Examination 2 (Solution)

Author: Masaki Nishimoto (maspy)

Subtask 1 ($S$ does not contain & or |)

In this subtask, the boolean value the IOI function maps to does not depend on the order of operations. The answer can be determined using the following two pieces of information:

- The number of $!$.
- The multiset of all integers $a$ that are the subject of Rule 1.

The multiset mentioned can be easily obtained by parsing only the parts enclosed in $[ ]$ brackets. This subtask can be solved in $O(N + Q \log N)$ time complexity.

Subtask 2 ($Q = 1$)

This subtask requires basic handling of parsing. The detailed implementation of the parsing method is omitted here.

Given that the problem statement does not provide a BNF notation, there might be some confusion regarding the correct implementation of rule precedence. If you are unable to implement it correctly, please refer to the following BNF notation:

- $<\text{FUNC}_1> ::= \[' <\text{number}> ' \]'$
- $<\text{FUNC}_2> ::= <\text{FUNC}_1> | (\text{'} <\text{IOI}\_\text{FUNCTION}> \text{'})$
- $<\text{FUNC}_3> ::= <\text{FUNC}_2> | \text{'}!\text{'} <\text{FUNC}_3>$
- $<\text{FUNC}_4> ::= <\text{FUNC}_3> | <\text{FUNC}_4> \text{'&'} <\text{FUNC}_3>$
- $<\text{FUNC}_5> ::= <\text{FUNC}_4> | <\text{FUNC}_5> \text{'\^'} <\text{FUNC}_4>$
- $<\text{FUNC}_6> ::= <\text{FUNC}_5> | <\text{FUNC}_6> \text{'\|'} <\text{FUNC}_5>$
- $<\text{IOI}\_\text{FUNCTION}> ::= <\text{FUNC}_6>$

This subtask can be solved in $O(N)$ time complexity.
Subtask 3 \((N \leq 10,000)\)

The boolean value \(X\) maps to is determined solely by its relationship with the integers subject to Rule 1. Therefore, as long as we calculate the boolean values for the \(O(N)\) integers, we can determine the answer for each query in \(O(\log N)\) time.

If we perform parsing for each of the \(O(N)\) integers, this subtask can be solved in \(O(N^2 + Q \log N)\) time complexity.

Additionally, if we can represent the IOI function as a tree (as will be discussed later), the boolean values can be determined in \(O(N)\) time. Thus, in this subtask, a solution that spends \(O(N^2)\) time on parsing is acceptable. For instance, one approach could involve spending \(O(N)\) time each time to identify the application point of the highest precedence rule.

Subtask 4 \((S\) does not contain \(!\) or \(^\)\)

In the following subtasks, it is assumed that the syntax of the given IOI function has been correctly parsed. It is convenient to represent the result of the syntax parsing in a tree, as shown below.

\[
S = (![2][3])\&![4]
\]

Under the constraints of Subtask 4, the IOI function exhibits the following monotonicity: there exists a value \(t\) such that integers less than \(t\) map to False, and integers greater than or equal to \(t\) maps to True.

By computing this \(t\) progressively from the leaves of the tree, this subtask can be solved in \(O(N + Q)\) time. Alternatively, we can consider a solution that leverages binary search, focusing on this monotonicity.
Subtask 5 (One child of vertices corresponding to & or | is a leaf)

Consider computing the IOI function’s information progressively from the leaves of the tree. Let $n$ denote the number of leaves in a subtree. The mapping of the IOI function is constant within each interval defined by $O(n)$ boundary integers.

Thus, we can maintain the IOI function’s information using the following:

- The boolean value $x_0$ that $X = 0$ maps to.
- The set change consisting of all integers $x$ at which the boolean value changes, i.e., the boundaries where $X = x - 1$ and $X = x$ map to different boolean values.

Rule 3 involves flipping $x_0$. For Rules 4 and 6, if one child of a node is not a leaf, we need to perform appropriate element deletions from the set change and handle the boundary conditions. Rule 5 involves performing the XOR operation between two sets.

Among these, only Rule 5 may present computational challenges with a straightforward implementation. However, by implementing a method where we move elements one by one from the smaller set to the larger set, known as the weighted union heuristic, the total number of element transfers across the entire tree becomes $O(\log N)$. Therefore, we can perform the computations for the entire tree in $O(N \log^2 N)$ time.

With the above approach, this subtask can be solved in $O(N \log^2 N + Q \log N)$ time complexity.

Subtask 6 and 7

To optimize the computation for not only Rule 5 but also Rules 4 and 6, we can apply the weighted union heuristic. By appropriately maintaining the IOI function’s information, we aim to achieve the following: flipping the boolean values, setting them to True, or setting them to False over a specified interval.

By enumerating all the relevant values at a particular tree vertex and applying coordinate compression, we can perform the necessary operations using a (dual) segment tree in $O(\log n)$ time. We ensure that for vertices inheriting the same data structure through the weighted union heuristic, the same set of values is computed. This is analogous to performing computations per heavy path as in the Heavy-Light Decomposition (HLD). For more details, you can search for "DSU on Tree”.

Alternatively, we can use a data structure known as a dynamic segment tree.

By implementing the above approach correctly, this problem can be solved in $O(N \log^2 N + Q \log N)$ time complexity.
Additional Information

Apart from the methods discussed above, another possible approach involves incrementing $X$ and sequentially determining the boolean values that the IOI function maps to. In this case, it suffices to quickly compute some results from tree DP each time the boolean values of the leaves are updated.

A data structure often used for such dynamic tree DP calculations is the Static Top Tree, which can also be applied to this problem. Additionally, given that the tree in this problem is a binary tree, HLD combined with segment trees can effectively handle dynamic tree DP updates.