Solution for task 'dna'

Let m denote the length of a query, i.e. $m = y - x + 1$

A solution exists for a given query if and only if the multiset of characters in $a[x..y]$ is the same as the multiset of characters in $b[x..y]$.

1 Subtask 1

Constraint: $y - x \leq 2$

There are at most 3 characters in the query, therefore at most two swaps will suffice.

Assuming a solution exists, check if there is a one swap solution, if yes, the distance is 1, else the distance is 2.

2 Subtask 2

For two characters, we can count the number of 'A's that should be 'T's and 'T's that should be 'A's.

Each swap can fix a mismatch pair, therefore a solution is to count the number of mismatch pairs.

This can be done in $O(m)$ time per query as the number of queries is small.

3 Subtask 3

The is similar to subtask 2 except there are more queries.

To speed up the counting of mismatches for each query, we can pre-compute the number of mismatched pairs for all possible prefixes in $O(n)$ time.

Using the prefix table, we can determine the number of mismatched pairs for any substring in constant time.

The total time complexity will be $O(n + q)$

4 Subtask 4

For three characters, we can first look for any swap that will result in fixing two characters. Looking for and applying these swaps takes $O(m^2)$ time.

After that, the remaining mismatches will take 2 swaps to fix three characters. Looking for and applying these swaps can also be done in $O(m^2)$ time.

The total time complexity will be $O(qm^2)$

5 Full solution

We need to pre-compute mismatches for each prefix as we did in subtask 3. This can be done by recording the number of 'P's that should be 'Q's for every P and Q and every prefix using n 3×3 mismatch matrices.

Given a query, we can compute the mismatch matrix for that query in constant time. This will allow us to count the number of swaps that fixes two characaters in constant time and subtract it from the mismatch matrix.

The remaining mismatches requires 2 swaps to fix three characters and we can also count these in constant time.

The total time complexity will be $O(n + q)$

6 Solving for larger alphabet sizes

We can view the mismatch matrix as a multigraph, where the nodes are the characters in the alphabet and there are edges from u to v for each position i where $a[i] = u$ and $b[i] = v$

It is optimal to proceed in the following greedy fashion for graphs with 5 or fewer nodes.

1. find cycles of length 2 and removing them with 1 swap
2. find cycles of length 3 and removing them with 2 swaps
3. find cycles of length 4 and removing them with 3 swaps
4. find cycles of length 5 and removing them with 4 swaps

However, this approach fails in general when the alphabet size 6 or larger.