Lottery (Solution)

Subtasks 1 and 2: $Q \leq 100, R_j - L_j + 1 \leq 100$

For simplicity, we assume $L_j = 0$ and $R_j = n - 1$. We also focus only on the selections that result in obtaining a prize.

We consider the decision problem of whether it is possible to achieve a total of K prizes. First, it is necessary that $X_i + Y_i \geq K$ holds for every i.

Under this condition, the problem can be restated as follows:

- You are given integers a_i and b_i such that $0 \leq a_i \leq b_i \leq K$.
- There is a sequence of integers $x = (x_0, \ldots, x_{n-1})$, initially all set to $x_i = 0$. Determine whether it is possible to perform an operation of *adding* 1 *to* $n/2$ *entries* K times so that the final x satisfies $a_i \leq x_i \leq b_i$ for all i.

The necessary and sufficient condition for this problem is: $\sum a_i \leq \frac{nK}{2} \leq \sum b_i$. It is obvious that this condition is necessary. To show sufficiency, we can construct a sequence x that satisfies $a_i \leq x_i \leq b_i$ and $\sum x_i = nK/2$. We now show that this x can be obtained by performing the operations.

It is sufficient to show that it is possible to reduce x to $(0, 0, \ldots, 0)$ by performing the operation of *subtracting* 1 *from* $n/2$ *entries*, K times. This can be done using a greedy strategy: in each step, subtract 1 from the $n/2$ entries with the largest x_i values. It can be proven by induction that this greedy strategy always succeeds.

Since the decision problem can be solved in $O(n)$ time, binary search can be used to find the optimal value in $O(n \log\{\max_i(X_i + Y_i)\})$ time.

The total time complexity for all queries is $O(Q \cdot \max_j(R_j - L_j + 1) \cdot \log \max_i(X_i + Y_i))$.

In Subtask 1, the values of X_i and Y_i are small, so it is also possible to use simulation-based greedy solutions for part of the decision process.

Subtask 3: $L_j \leq L_{j+1}, R_j \leq R_{j+1}$

The conditions explained in Subtask 1 can be restated in terms of X_i and Y_i as follows:

- $X_i + Y_i \geq K$.
- $\sum_i \min(X_i, K) \geq nK/2$.

- $\sum_i \min(Y_i, K) \geq nK/2$.

Note that these correspond to the requirements that *the total number of balls in the bag i is sufficient*, *the number of red balls is sufficient*, and *the number of blue balls is sufficient*, respectively, so the necessity of each condition is clear.

The goal is to find the maximum value of K that satisfies all three conditions simultaneously. To do this, it suffices to find the maximum K that satisfies each condition individually.

The first condition can be handled with a simple range minimum query. Since the other two conditions have the same form, solving the problem for X_i suffices.

In Subtask 3, the multiset $\{X_{L_j}, \ldots, X_{R_j}\}$ changes at most $O(N)$ times in total across all queries. Therefore, we can use a data structure that supports:

- Insertion and deletion of elements in a multiset.
- Computing the count and sum of elements less than K.

Since all possible values to be inserted are given in advance, coordinate compression can be applied. This allows us to use data structures such as Fenwick Tree.

For example, the problem can be solved in $O(Q \log N \log \max(X_i + Y_i))$ time.

Subtasks 4, 5, and 6

Let us extend the solution described in Subtask 3 to handle arbitrary range queries $[L_j, R_j)$.

This can be done using a Wavelet Matrix (alternatively, a persistent segment tree can achieve a similar solution).

Wavelet Matrix is a data structure that, when a Binary Trie is built over the elements of array X, allows efficient retrieval of all elements in a given range that fall into a particular node of the Trie. When combined with prefix sum tables, it allows us to compute the total sum of these elements.

The count and sum of elements in $X_{L_j}, \ldots, X_{R_j}$ that are less than K correspond to sums over the nodes representing the range $[0, K)$. These can be computed in $O(\log \max X_i)$ time, or $O(\log N)$ time after coordinate compression.

This allows us to solve the problem in $O(Q \log^2 N)$ time. However, obtaining full marks may be challenging.

Instead of separating evaluation and binary search, one can perform both simultaneously while traversing the Wavelet Matrix recursively. This reduces the time complexity to $O(Q \log N)$ and allows the problem to be solved within the time limit.