중등부 4, 고등부 3. 화려한 정사각형

풀이 작성자: 구재현

좌표 범위의 최댓값을 X 라 표기한다.

부분문제 1 ($N, X \leq 50$)

격자 위의 정사각형은 왼쪽 아래 모서리의 x 좌표, y 좌표, 그리고 변의 길이로 표현할 수 있다. 고로 모든 가능한 $O(X^3)$ 개의 정사각형을 나열한 후, 이들 안에 들어있는 점들의 서로 다른 색이 K 개 인지를 배열에 마킹하면 된다. 전체 문제가 $O(X^3N)$ 에 해결된다.

부분문제 2 ($K \leq 50, X \leq 2500$)

왼쪽 아래 모서리의 x 좌표, y 좌표를 고정하자. 각각의 색에 대해서, 해당 색의 점을 포함하기 위해 필요한 변의 길이 최솟값을 계산하고, 이의 최댓값을 변의 길이로 취할 것이다. $DP[X][i][j]$ 를 (i, j) 를 왼쪽 모서리로 하는 정사각형이 색 X 의 점을 포함하기 위해 가져야 하는 최소 변의 길이라고 하자. 색 X 의 점이 (i, j) 위치에 있다면 이는 0이고, 아니면 이는 $min(DP[X][i+1][j], DP[X][i+1][j+1], DP[X][i][j+1]) + 1$ 이라는 점화식으로 계산 가능하다. 전체 문제가 $O(KX^2)$ 에 해결된다.

부분문제 3 ($K = 2$)

두 점 (x_1, y_1), (x_2, y_2) 를 포함하는 최소 크기 정사각형의 변의 넓이는 $max(|x_1 - x_2|, |y_1 - y_2|)$ 이다. 이는 두 점의 L_∞ 거리 (체비셰프 거리) 와 동일하다. 색깔 1의 점과 색깔 2의 점을 각각 하나 골라서 위 값을 최소화해야 한다. 즉, 체비셰프 거리 상 가장 가까운 두 점 쌍을 찾는 것이다.

평면을 45도 회전하면, 위 거리는 $|x_1 - x_2| + |y_1 - y_2|$ 형태의 L_1 거리 (맨하탄 거리)로 변환된다. 일반성을 잃지 않고 $x_1 \leq x_2, y_1 \leq y_2$ 라고 하자 (4방향으로 모두 회전시키면서 해 보면 된다). 우리는 고정된 (x_1, y_1) 에 대해서 $x_1 \leq x_2, y_1 \leq y_2$ 를 만족하며 $x_2 + y_2$ 를 최소화하는 점을 찾아야 한다. 이는 x 좌표에 대해 스위핑을 하고, y 좌표에 대해서 $x + y$ 의 최솟값을 저장하는 세그먼트 트리를 관리하면 된다. 시간 복잡도는 $O(N \log N)$ 이다.

부분문제 4 ($N \leq 50$)

부분문제 4에서 6까지는 정사각형을 찾는다고 생각하지 않고, 각 변의 길이가 w, h 일 때 $\max(w, h)$ 를 최소화하는 직사각형을 찾는다고 생각한다. 직사각형은 x 축에 수직한 두 변 $x = x_s, x = x_e$, y 축에 수직한 두 변 $y = y_s, y = y_e$, 이렇게 총 4개의 정수로 표현할 수 있다. 직사각형의 4개의 변 중 하나에 점이 닿지 않는다면, 점이 닿을 때까지 해당 변을 안쪽으로 당기면서 $\max(w, h)$ 를 유지하거나 줄일 수 있다. 이는 우리가 고려해야 할 (x_s, x_e, y_s, y_e) 쌍의 후보가 $O(N^4)$ 개라는 것을 뜻한다: x_s, x_e 는 입력으로 주어진 $O(N)$ 개의 x 좌표 중 하나고, y 축에 대해서도 동일하기 때문이다. 고로 모든 가능한 $O(N^4)$ 개의 정사각형을 나열한 후, 이들 안에 들어있는 점들의 서로 다른 색이 K 개 인지를 배열에 마킹하는 식으로 확인하면 된다. 시간 복잡도는 $O(N^5)$ 이다.

부분문제 5 ($N \leq 150$)

x_s, x_e 를 고정시키고, $x_s \leq x \leq x_e$ 를 만족하지 않는 점들을 모두 무시해 주자. 이제 y 좌표만이 중요하니, 1차원에서 이 문제를 해결하면 된다. 점을 y 좌표 순으로 정렬한 후, y_s 를 고정해 주고 y_e를 늘려주면서, 서로 다른 색이 K 개가 되면 답을 갱신하자. 1차원 문제가 $O(N^2)$ 에 풀리니 전체 문제는 $O(N^4)$ 에 풀린다.

부분문제 6 ($N \leq 600$)

부분문제 5와 동일하게 1차원 문제를 해결한다. y_s 를 고정해 주고 y_e 를 늘려주면서, 서로 다른 색이 K 개인 것이 확인되면 처음부터 시작하는 것이 아니라 y_s 를 한 칸 늘려준 후 (가장 왼쪽 점을 제거한 후) 다시 y_e 를 서로 다른 색이 K 개일 때까지 늘려주는 것을 반복한다. 이러한 방식을 Sliding Window 혹은 Two Pointers라고 하며, 점이 한 번 구간에 들어가고 한 번 나가니 $O(N)$ 에 1차원 문제가 해결된다. 정렬을 해야 한다고 생각할 수 있겠지만, 맨 처음에 점을 y 좌표 기준으로 한번만 정렬해 주면 굳이 매번 정렬할 필요가 없다. 전체 문제는 $O(N^3)$ 에 풀린다.

부분문제 7 ($N \leq 2500$)

길이 R 이하의 화려한 정사각형이 존재하는가? 라는 결정 문제로 문제를 변형해서 해결한다. 이 문제를 해결할 수 있다면, 최소의 R을 이분 탐색으로 찾을 수 있다. R 을 고정할 경우, x_s, x_e 로 가능한 후보는 $O(N)$ 개가 된다. 왼쪽 변에 점이 닿는 경우가 N 개이고, 오른쪽 변에 점이 닿는 경우가 N 개이니 $2N$ 개만 시도해 보면 되기 때문이다: 둘 다 닿지 않는 경우는 앞과 비슷하게 왼쪽이나 오른쪽 점에 부딪힐 때까지 구간을 움직여 주면 된다. 이제 부분문제 6처럼 1차원 문제를 $O(N)$ 에 푼 후 그 답이 R 이하인지를 $2N$ 번 확인해 주면, 결정 문제가 $O(N^2)$ 에 풀리고, 전체 문제는 $O(N^2 \log X)$ 에 풀린다.

부분문제 8

부분문제 7과 동일한 결정 문제를 Plane sweeping을 사용하여 효율적으로 해결한다. 길이 R 의 x 좌표 구간 $[x_c, x_c + R]$를 왼쪽에서 오른쪽으로 움직이자. x_c 가 증가하는 과정에서 N 개의 점들은 $x_c = x_i - R$ 시점에 구간에 삽입되고, $x_c = x_i + 1$ 시점에 구간에서 제거된다. 문제를 해결하기 위해서는, 점들이 추가되고 제거될 때, 모든 K 개의 색의 점을 포함하는 길이 R 의 구간이 있는지를 빠르게 판별해야 한다.

$distinct[y]$ 를 $[y - R, y]$ 구간을 덮었을 때 얻을 수 있는 서로 다른 색의 개수라고 정의하자. $distinct[y] = K$ 인 x가 존재한다면, 답이 존재한다고 판별할 수 있다. 점 하나가 추가되고 제거될 때 $distinct[y]$ 가 어떠한 식으로 변경되는지 살펴보자.

각 색에 대해서, 해당 색의 점들의 y 좌표들을 `std::multiset` 과 같은 자료 구조로 관리하자. y 좌표가 y_j 인 점을 추가할 때, 기본적으로 이 점이 새롭게 $distinct[i]$ 를 1 증가시킬 구간은 $[y_j, y_j + R]$ 이 된다. 하지만, 이미 y_j 의 직전에 있는 점이 합집합을 덮고 있어서 더하지 않아도 되는 구간, 직후에 있는 점에 가로막혀서 더하지 않아도 되는 구간들이 존재한다. 이 구간은 y_j 의 양 옆으로 인접한 점들의 좌표만 알면 케이스 처리로 구할 수 있다. 고로, `std::multiset`의 `lower_bound` 함수를 사용하면 새롭게 기여하게 되는 구간을 $O(\log N)$ 에 계산할 수 있다. 삭제에서도 역시, $distinct[i]$ 가 1 감소하는 구간을 아주 유사한 방식으로 계산할 수 있다.

이제 문제는 구간에 대해서 특정 수를 더하고, 전체 배열에 값이 K 인 원소가 있는지를 체크하는 구간 쿼리 문제가 된다. 배열의 특정 원소의 값은 항상 K 이하이므로, 세그먼트 트리에 Lazy propagation을 사용하여 해결할 수 있다. 결정 문제가 $O(N \log X)$ 에 해결되니, 전체 문제도 $O(N \log^2 X)$ 에 해결되어 만점을 받을 수 있다.