Selling RNA Strands (Solution)

Subtask 1 [10 points]

$N, M, |S_i|, |P_j|, |Q_j|$ are small. We can solve this subtask by checking all RNA sequences for each order.

Subtask 2 [25 points]

Let's consider a simpler version of this problem:

> There are N RNA sequences $S_1, \ldots, S_N$ and M orders $P_1, \ldots, P_M$. For each j ($1 \leq j \leq M$), compute the number of i such that the first $|P_j|$ characters of S_i are P_j.

We can store all of $S_1, \ldots, S_N$ to a trie. Now we identify a string X and the node of the trie which corresponds to X. For any two strings X, Y, the following are equivalent:

- X is a prefix of Y, that is, the first $|X|$ characters of Y are X.
- Node X is an ancestor of node Y.

Let $C(S)$ be the number of occurences of S in $S_1, S_2, \ldots, S_N$. Then the problem is computing $\sum_{X:\text{predecessor of } P_j} C(X)$ for each j.

This can be done by precalculation with DFS, but here we use a different approach. Using the Euler-Tour technique, checking whether node x is an ancestor of node y can be reduced to checking whether a point (corresponding to y) is included in a range (corresponding to x). Now we get an algorithm for this problem:

- Store $S_1, \ldots, S_N, P_1, \ldots, P_M$ to a trie.
- Using the Euler-Tour technique, compute the range and the point for nodes in the trie. Note that we need not to compute the values for nodes which don't correspond to any of $S_1, \ldots, S_N, P_1, \ldots, P_M$.
- For each j, count the number of i such that the point corresponding to node S_i is included in the range corresponding to node P_j.

Let's go back to the original problem. Now we have the constraint by Q_j as well as by P_j. We saw that handling P_j can be done by a trie and the Euler-Tour technique. It can be easily checked that the same approach can be used for handling Q_j if we consider ${S_1}^R, \ldots, {S_N}^R, {Q_1}^R, \ldots, {Q_M}^R$ (where S^R is the reverse string of S).

Finally, we get an algorithm as follows:

- Store $S_1, \ldots, S_N, P_1, \ldots, P_M$ to a trie.

- Using the Euler-Tour technique, compute the range and the point for nodes in the trie.
- Do the same for ${S_1}^R, \ldots, {S_N}^R, {Q_1}^R, \ldots, {Q_M}^R$.
- For each j, compute the answer by checking two ranges.

Subtask 3 [25 points]

Let $\Sigma_S, \Sigma_P, \Sigma_Q$ be $|S_1| + \ldots + |S_N|$, $|P_1| + \ldots + |P_M|$, $|Q_1| + \ldots + |Q_M|$, respectively.

We saw that this problem becomes relatively easy if we can ignore P or Q. Now consider the following algorithm:

- For each j, extract only RNA sequences whose last $|Q_j|$ characters are Q_j. Then compute the number of RNA sequences in the extracted ones whose first $|P_j|$ characters are P_j.

The later part of the algorithm can be done by using a trie. The former part also can be done with a trie. This algorithm is not so efficient because it may be possible that all of the RNA sequences are extracted and thus stored to a trie for each j. In this case, it works in $O(M\Sigma_S)$. But we can add a (seemingly small) optimization:

- Extract RNA sequences only once for the same Q.

In fact, it significantly reduces the complexity to $O\left(\Sigma_S \sqrt{\Sigma_Q} + \Sigma_P + \Sigma_Q\right)$.

Here is a proof. For each i, S_i is extracted only if Q_j is a suffix of S_i (that is, the last $|Q_j|$ characters of S_i are Q_j). Let L be the set of such Q_j's. Each string of L has different length, so $\sum_{X \in L} |X| \geq 1 + \ldots + \#L = \frac{(\#L+1)\#L}{2}$. Also $\sum_{X \in L} |X| \leq \Sigma_Q$ holds, so we have $\frac{(\#L+1)\#L}{2} \leq \Sigma_Q$. Thus $\#L = O\left(\sqrt{\Sigma_Q}\right)$. It means that each S_i is added to a trie $O\left(\sqrt{\Sigma_Q}\right)$ times. Adding S_i to a trie once can be done in $O(|S_i|)$. So adding strings to tries can be completed overall in $O\left(\Sigma_S \sqrt{\Sigma_Q}\right)$. Now we have tries, so computing the answer can be done by accessing node P_j of a trie. This can be done in $O(\Sigma_P)$. Also, checking all of $Q_1, \ldots, Q_M$ itself takes $O(\Sigma_Q)$ time.

After all, it was proved that this algorithm works in $O\left(\Sigma_S \sqrt{\Sigma_Q} + \Sigma_P + \Sigma_Q\right)$.

Subtask 4 [40 points]

The approach to the solution for Subtask 2 is reducing the problem to the following:

> There are N points X_i, Y_i and M rectangles $[A_j, B_j] \times [C_j, D_j]$ (whose axes are parallel to the X or Y axis) on a 2D plane. For each rectangle, compute the number of points which are included in it.

If we ignore irrelevant nodes of tries, the coordinates of the points and the rectangles will be integers between 0 and $N + M$.

Let $K(A, B, C, D)$ be the number of i such that $A \leq X_i \leq B$ and $C \leq Y_i \leq D$. As the coordinates of the points are not less than 0, we have $K(A, B, C, D) = K(A, B, -1, D) - K(A, B, -1, C - 1)$. So we just need to compute $K(A_k, B_k, -1, D_k)$ for some queries A_k, B_k, D_k efficiently. This can be done by the following algorithm:

- Sort the points and the queries by Y_i (for points) or D_k (for queries). If a point and a query have the same value, make the point appear earlier in the sequence.
- Initialize an array $V_0, \ldots, V_{N+M}$ with 0.
- Scan the sorted sequence from the beginning and do the following:
  - For a point (X, Y), set $V_X \leftarrow V_X + 1$.
  - For a query $(A, B, -1, D)$, answer $V_A + V_{A+1} + \ldots + V_B$.

In this algorithm, we need to do array operations efficiently. This can be done with a Binary Indexed Tree.