"안전 지대 (safezone)" 문제 풀이

작성자: 박상훈, 구재현

부분문제 1

$B[i] \le 0 \le D[i]$ 조건에 따라서, 직사각형을 $[A[i], C[i]]$ 구간의 점을 포함하는 수직선 상 선분으로 생각할 수 있다.

만약에 i 번 선분과 $j(>i)$ 번 선분이 같은 연합에 속한다면, $i, i+1, \ldots, j-1, j$ 번 선분이 모두 같은 연합에 속함을 알 수 있다. 각 $0 \le i \le N-2$ 에 대해서, i 번 선분과 $i+1$ 번 선분이 같은 연합에 속한다는 것은, $\max_{j \le i} C[j] \ge \min_{j>i} A[j]$ 를 만족한다는 것과 동치이다. Prefix/Suffix Minimum을 취하여 이를 판단할 수 있고, 이 값을 토대로 각 선분이 속하는 연합을 구할 수 있다.

부분문제 2

선분을 $A[i]$ 가 증가하는 순으로 정렬하면, 부분문제 1과 같은 풀이가 성립한다.

부분문제 3

$O(N^2)$ 시간에 그래프를 구성한 후 연결 컴포넌트를 계산하여 해결할 수 있다.

부분문제 4

$A[i] = C[i]$ 인 직사각형을 *세로 선분*, $B[i] = D[i]$ 인 선분을 *가로 선분* 이라고 부르자. 또한, $O(N \log N)$ 시간에 모든 직사각형 좌표를 $[0, 2N-1]$ 범위로 줄여주자 (좌표 압축).

x 좌표가 증가하는 방향으로 스위핑한다. 가로 선분은 $x = A[i]$ 시점에 삽입, $x = C[i]+1$ 시점에 삭제되며, 세로 선분은 $x = A[i]$ 시점에 처리된다. 세로 선분이 들어올 때, 현재 자료구조에 있는 가로 선분 중 y 좌표가 $[B[i], D[i]]$ 사이에 있는 선분들을 모두 모아서, 이들을 같은 연결 컴포넌트에 넣어주어야 한다.

고로, 다음 연산을 빠르게 하는 자료구조 T가 필요하다:

- UPDATE(i, x): $T[i] = x$ 로 값 설정.
- MERGE(l, r, x): $T[l], T[l+1], \ldots, T[r]$ 중 -1 이 아닌 모든 $T[i]$ 에 대해, $T[i]$ 와 x 간에 간선 추가 (혹은 Union-Find에서 같은 정점으로 합쳐줌)

이는 세그먼트 트리를 사용하여 해결할 수 있다. 세그먼트 트리의 각 노드에 대해서 다음 세 값을 관리하자:

- C: 해당 구간 안에서 $T[i] \ne -1$ 인 i 의 수
- L: 해당 구간에 있는 모든 $T[i] \ne -1$ 에 대해서, 간선을 추가해 줘야 할 컴포넌트 (없으면 -1)
- V: (리프 노드의 경우) $T[i]$

C, V 는 어렵지 않게 관리할 수 있다. L 은 Lazy Propagation을 사용하여 관리한다. 부모 구간에 있는 L 을 자식 구간으로 전파시키는 연산을 다음과 같이 처리한다: 자식이 $C = 0$ 일 경우 무시하고, 자식 구간의 L 이 -1 이 아닐 경우 자식 구간과 부모 구간 간에 간선을 추가하자. 이제, 자식 구간의 L은 -1 이거나 이미 부모와 연결되어 있으니 부모의 L 을 덮어쓰면 된다. 이렇게 할 경우 어떠한 노드의 L 을 자식으로

전파시켜줄 수 있다. 이를 사용하여 MERGE 연산은 $O(\log N)$ 개의 구간에만 적용해 주고, UPDATE 연산을 수행할 때도 $O(\log N)$ 개의 구간에 대해서 전파를 수행하여 $L = -1$ 인 상황을 만들어 줄 수 있다.

시간 복잡도는, 간선 추가가 $O(1)$ 에 된다고 가정하였을 때 $O(N \log N)$ 이다.

이 풀이에 분할 정복을 추가하여 전체 문제를 $O(N \log N)$ 에 해결하는 별해가 존재한다.

부분문제 5

$O(N^2)$ 보다 효율적이지만 의도된 풀이만큼 빠르지는 않은 풀이들은 부분문제 5를 맞을 수 있다. 출제진이 작성한 모든 $O(N\sqrt{N})$ 풀이와 일부 $O(N \log^2 N)$ 풀이가 이에 해당한다.

부분문제 6

x 좌표가 증가하는 방향으로 스위핑한다. y좌표를 인덱스로 하는 세그먼트 트리를 관리하면 현재 x좌표에서 특정 y좌표 구간에 직사각형이 존재하는지 $O(\log N)$에 판별할 수 있다. y좌표 구간 $[l, r]$에 직사각형이 있다면, i번 직사각형과 같은 컴포넌트라는 것을 의미하는 (l, r, i) 튜플을 관리하는 자료구조를 생각하자. 직사각형을 삭제할 때, 실제로 직사각형이 존재하는 구간과 일대일 대응되도록 튜플을 저장하면 최악의 경우 튜플의 개수가 $O(N)$씩 변하게 되어 전체 시간복잡도가 $O(N^2)$ 이상이 된다. 그러나, 직사각형이 삭제되어도 저장되어 있는 튜플은 여전히 정의를 만족하므로 정보를 갱신할 필요가 없다. 따라서, 직사각형을 새로 삽입하는 경우만 추가로 관리해주면 된다. 직사각형을 새로 삽입할 때 교집합이 존재하는 튜플을 모두 순회하며 새로운 튜플 하나로 합쳐주게 되면 삽입/삭제는 $O(N)$번 발생하게 된다. 이때, 튜플로 표현된 구간에 직사각형이 실제로 존재하는지 확인하기 위해 앞에서 언급한 세그먼트 트리를 사용해야 하며, Union-Find로 컴포넌트를 관리하면 된다. 튜플을 저장하는 자료구조는 `std::set`을 사용하면 된다. 시간복잡도는 $O(N \log N)$이다.