"과일 게임 (fruitgame)" 문제 풀이

작성자: 이온조

부분문제 1

제한이 매우 작기 때문에 쿼리마다 수행할 수 있는 모든 방법을 백트래킹을 통해 시도해보면 $O(QN!)$ 시간에 문제를 해결할 수 있다.

부분문제 2

각 과일에 대해서 그 과일을 왼쪽 끝으로 해서 만들 수 있는 가장 큰 과일을 계산할 수 있다. 이는 스택을 사용하여 과일을 합치는 것이 가능할 때마다 합쳐 주는 그리디한 알고리즘을 사용하면 된다. 쿼리마다 모든 과일에 대해서 이를 계산해보면 $O(QN^2)$ 시간에 문제를 해결할 수 있다.

부분문제 3

모든 과일의 번호가 2 이하인 부분문제이다. 번호가 1인 과일들이 연속으로 짝수 개만큼 위치해있는 경우에는 두 개씩 짝지어서 모두 번호가 2인 과일로 만들어주어도 된다. 하지만 번호가 1인 과일들이 연속으로 홀수 개만큼 위치해있는 경우에는 그 양옆의 과일들이 합쳐질 수 없다는 사실을 관찰할 수 있다. 그러므로 번호가 1인 과일들이 연속으로 홀수 개만큼 있는 위치들을 기준으로 과일들을 최대한 합쳐 보고, 가장 큰 과일의 크기를 구하면 된다는 사실을 알 수 있다. 이는 쿼리마다 선형 시간에 계산 가능하므로 $O(NQ)$ 시간에 문제를 해결할 수 있다.

부분문제 4

동적 계획법을 사용해보자. $D[i][j]$:= (i번째 과일을 왼쪽 끝으로 해서 번호가 j인 과일을 만들 때 그 과일의 오른쪽 끝 위치) 라고 정의하자. 이 값들을 쿼리마다 $O(|A_i|N)$ 시간에 계산할 수 있고, $O(|A_i|NQ)$ 시간에 문제를 해결할 수 있다.

부분문제 5

생각해보면 부분문제 3의 풀이에서 중요한 정보는 1번 과일이 홀수 개만큼 연속해 있을 때 그 연속한 묶음의 위치라는 것을 알 수 있다. 이 위치들을 알고 있으면 그 사이에 있는 과일들을 모두 합칠 수 있기 때문에 가장 큰 과일의 크기를 계산할 수 있다. 즉, 과일이 업데이트될 때마다 연속한 과일 묶음에 대한 정보를 관리해주면 답을 빠른 시간 안에 계산할 수 있다. 이는 `std::set` 등을 사용하면 빠르게 관리할 수 있으므로 $O((N+Q)\log N)$ 시간에 문제를 해결할 수 있다.

부분문제 6

부분문제 4에서 사용했던 동적 계획법을 마찬가지로 먼저 계산해두자. DP 정보를 이용해서 구간 쿼리가 주어지면 그 구간에서 만들 수 있는 가장 큰 과일의 번호를 빠르게 구하면 좋을 것이다. $[l, r]$ 구간 쿼리가 주어졌을 때 번호가 x인 과일을 만들 수 있는지 판별하려면 $l \le i, D[i][x] \le r$인 DP 값이 존재하는지 확인하면 된다. 이 때 유효한 DP값들에 대해서 단조성이 성립한다는 사실을 관찰할 수 있다. 그러므로 이분 탐색을 이용하여 $l \le i$이고 $D[i][x]$가 유효한 i 중 가장 작은 i를 찾아서, $D[i][x] \le r$인지 판별하면 충분하다. 쿼리마다 모든 과일 번호 x에 대해서 이를 확인하면 만들 수 있는 가장 큰 과일의 번호를 알 수 있고, 시간복잡도는 $O((N + Q\log N)|A_i|)$이다.

부분문제 7

번호가 x인 과일이 y개 연속해서 위치해 있다면 이를 (x, y)로 표현하자. 이제 과일들을 $(x_1, y_1), (x_2, y_2), \cdots, (x_k, y_k)$ 꼴로 표현할 수 있다. 이 때 $x_{i-1} > x_i < x_{i+1}$인 경우를 생각해보자. 만약 y_i가 짝수라면, 부분문제 3의 풀이에서처럼 두 개씩 짝지어서 합쳐주어도 된다는 사실을 알 수 있다. 즉, $(x_i, y_i) \to (x_i + 1, \frac{y_i}{2})$ 변환이 가능하다. 반대로 y_i가 홀수라면 남는 과일이 하나 생기기 때문에 (x_i, y_i) 묶음을 기준으로 양쪽의 과일들은 합쳐질 수 없다. 그러므로 $(x_i, y_i) \to (x_i, y_i - 1), (\infty, 1), (x_i, y_i - 1)$과 같이 변환이 가능하다. 이러한 변환이 불가능할때까지 계속 변환을 반복하면, $(\infty, 1)$ 사이에 바이토닉 수열 형태로 과일들이 있는 형태가 될 것이다. 바이토닉한 형태이기 때문에 그리디하게 작은 번호의 과일부터 차례대로 합쳐 주면 가장 큰 과일의 크기를 계산할 수 있음을 알 수 있다. 또한 $(\infty, 1)$을 기준으로 양 옆의 과일들은 분리되어 있기 때문에 전체 수열에서 만들 수 있는 가장 큰 과일의 번호를 알 수 있다.

이제 이 바이토닉 수열의 형태를 세그먼트 트리를 사용하여 관리해보자. 세그먼트 트리의 각 노드에 수열의 형태 정보와 그 구간 안에서 만들 수 있는 가장 큰 과일의 번호를 저장하자. 만약 구간 내에 $(\infty, 1)$이 두 개 이상 존재한다면 그 사이에 존재하는 과일들은 독립적으로 계산해주고, 구간의 최대 답을 갱신해줄 수 있다. 그러므로 양쪽 끝의 두 바이토닉 수열 정보를 관리하면 되므로, 노드마다 $O(|A_i|)$ 크기의 정보를 스택 등의 자료구조를 사용해 관리하면 된다. 두 노드를 합칠 때는 위의 변환을 사용하고, 만약 그 과정에서 $(\infty, 1)$이 두 개 이상 생긴다면 마찬가지로 그 사이의 수열에서 가장 큰 과일을 답에 갱신한 뒤, 양쪽 끝의 두 수열만 계속 관리하면 된다. 시간복잡도는 $O((N + Q)|A_i| \log N)$이다.