Kick Start 2021 - Round E

Analysis: Shuffled Anagrams

Test set 1

For this test set, since the length of $\mathbf{S} \leq 8$, we can try every permutation of characters and check whether there exists a permutation such that for all i, $S[i] \neq A[i]$. To find every permutation, we can first convert the string to a character array. Then, we swap the first element with every other element and recursively find permutations of the rest of the string.

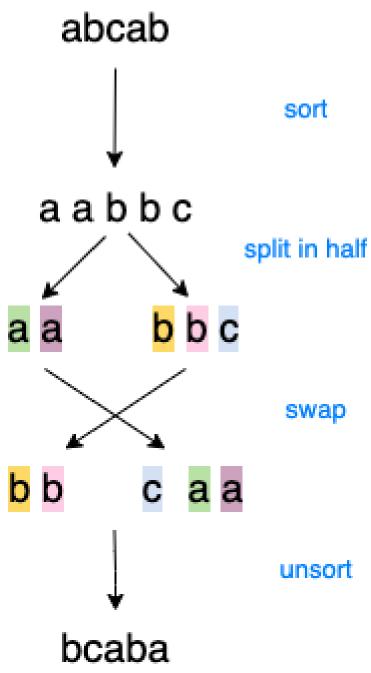
This can be performed in $O(N!)$, where N is the length of $\mathbf{S}$.

Test set 2

For this test set, the above solution would exceed the time limits.

The key observation here is that if a character exists more than $\lfloor \frac{N}{2} \rfloor$ times, then it's impossible to find such a permutation, because at least one position will have a letter that stays the same. Otherwise, we can sort the letters and keep track of the initial position of each letter.

Let the new sorted letters be P. We can split the sorted letters into two halves, from index 0 to $\frac{N}{2}$, $P[0:\frac{N}{2}]$, and from $\frac{N}{2}$ to the end, $P[\frac{N}{2}:]$. If N is odd, split P such that the second half has an extra letter, where the first half is 0 to $\lfloor \frac{N}{2} \rfloor$ and the second half is from $\lceil \frac{N}{2} \rceil$ to the end. Then, we put each character from the second half of the sorted letters $P[i+(\frac{N}{2})]$ into the original position of the corresponding letter in the first half $P[i]$. Similarly, we put each character from the first half of the sorted letters $P[i]$ into the original position of the corresponding letter in the second half $P[i+(\frac{N}{2})]$. Note that if N is odd, the second half of the sorted letters $P[i+(\frac{N}{2})]$ will occupy the first $\lfloor \frac{N}{2} \rfloor + 1$ spaces, while the original first half will occupy the last $\lfloor \frac{N}{2} \rfloor$ spaces, as shown in the example below. The letter orginally at $P[N-1]$ will be in the middle of the array after the swap, replacing $P[i+\lfloor \frac{N}{2} \rfloor]$.



This works because we know that no more than half the characters are equal, and hence the character at $P[i]$ cannot be equal to the letter at $P[i + (\frac{N}{2})]$.

This can be performed in $O(N \log N)$, due to sorting. However, due to the limited size of the alphabet, we can actually sort even faster using a non-comparative sorting algorithm such as [counting sort].