"멋진 구간 (maxsum)" 문제 풀이

작성자: 구재현, 박상훈

부분문제 1

어떤 구간 $[l, r]$이 멋진 구간일 필요충분조건은 다음과 같다: $l \le i \le r$인 i에 대해 $C[i] = B[i]$, 그렇지 않은 j에 대해서는 $C[j] = A[j]$라고 할 때, $[l, r]$이 C의 최대 부분배열이다. 배열 C에서 Kadane's Algorithm을 적용하면 $O(N)$ 시간에 멋진 구간인지 판별할 수 있다. 따라서, 문제를 $O(N^3 + Q)$ 시간에 해결할 수 있다.

부분문제 2

배열 D와 구간 $[l, r]$에 대해, $\text{info}_D(l, r) = (a, b, c, d)$를 다음과 같이 정의하자.

- a는 $D[l] + \cdots + D[i]$의 최댓값이다. (단, $l \le i \le r$) 즉, a는 $D[l..r]$의 prefix sum 중 최댓값이다.
- b는 $D[i] + \cdots + D[r]$의 최댓값이다. (단, $l \le i \le r$) 즉, b는 $D[l..r]$의 suffix sum 중 최댓값이다.
- c는 $D[i] + \cdots + D[j]$의 최댓값이다. (단, $l \le i \le j \le r$) 즉, c는 $D[l..r]$의 구간합 중 최댓값이다.
- d는 $D[l] + \cdots + D[j]$이다.

$\text{info}_D(l, m) = (a_1, b_1, c_1, d_1)$과 $\text{info}_D(m+1, r) = (a_2, b_2, c_2, d_2)$가 주어졌을 때, $\text{info}_D(l, r) = (a_3, b_3, c_3, d_3)$의 값은 다음과 같이 $O(1)$ 시간에 계산할 수 있다. 이러한 연산을 $+$를 사용하여 $\text{info}_D(l, m) + \text{info}_D(m+1, r)$과 같이 표기한다.

- $a_3 = \max(a_1, d_1 + a_2)$
- $b_3 = \max(b_2, d_2 + b_1)$
- $c_3 = \max(c_1, c_2, b_1 + a_2)$
- $d_3 = d_1 + d_2$

따라서, $\text{info}_A(i, j)$와 $\text{info}_B(i, j)$를 $O(N^2)$ 시간에 모두 계산할 수 있다. 구간 $[l, r]$이 멋진 구간인지 판별하려면 $\text{info}_A(0, l-1) + \text{info}_B(l, r) + \text{info}_A(r+1, N-1)$을 계산하면 충분하다. 따라서, 문제를 $O(N^2 + Q)$ 시간에 해결할 수 있다.

부분문제 4

info_A와 info_B를 세그먼트 트리에 저장하면 $O(\log N)$ 시간에 주어진 구간이 멋진 구간인지 확인할 수 있다. 따라서, 부분문제를 $O(N + Q \log N)$ 시간에 해결할 수 있다.

부분문제 3

다음 6가지 조건을 모두 만족하는 것은 구간 $[l, r]$이 멋진 구간인 것과 동치이다.

1. $A[i] + \cdots + A[l-1] > 0$인 i가 존재하지 않는다. $(0 \le i \le l-1)$

2. $A[r+1] + \cdots + A[i] > 0$인 i가 존재하지 않는다. $(r+1 \le i \le N-1)$

3. $B[l] + \cdots + B[i] < 0$인 i가 존재하지 않는다. $(l \le i < r)$

4. $B[i] + \cdots + B[r] < 0$인 i가 존재하지 않는다. $(l < i \le r)$

5. 어떤 i가 존재하여, 임의의 x, y에 대해 $A[x] + \cdots + A[y] \le B[l] \cdots + B[i]$을 만족한다. $(l \le i \le r,$ $r+1 \le x \le y \le N-1)$

6. 어떤 i가 존재하여, 임의의 x, y에 대해 $A[x] + \cdots + A[y] \le B[i] \cdots + B[r]$을 만족한다. $(l \le i \le r,$ $0 \le x \le y \le l-1)$

1번과 2번 조건은 각각 l, r에만 의존하는 조건이다. $O(N)$ 시간에 모든 l과 r에 대해 이를 확인할 수 있다.

3번 조건의 경우, 어떤 함수 f_3가 존재하여 $r \le f_3(l)$일 때만 3번 조건을 만족한다. Sparse table과 이분탐색을 사용하면 $O(N \log N)$ 시간에 $f_3(0), \cdots, f_3(N-1)$을 계산할 수 있다.

4번 조건의 경우, 어떤 함수 f_4가 존재하여 $l \ge f_4(r)$일 때만 4번 조건을 만족한다. 3번 조건과 동일한 형태이므로 $O(N \log N)$ 시간에 $f_4(0), \cdots, f_4(N-1)$을 계산할 수 있다.

5번 조건의 경우, l을 고정하고 r을 증가시키면 $A[x] + \cdots + A[y]$의 최댓값은 감소하고, $B[l] + \cdots + B[i]$의 최댓값은 증가하므로, 처음에는 조건을 만족하지 않다가 특정 지점부터 조건을 만족하기 시작한다. 따라서, 어떤 함수 f_5가 존재하여 $r \ge f_5(l)$일 때만 5번 조건을 만족한다. Sparse table과 이분탐색을 사용하면 $O(N \log N)$ 시간에 $f_5(0), \cdots, f_5(N-1)$을 계산할 수 있다.

6번 조건의 경우, 5번 조건과 동일한 형태이므로 어떤 함수 f_6가 존재하여 $l \le f_6(r)$일 때만 6번 조건을 만족한다. $O(N \log N)$ 시간에 $f_6(0), \cdots, f_6(N-1)$을 계산할 수 있다.

1번 조건을 만족하는 l에 대해, r은 $f_5(l) \le r \le f_3(l)$을 만족해야 한다. 2번 조건을 만족하는 r에 대해, l은 $f_4(r) \le l \le f_6(r)$을 만족해야 한다. l을 x좌표, r을 y좌표에 대응시켜 2차원 좌표평면에 나타내면 x축 또는 y축과 평행한 선분 여러 개로 표현되며, 어떤 두 선분의 교점에 위치한 (l, r)만 모든 조건을 만족한다.

따라서, 교점 개수를 세면 부분문제를 해결할 수 있다. 이는 세그먼트 트리와 스위핑을 사용하면 해결할 수 있다. 전체 시간복잡도는 $O((N+Q) \log N)$이다.

부분문제 5

직사각형 영역에 존재하는 교점 개수를 세는 쿼리를 처리하면 된다. x좌표가 증가하는 방향으로 스위핑을 하면서 세그먼트 트리에 (현재 활성화된 x축과 평행한 선분 개수, 현재까지의 답)을 저장하면 lazy propagation을 적용하여 정보를 관리할 수 있다. 쿼리의 답은 쿼리의 x좌표 경계에서 오프라인으로 계산해줄 수 있다. 따라서, 문제를 $O((N+Q) \log N)$에 해결할 수 있다.