"중등부 4번 / 고등부 3번. 수열과 쿼리" 문제 풀이

작성자: 구재현

부분문제 1

쿼리가 주어졌을 때, i를 고정하고 j를 증가시키면서 모든 연속 구간을 열거하고 각각의 합을 계산할 수 있다. 이 중 최댓값을 출력하면 된다. 시간 복잡도는 $O(QN^2)$이다.

부분문제 2

배열 $S_i = \sum_{j=0}^{i-1} A_j$ 를 정의하자. 이는 점화식 $S_{i+1} = S_i + A_i - X$ 를 사용하여 $O(N)$ 에 계산할 수 있다.

쿼리 X가 주어졌을 때, 문제의 정답은 $\max_{0\le j<i\le N}(S_i + iX - S_j - jX) = \max_{1\le i\le N}(S_i + iX - (\min_{0\le j\le i-1} S_j + jX))$ 이다. 배열 $D_i = \min_{j\le i}(S_j + jX)$ 를 정의하자. 이는 점화식 $D_{i+1} = \min(D_i, S_i + iX)$ 를 사용하여 $O(N)$ 에 계산할 수 있다. 문제의 정답은 $\max_{1\le i\le N}(S_i + iX - D_{i-1})$ 이고, $O(N)$ 에 계산할 수 있다. 시간 복잡도는 $O(QN)$이다.

부분문제 3

M_l 을 길이 l의 부분 배열 중 합의 최댓값이라고 정의하자. 배열 M 은 $O(N^2)$ 시간에 계산할 수 있다. 이제 문제의 정답은 $\max_{l=1}^{N}(M_l + lX)$ 이다. 이를 단순히 계산하면 쿼리당 $O(N)$ 시간이 걸리니, 최적화하자.

2차원 평면 상에 N 개의 점 $(x_l, y_l) = (l, M_l)$ 을 그렸을 때, 쿼리 P에 대해 우리는 $x_l \cdot P + y_l$ 의 최댓값을 계산하고 싶다 (입력에는 X로 주어지지만 이제부터 혼동을 막기 위해 P로 표현한다). N 개의 점들로 볼록 껍질 (convex hull) 을 구했을 때, 최댓값이 얻어지는 점은 볼록 껍질의 위쪽에 기울기 $-P$ 의 직선을 그었을 때의 접점임을 알 수 있다. 이러한 접점은, 볼록 껍질의 윗 부분 (upper convex hull) 을 구하고, 각 직선마다 이분 탐색을 사용하여 $O(\log N)$ 시간에 찾을 수 있다. 시간 복잡도는 $O(N^2 + Q \log N)$ 이다.

부분문제 6

배열 M을 빠르게 구하는 것은 어렵기 때문에, 다른 접근이 필요하다.

부분문제 3의 알고리즘과 동치이지만 약간 더 비효율적인, 다음과 같은 알고리즘을 생각하자: 2차원 평면 상에 $N(N+1)/2$ 개의 점 $\{(i-j, S_i - S_j)\}_{0\le j<i\le N}$ 이 주어졌을 때, 쿼리 P 에 대해 $x_l \cdot P + y_l$ 의 최댓값이 문제의 정답이 된다. 모든 $N(N+1)/2$ 개의 점들을 가지고 upper convex hull을 구한 이후, 부분문제 3과 똑같은 이분 탐색을 사용하면 각 쿼리를 $O(\log N)$ 시간에 계산할 수 있다. 하지만, 점의 개수가 너무 많기 때문에, 컨벡스 헐을 구하는 것이 쉽지 않아 보인다.

이 문제를 분할 정복으로 해결하자. $f(L, R)$ 을, 점 집합 $\{(i-j, S_i - S_j)\}_{L\le j<i\le R}$ 의 upper convex hull을 반환하는 함수로 정의하자. $M = (L+R)/2$ 로 정의하면, $f(L, M), f(M+1, R)$ 을 호출하여 $i \le M, j \ge L+1$ 인 경우의 컨벡스 헐의 후보를 모두 얻을 수 있다. 따라서, 점 집합 $\{(i-j, S_i - S_j)\}_{L\le j\le M<i\le R}$ 의 upper convex hull을 계산하고, 재귀적으로 계산한 컨벡스 헐의 후보랑 합쳐준 다음에, 해당 점들로 upper convex hull을 다시 계산한 후 반환하면 된다.

어떠한 점 $(i-j, S_i - S_j)$ 가 upper convex hull에 있다는 것은, 기울기 p 가 존재해서 $(i-j)p + S_i - S_j$ 의 최댓값이 해당 점에서 유일하게 얻어진다는 것과 동치이다. (다시 말해, 모든 다른 점들은 $(i-j)p + S_i - S_j$ 의 값이 해당 점보다 작다). 이를 위해서는 $i \in [M+1, R]$ 에서 $S_i + ip$ 가 최대여야 하고, $j \in [L, M]$ 에서 $S_j + jp$ 가 최소여야 한다. 이는, $M+1 \le i \le R$ 에 있는 점 (i, S_i) 중 upper convex hull에 있는 점들만

고려해도 되고, $L \le j \le M$ 에 있는 점 (j, S_j) 중 lower convex hull에 있는 점들만 고려해도 된다는 것과 동치이다.

$\{(i, S_i)\}_{i=M+1}^{R}$ 에 있는 점들로 upper convex hull을 구하게 되면, 컨벡스 헐 상에서 인접한 점들을 사용하여 각 점이 최댓값을 이루는 p 의 범위를 계산할 수 있다. 같은 방식으로, $\{(j, S_j)\}_{j=L}^{M}$ 에 있는 점들로 lower convex hull을 구하게 되면, 각 점이 최솟값을 이루는 p 의 범위를 계산할 수 있다. 위의 논리로 이 p 의 범위가 겹치는 (i, j) 들의 쌍들만 계산하면 되고, 이는 머지 소트에서 병합하는 과정과 유사한, 아주 짧은 코드로 해결할 수 있다. (이러한 계산을 Minkowski Sum이라고 부른다.)

분할 정복 알고리즘은 컨벡스 헐을 구하는 과정에서 정렬을 필요로 하기 때문에 $T(N) = 2T(N/2) + O(N \log N) = O(N \log^2 N)$ 시간을 필요로 한다. 고로 시간 복잡도는 $O(N \log^2 N + Q \log N)$ 이다.

부분문제 7

위 분할 정복을 $O(N \log N)$ 에 구현하면 된다. 먼저, $\{(i, S_i)\}_{i=M+1}^{R}$ 에 있는 점들로 upper convex hull을 구할 때, 자연스럽게 각 점을 i 가 증가하는 순서대로 본 후 Monotone Chain을 사용하면 $O(N)$ 시간에 upper convex hull을 계산할 수 있다. lower convex hull도 동일하다. 또한, $f(L, R)$ 이 정확히 upper convex hull을 반환하게끔 구현할 필요가 없다: 병합 과정에서 나온 점들을 아무 순서 없이 모은 후 마지막에 한번 upper convex hull을 계산해 주면 된다. 최종적으로, 쿼리 처리 직전에 upper convex hull을 계산해 줄 때, 각 점의 x 좌표가 $[1, N]$ 범위이기 때문에, 단순히 각 x좌표에 대한 y좌표 최댓값을 배열에 기록해 주는 방식으로 정렬을 대체할 수 있다. 시간 복잡도는 $O((N + Q) \log N)$ 이다.

Remark 1. 입력 제한이 커서, 컨벡스 헐 등을 CCW를 사용하여 계산하였을 때 long long 범위를 초과하는 문제가 발생할 수 있다. 이 문제는 네 가지 서로 다른 방법으로 해결할 수 있다 (편리성 증가하는 순으로 나열). 모범 코드는 두 번째 방법으로 구현되었다.

- CCW 대신 기울기를 비교하는 함수를 사용하자. 쿼리 값이 모두 정수이기 때문에, 기울기를 정확한 실수가 아니라 정수 몫을 취한 값으로 부정확하게 비교하여도 문제가 없다.
- CCW의 값을 `double` 자료형으로 계산한 후, 절댓값이 10^{18} 이하일 경우 `long long` 으로 (오버플로우가 나도록) 다시 정확히 계산.
- CCW의 값을 `long double` 자료형으로 계산.
- CCW의 값을 `__int128` 자료형으로 계산: 이번 대회에 사용된 Biko의 채점 환경은 이를 지원한다.

Remark 2. 이 문제에 대해 기존에 알려진 풀이는 Sakai의 2024년 논문으로 [Sak24], $O(N \log^2 N)$의 전처리 시간이 걸린다. 해설의 풀이는 동일한 공간 복잡도와 쿼리 시간을 유지하며, 전처리 시간을 $O(N \log N)$ 으로 개선한다. 또한, 해설의 풀이는 논문의 접근과 비교했을 때 훨씬 단순하다.

References

[Sak24] Yoshifumi Sakai. A data structure for the maximum-sum segment problem with offsets. In 35th Annual Symposium on Combinatorial Pattern Matching (CPM 2024), pages 26–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.