Code Jam 2022 - Round 1C

Analysis: Letter Blocks

Let us first notice that this problem is equivalent to finding an order in which partial strings should be concatanted such that occurrences of the same letter appear together.

Test Set 1

Since $\mathbf{N}$ is at most 6, the number of permutations of these strings is at most $6! = 720$.

We can therefore generate all permutations and for each of them we need to verify the final string which results of concatenating the input strings in that particular order.

Let us take a look at the string `CCCABDAEEF`. To verify the string it is enough to:

- Get a set of letters: {A, B, C, D, E, F}
- Create a grouped represantation of a string: `"CABDAEF"`
- The string is good if and only if the lengths coincide.

If for at least one permutation the size matches, we can print out this string. Otherwise, it's impossible.

The time complexity of this solution is $O(\mathbf{N!} \times \sum_{i=1}^{N} |\mathbf{S_i}|)$, which means $\mathbf{N!}$ times the sum of the lengths of the input strings, because there are $\mathbf{N!}$ permutations and the verification of the final string takes linear time.

Test Set 2

In this test set $\mathbf{N}$ can be 100, so $\mathbf{N!}$ is too large to enumerate all permutations.

First of all, we can verify that each of the strings $\mathbf{S_i}$ meets the requirements from the task. This can be done by applying the verification method described in the section above for each $\mathbf{S_i}$ individually. If the verification fails for one of the input strings, then it will certainly fail for any permutation of them and therefore we output `IMPOSSIBLE`.

Let us call middle letters all letters other than the first and last consecutive segment of letters. Next, let us notice that if the string $\mathbf{S_i}$ has more than 2 distinct letters, then:

- If a letter is a middle letter in more than one input string, then those occurrences will not be together in the final string regardless of the order in which we concatenate them. Therefore, this case is impossible.
- If the middle letters exist in a single input string, they don't influence the outcome as those occurrences will be together in the final string regardless of the order in which we concatenate. Therefore, in this case we can assume the input string is simply two letters long removing everything except the first and last letter of it.

Therefore, for each middle letter we can count in how many strings it appears and if the answer is more than 1 for any of them, we can print out `IMPOSSIBLE`.

Since we also verified each string already, we know that each letter appears only in 1 block inside each $\mathbf{S_i}$.

After this step the problem is now simplified into strings of two forms:

- X: Represents the string consisting of only one block of letter X.
- XY: Reperesents the string starting with a block of X letters and ending with a block of Y letters.

If there are two strings of the form X for the same letter, we can concatenate them as they can't be separated by other strings in the final solution. If there are two strings of the form X_1Y_1 and X_2Y_2, then the answer is `IMPOSSIBLE` if $X_1 = X_2$ (due to Y_1) or $Y_1 = Y_2$ (due to X_2), because it means that in any ordering, there would be at least one block of a different letter between letters X_1 and Y_1.

Therefore for each string $\mathbf{S_i}$, we can create the following mappings using sets:

- If $\mathbf{S_i}$ is of the form X, then insert $\mathbf{S_i}$ into $single[X]$.
- If $\mathbf{S_i}$ is of the form XY, then insert $\mathbf{S_i}$ into both $starts[X]$ and $ends[X]$.

In other words, $single[X]$, $starts[X]$ and $ends[X]$ must all contain exactly 1 element for each letter X. If the element already exists, we return `IMPOSSIBLE`.

Starting string

Let us consider starting string as the input string which is not forced by any previous strings in the final answer. When can the given string $\mathbf{S_i}$ be a starting string?

- If $\mathbf{S_i}$ is of the form X, then there must be no other strings ending with letter X, i.e. $ends[X] = \mathbf{S_i}$.
- If $\mathbf{S_i}$ is of the form XY, then $starts[X] = \mathbf{S_i}$ and $ends[X] = null$ and $single[X] = null$.

With these two conditions, we consider a set of candidates C containing all such starting strings.

Extending the block

Let us consider we already built the partial answer A which ends with letter c. If there exists a string at $single[c]$, it is the last chance to append it, because otherwise it would be separated by at least one block of another letter. Similarly, if there exists a string at $starts[c]$, we must extend it now for the same reason.

How to choose the starting string?

It turns out that for starting a new block, we can choose an arbitrary candidate from the candidates set.

$Proof$: Let us assume that we picked string $\mathbf{S_i}$ as the starting string but the optimal solution started the block with $\mathbf{S_j}$. Let us consider the swapped optimal solution in which we swap these blocks.

Let $\mathbf{S_i}$ start with letter a and $\mathbf{S_j}$ with letter b. Then:

- Optimal = |b..X|a....Y|
- Optimal (swapped) = |a....Y|b..X|

Let us consider what happens after swapping:

- Middle letters of these blocks: Any letters between b and a can't be after a and any letters after a can't be before a. Therefore after swap they also remain fine.
- Letters b: Optimal solution can't have any b letters after a. Therefore this swap is okay.
- Letters a: Since $\mathbf{S_i}$ belongs to the candidate set, therefore $ends[a] = \mathbf{S_i}$ or $ends[a] = null$. It means, that there are either no strings ending in a or $\mathbf{S_i}$ is the only string ending with a. Therefore X can't end with a and the swap remains correct.

Final solution

Repeat:

- 1. Pick an arbitrary string from the candidate's set and start the block with it.
- 2. Let e = last letter of the current solution. If starts[e] != null, add starts[e] to current solution and repeat step 2. Otherwise goto step 1.
- 3. If candidate's set is empty, print solution. Otherwise, print `IMPOSSIBLE`.

Time complexity: $O(\mathbf{N} \times \sum_{i=1}^{N} |\mathbf{S_i}|)$, since we are touching each candidate only once.