Crossing (Solution)

We associate `J`, `O`, `I` with 0, 1, 2, respectively, and consider a gene sequence as an N-dimensional vector over the residue field F_3. Let $v \circ u$ be the gene sequence obtained from v, u by crossing. Note that $v \circ u = 2(v + u)$ holds. By induction on the number of crossing operations, it can be shown that a gene sequence which can be obtained from S_A, S_B, S_C is written as a linear combination of the form $aS_A + bS_B + cS_C$ for some $a, b, c \in F_3$ satisfying $(a + b + c) \bmod 3 = 1$. This is a necessary condition. There are 9 possible choices of tuples (a, b, c), and all of them are obtained as follows. Hence this is also a sufficient condition.

- If $(a, b, c) = (1, 0, 0)$, it is the sequence S_A itself.
- If $(a, b, c) = (0, 1, 0)$, it is the sequence S_B itself.
- If $(a, b, c) = (0, 0, 1)$, it is the sequence S_C itself.
- If $(a, b, c) = (2, 2, 0)$, it is the sequence $S_A \circ S_B$.
- If $(a, b, c) = (0, 2, 2)$, it is the sequence $S_B \circ S_C$.
- If $(a, b, c) = (2, 0, 2)$, it is the sequence $S_C \circ S_A$.
- If $(a, b, c) = (2, 1, 1)$, it is the sequence $S_A \circ (S_B \circ S_C)$.
- If $(a, b, c) = (1, 2, 1)$, it is the sequence $S_B \circ (S_C \circ S_A)$.
- If $(a, b, c) = (1, 1, 2)$, it is the sequence $S_C \circ (S_A \circ S_B)$.

Therefore, calculating the above 9 gene sequences in advance, we need to compare them with each sequence T_j. Under the constraints of Subtask 3, it can be done in time by a simple comparison of strings.

In the following, we fix a gene sequence S. We want to decide whether it coincides with each sequence T_j or not. This immediately gives a solution of Subtask 2. Combining it with the above consideration, we obtain a solution of Subtask 4.

For simplicity, we assume the integer N is a power of 2. We put the additional data

$$aux[l, r) := \begin{cases} \texttt{J} & (S[k] = \texttt{J} \text{ for all } l \le k < r) \\ \texttt{O} & (S[k] = \texttt{O} \text{ for all } l \le k < r) \\ \texttt{I} & (S[k] = \texttt{I} \text{ for all } l \le k < r) \\ -1 & (\text{otherwise}) \end{cases}$$

on each node of a segment tree. Defining lazy propagation functions appropriately, we can construct a lazy propagation segment tree $(seg, lazy)$ with

$$seg[l, r) := \begin{cases} 1 & \left(S[l, r) = T_j[l, r)\right) \\ 0 & (\text{otherwise}) \end{cases}$$

for every evaluated node $seg[l, r)$. The time complexity of this algorithm is $O(\log N)$ per a query. It is sufficiently fast to solve this task.