Coding Competitions Farewell Rounds - Round A

Analysis: Rainbow Sort

For simplicity, let us say the positive integers to be written on the cards are represented with an array *A*.

Test Set 1

In this test set, since $\mathbf{N}$ is very small, we can try all possible orderings of unique values of colours and use their indices in these orderings to build candidate *A* arrays and find one non-decreasing *A* from them.

Find all the unique values of $\mathbf{S}$ using a hash set in $O(\mathbf{N})$ time. Let M be the array holding the unique values.
Find all the permutations of M. For each permutation P of M, follow the below logic to create *A*.

Initialize a hash map X. For each P_i, assign $X[P_i] = i$. Iterate through $\mathbf{S}$ and for each $\mathbf{S_i}$, assign *A*$[i] = X[\mathbf{S_i}]$. Once *A* is constructed, we can easily check whether *A* is non-decreasing by iterating through the array.

Return `IMPOSSIBLE` if none of the *A* arrays is non-decreasing, else return one of them as the answer. For each permutation, we take $O(\mathbf{N})$ time to check whether the array *A* is non-decreasing. Since there are $O(\mathbf{N}!)$ such permutations, the overall complexity of this solution is $O(\mathbf{N}! \times \mathbf{N})$.

Test Set 2

The previous algorithm would be too slow for Test Set 2, so we must find a more efficient solution.

For cards to have a possible valid *A* array, we can conjecture that cards with the same value should be consecutive to each other, i.e., not separated by another card of a different value.
By way of contradiction, let us suppose our above conjecture is incorrect i.e., at least one card is present with a different number between two cards with same value.
Let $\mathbf{S_i}$, $\mathbf{S}_j$, $\mathbf{S}_k$ be three values in S such that $i < j < k$ and $\mathbf{S_i} = \mathbf{S}_k \neq \mathbf{S}_j$. Hence *A*$[i]$ =*A*$[k] \neq$ *A*$[j]$.
Since $\mathbf{S_i} \neq \mathbf{S}_j$, *A*$[i]$ <*A*$[j]$.
Similarly, since $\mathbf{S}_j \neq \mathbf{S}_k$, *A*$[j]$ <*A*$[k]$.
Hence, *A*$[i]$ <*A*$[k]$ which is contradictory.
So cards with the same values should be consecutive to each other.

It follows from the above observation that we only need to consider unique values of $\mathbf{S}$. For simplicity, let K be the array after removing consecutive duplicates while keeping one instance of one unique value in $\mathbf{S}$ and also keeping the order of unique elements same. Let L be the array such that L_i is the index position of $\mathbf{S_i}$ in K. We can easily see that L is a non-decreasing array and the mapping between $\mathbf{S}$ and L is one to one.

Initialize an *A* array and an empty hash set to store elements of $\mathbf{S}$ seen while iterating. Iterate through $\mathbf{S}$ from left to right and for each $\mathbf{S_i}$, find if the element appeared before in the list i.e., if the element is present in the hash set.
If the element is not present, then insert the element into the hash set and *A* array and continue

iterating.
If the element is present in the hash set and it is equal to the previous element in $\mathbf{S}$ then continue iterating else return the answer as `IMPOSSIBLE`.

If we are able to successfully iterate all the way through $\mathbf{S}$, then return the *A* array built while iterating. Since we are iterating through $\mathbf{S}$ only once and using the hash set, the overall time complexity of this solution is $O(\mathbf{N})$.