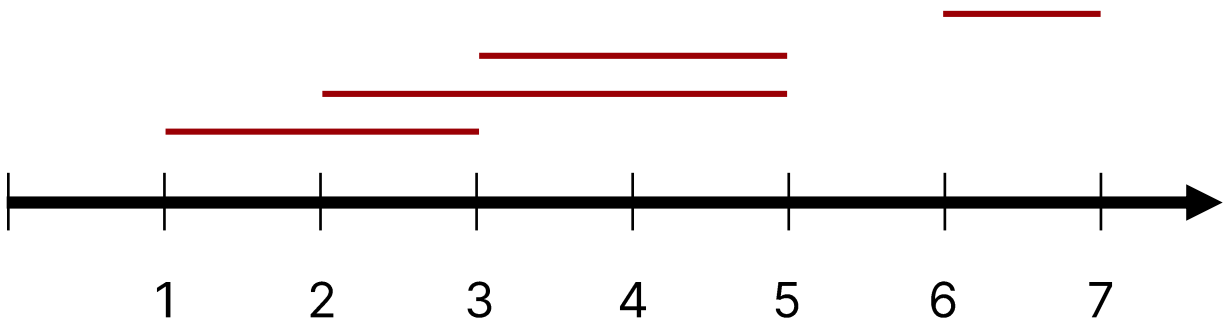
#21~26 : 합쳐진 segment의 left와 right값 갱신

#22, 28 : total length = 합쳐진 segment들의 길이의 합

$$O(NlogN)$$

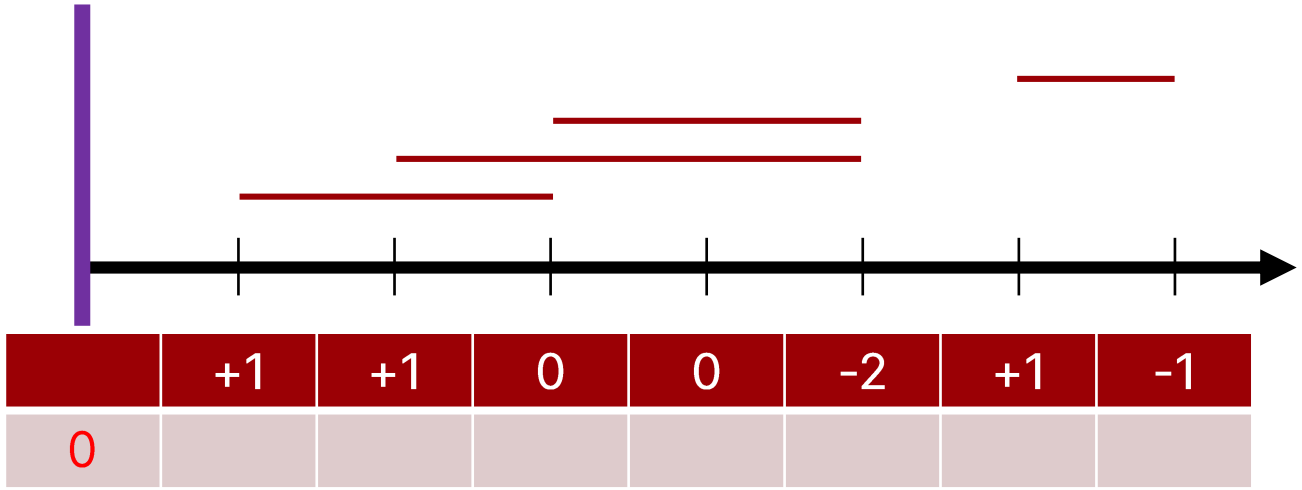
- 문제 해결 과정 2

1. 정렬을 하지 않고 $O(N)$에 구간 정보 처리
   - update 중간에 쿼리 요청이 없는 경우 유효한 방법
2. 구간의 시작 지점에 +, 구간의 끝 지점에 -



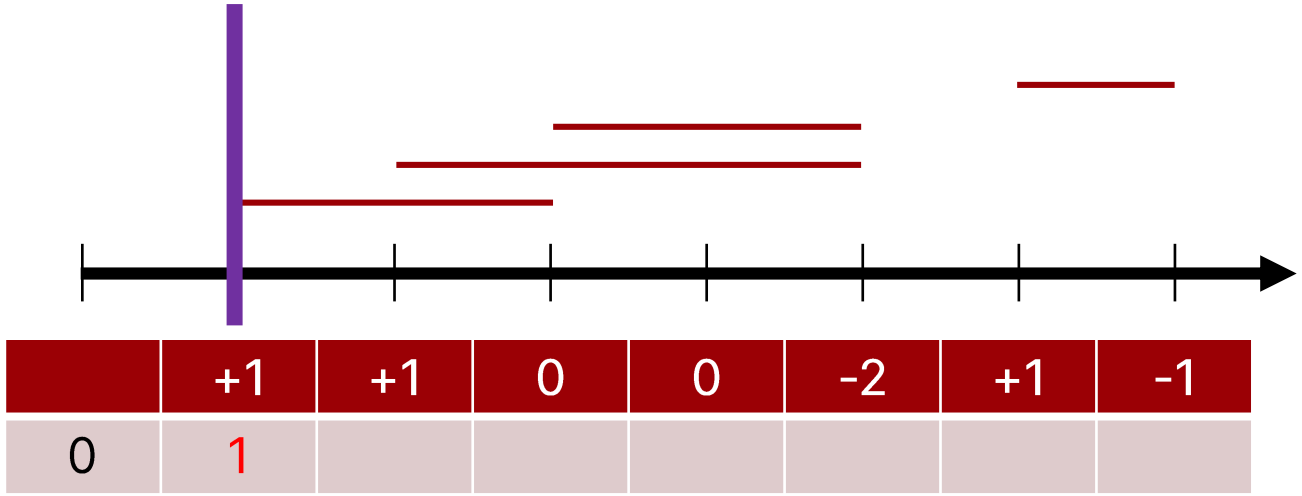
- 문제 해결 과정 2

1. 정렬을 하지 않고 $O(N)$에 구간 정보 처리
   - update 중간에 쿼리 요청이 없는 경우 유효한 방법
2. 구간의 시작 지점에 +, 구간의 끝 지점에 -



- 문제 해결 과정 2

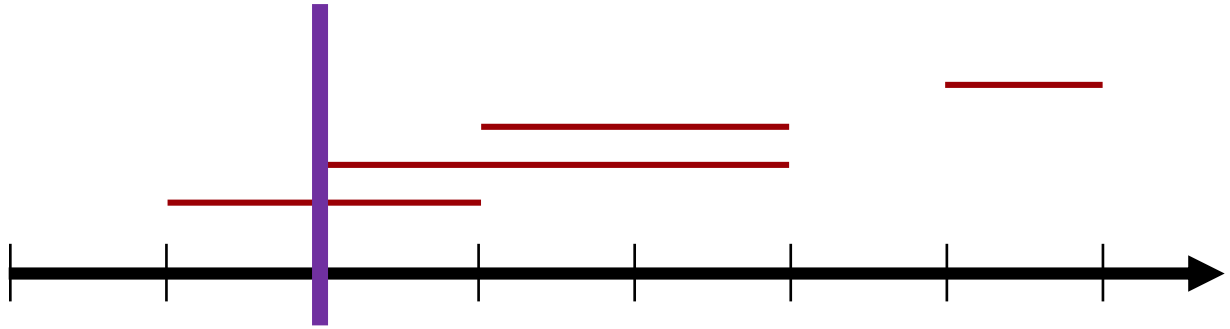
1. 정렬을 하지 않고 $O(N)$에 구간 정보 처리
   - update 중간에 쿼리 요청이 없는 경우 유효한 방법
2. 구간의 시작 지점에 +, 구간의 끝 지점에 -



	+1	+1	0	0	-2	+1	-1
0	1						

- 문제 해결 과정 2

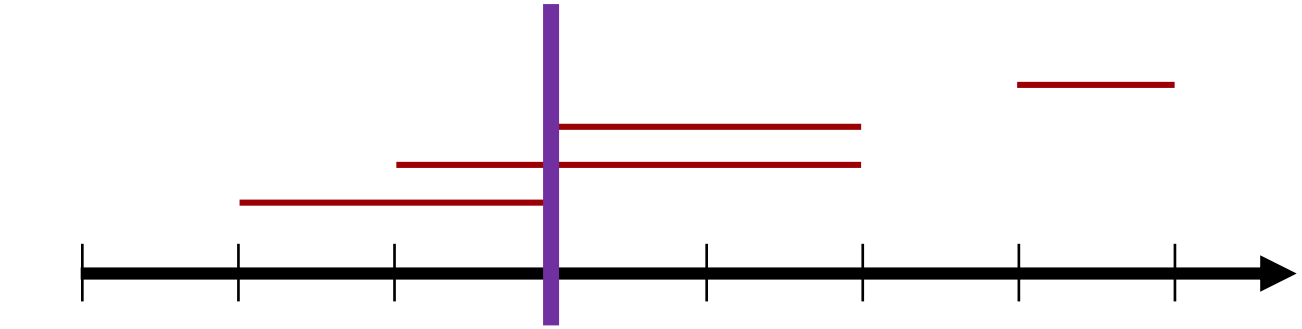
1. 정렬을 하지 않고 $O(N)$에 구간 정보 처리
   - update 중간에 쿼리 요청이 없는 경우 유효한 방법
2. 구간의 시작 지점에 +, 구간의 끝 지점에 -



	+1	+1	0	0	-2	+1	-1
0	1	2					

- 문제 해결 과정 2

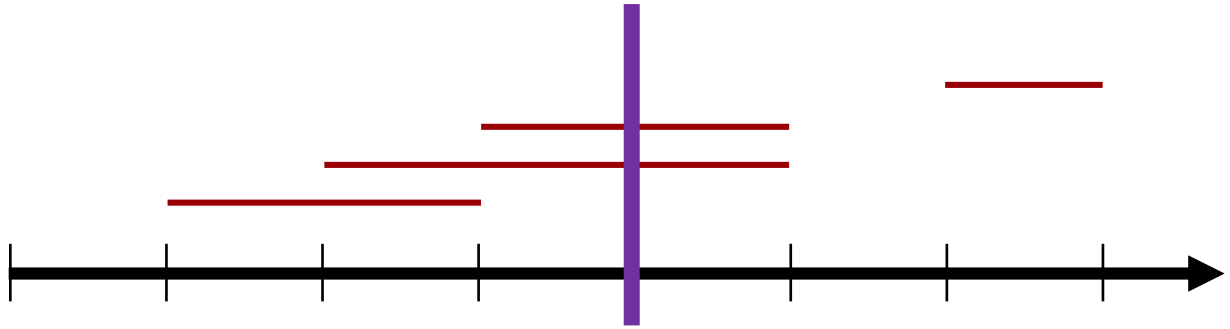
1. 정렬을 하지 않고 $O(N)$에 구간 정보 처리
   - update 중간에 쿼리 요청이 없는 경우 유효한 방법
2. 구간의 시작 지점에 +, 구간의 끝 지점에 -



	+1	+1	0	0	-2	+1	-1
0	1	2	2				

- 문제 해결 과정 2

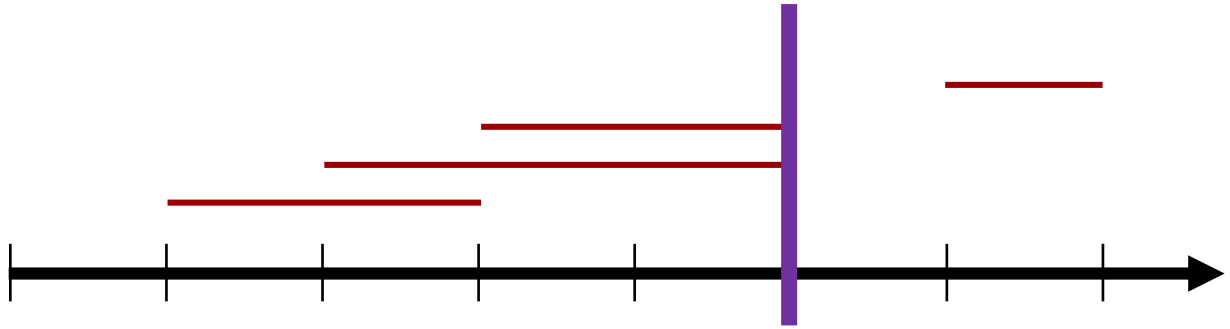
1. 정렬을 하지 않고 $O(N)$에 구간 정보 처리
   - update 중간에 쿼리 요청이 없는 경우 유효한 방법
2. 구간의 시작 지점에 +, 구간의 끝 지점에 -



	+1	+1	0	0	-2	+1	-1
0	1	2	2	2			

- 문제 해결 과정 2

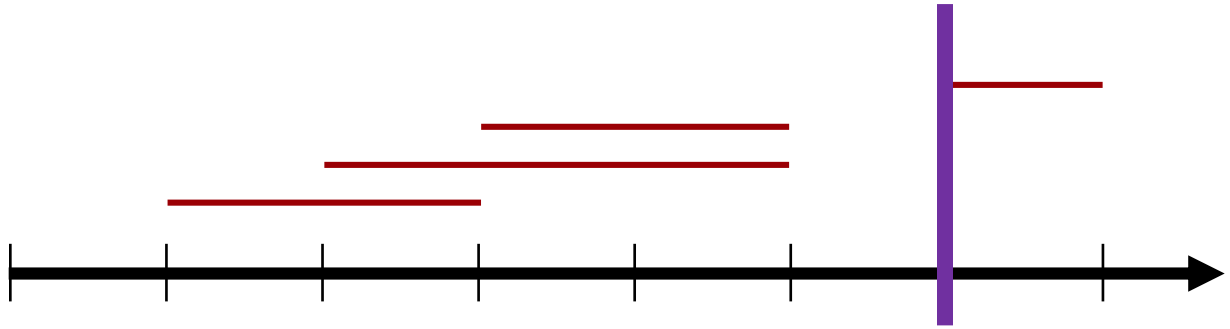
1. 정렬을 하지 않고 $O(N)$에 구간 정보 처리
   - update 중간에 쿼리 요청이 없는 경우 유효한 방법
2. 구간의 시작 지점에 +, 구간의 끝 지점에 -



	+1	+1	0	0	-2	+1	-1
0	1	2	2	2	0		

- 문제 해결 과정 2

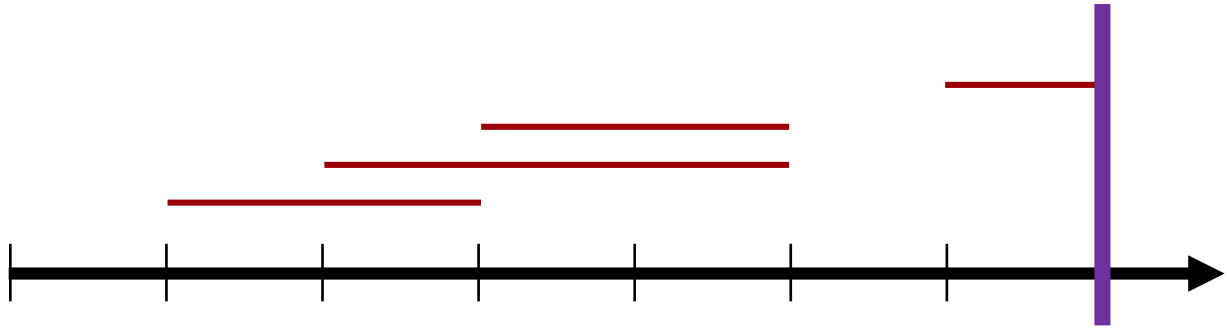
1. 정렬을 하지 않고 $O(N)$에 구간 정보 처리
   - update 중간에 쿼리 요청이 없는 경우 유효한 방법
2. 구간의 시작 지점에 +, 구간의 끝 지점에 -



	+1	+1	0	0	-2	+1	-1
0	1	2	2	2	0	1	

- 문제 해결 과정 2

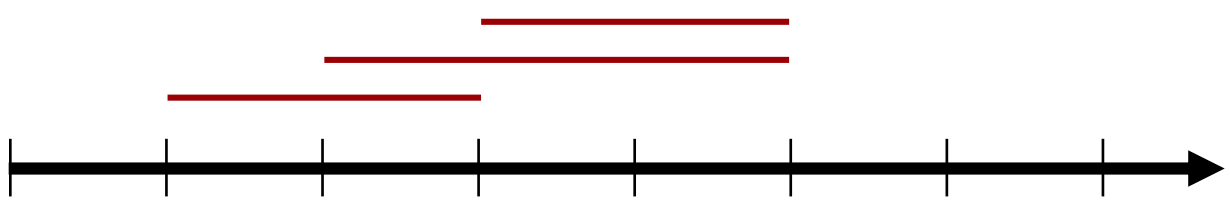
1. 정렬을 하지 않고 $O(N)$에 구간 정보 처리
   - update 중간에 쿼리 요청이 없는 경우 유효한 방법
2. 구간의 시작 지점에 +, 구간의 끝 지점에 -



	+1	+1	0	0	-2	+1	-1
0	1	2	2	2	0	1	0

- 문제 해결 과정 2

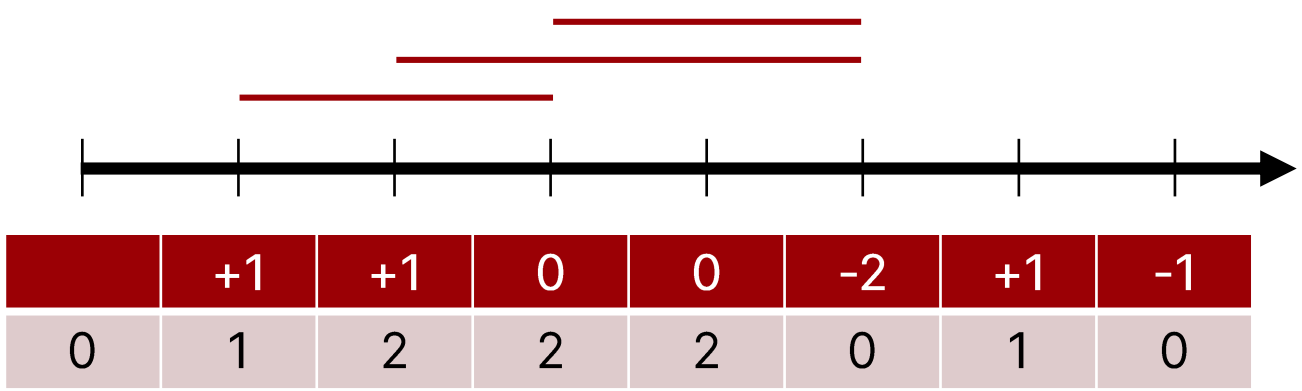
1. 정렬을 하지 않고 $O(N)$에 구간 정보 처리
   - update 중간에 쿼리 요청이 없는 경우 유효한 방법
2. 구간의 시작 지점에 +, 구간의 끝 지점에 –
3. Sweeping 이후 0이 아닌 지점이 나타내는 값들의 실질적인 길이들의 합 구하기



	+1	+1	0	0	-2	+1	-1
0	1	2	2	2	0	1	0

- 문제 해결 과정 2

1. 정렬을 하지 않고 $O(N)$에 구간 정보 처리
   - update 중간에 쿼리 요청이 없는 경우 유효한 방법
2. 구간의 시작 지점에 +, 구간의 끝 지점에 –
3. Sweeping 이후 0이 아닌 지점이 나타내는 값들의 실질적인 길이들의 합 구하기



	+1	+1	0	0	-2	+1	-1
0	1	2	2	2	0	1	0

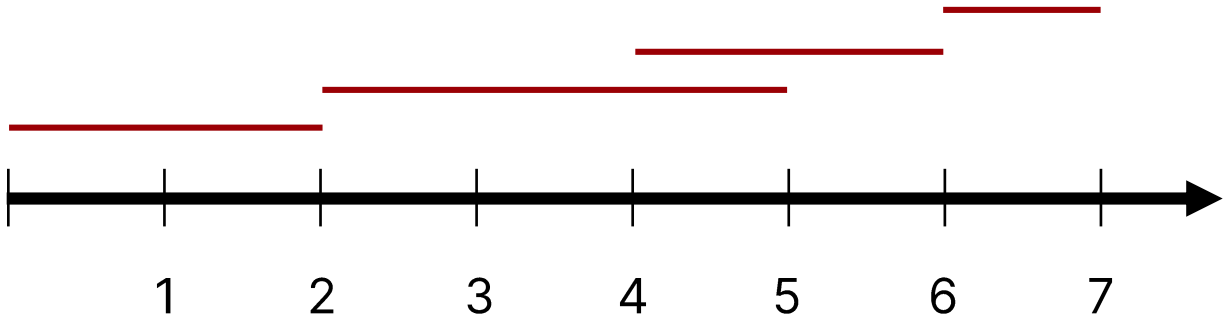
$O(N)$

#2170 선 긋기

```
27  grid_compression();
28
29  for (int i = 0; i < N; i++) {
30      l[i] = lower_bound(xl.begin(), xl.end(), l[i]) - xl.begin();
31      r[i] = lower_bound(xl.begin(), xl.end(), r[i]) - xl.begin();
32      a[l[i]]++, a[r[i]]--;
33  }
34  long long ans = 0;
35  for (int i = 1; i < (int)xl.size() - 1; i++) {
36      a[i] += a[i - 1];
37      if (a[i]) ans += xl[i + 1] - xl[i];
38  }
```

#15589 Lifeguards (Silver)

- time domain이 $[0, 10^9]$인 interval에 $N(1 \leq N \leq 10^5)$개의 근무 시간 정보가 주어짐.
- 단 하나의 근무자만 없앤다 할 때, 다른 근무자들에 의해 cover되는 시간의 최댓값?

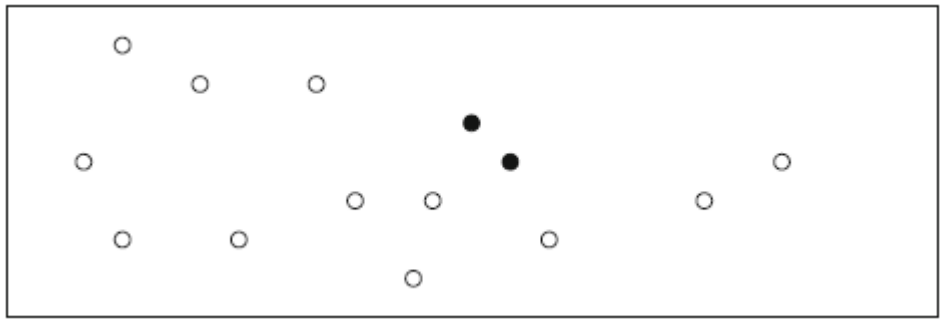


평면 상에서의 Sweeping Problem

- 가장 가까운 두 점
- Plane Sweeping Examples
- Convex hull algorithm – Andrew’s monotone chain algorithm

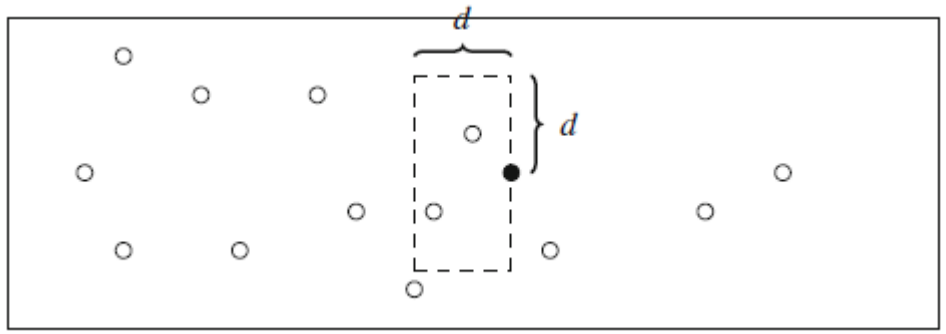
가장 가까운 두 점

- 왼쪽부터 오른쪽으로 sweeping해가며 진행



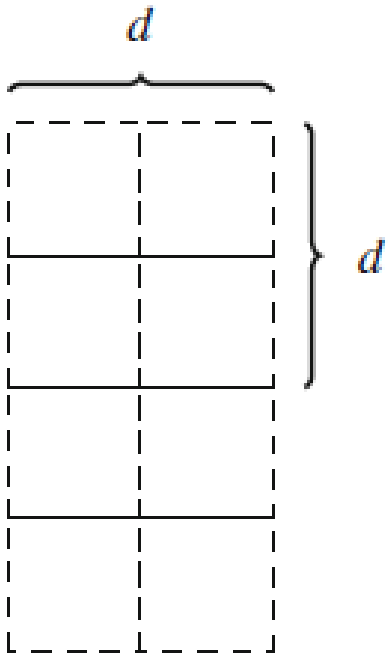
가장 가까운 두 점

- 왼쪽부터 오른쪽으로 sweeping해가며 진행
- 현재까지 구한 두 점 사이의 최소 거리 d를 관리
- 현재 보고 있는 점을 기준으로 왼쪽에 있는 $d \times 2d$의 사각형 내의 점만 확인



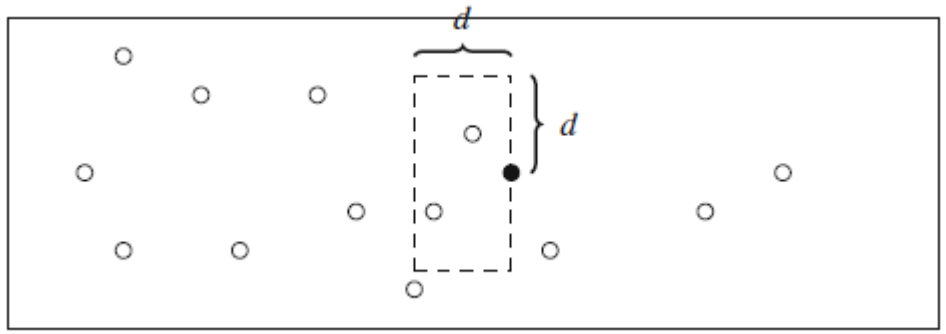
가장 가까운 두 점

- 현재 보고 있는 점의 왼쪽 점들에 의해 구해진 최소 거리 : d
- 비둘기집 원리에 의해 각 $\frac{d}{2} \times \frac{d}{2}$ 사각형 내부에는 최대 하나의 점만이 존재해야 최소 거리가 d 라는 조건을 만족함



가장 가까운 두 점

- 왼쪽부터 오른쪽으로 sweeping해가며 진행
- 현재까지 구한 두 점 사이의 최소 거리 d를 관리
- 현재 보고 있는 점을 기준으로 왼쪽에 있는 $d \times 2d$의 사각형 내의 점만 확인 => $O(1)$개
- 정렬 후 sweeping => $O(NlogN + N) = O(NlogN)$



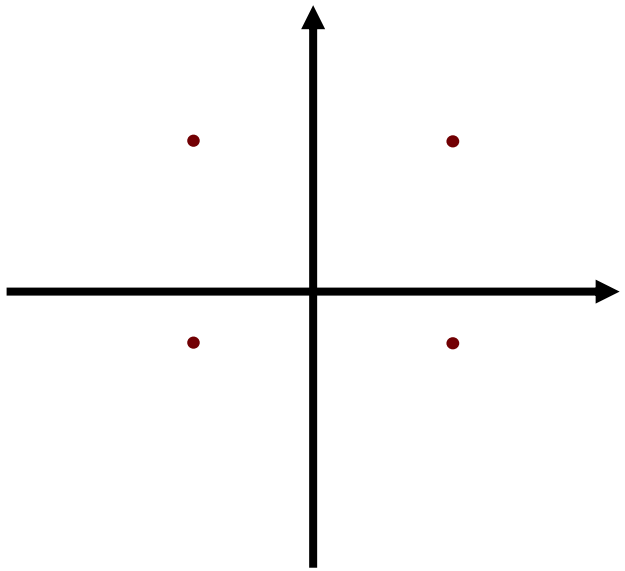
BOJ blog link

Plane sweeping examples

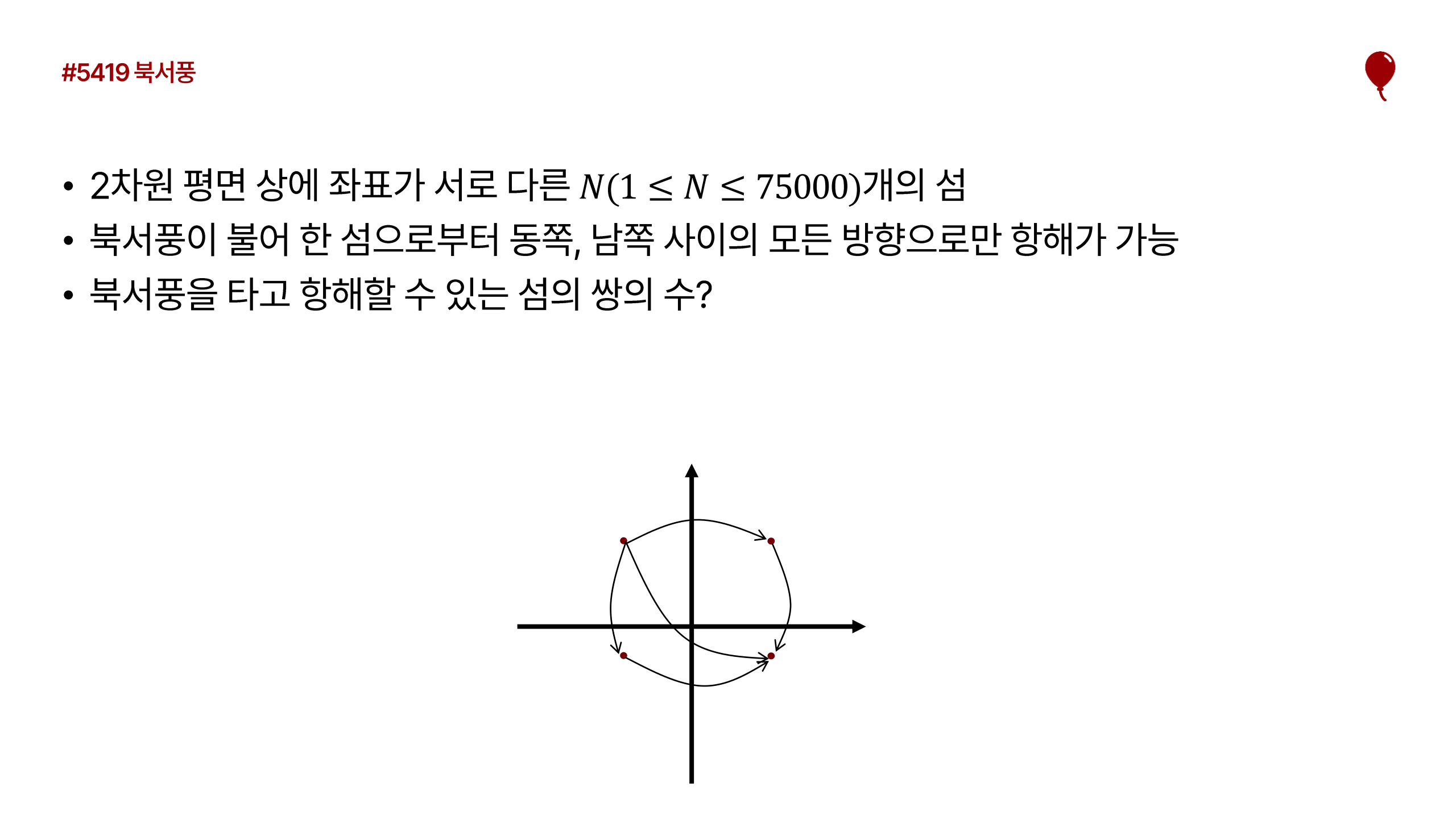
- sweeping : x값에 따라, y값에 따라, 혹은 **어떠한 기준**에 의거한 점에 따라 순차적으로 실행

#5419 북서풍

- 2차원 평면 상에 좌표가 서로 다른 $N(1 \leq N \leq 75000)$개의 섬
- 북서풍이 불어 한 섬으로부터 동쪽, 남쪽 사이의 모든 방향으로만 항해가 가능
- 북서풍을 타고 항해할 수 있는 섬의 쌍의 수?



#5419 북서풍



- 2차원 평면 상에 좌표가 서로 다른 $N(1 \leq N \leq 75000)$개의 섬
- 북서풍이 불어 한 섬으로부터 동쪽, 남쪽 사이의 모든 방향으로만 항해가 가능
- 북서풍을 타고 항해할 수 있는 섬의 쌍의 수?

- Sweeping on line

#2170 선 긋기
#2492 보석
#2672 여러 직사각형의 전체 면적
#13334 철로
#15589 Lifeguards (Silver)

- Sweeping on plane

#2261 가장 가까운 두 점
#3392 화성지도
#5419 북서풍