Intuition

The intuition behind the code seems to be finding the minimum string formed by merging three given strings while maintaining the order of the original strings and taking the common parts into account.

Approach

The approach consists of a minimumString function that takes three input strings a, b, and c. The function then generates all possible permutations of merging these three strings and checks for the minimum length and lexicographically smallest string among them.

To merge two strings a and b, the checkCommonPart function is used. It checks if b is already present at the end of a. If it is, then a is returned as it already contains b. If not, it tries to find the common part between a and b and merges them by removing the common part once.

Complexity

- **Time complexity:**

The code uses three nested loops to generate all possible permutations of merging the three input strings. Therefore, the time complexity is O(3^3),and within the nested loop every time checkCommonPart is called. Inside checkCommonPart one loop is running and inside it substring fuction is called. Time complexity of this checkCommonPart fuction will be O(N^2); So, Total complexity is O(27*N^2) ~ (N^2);

- **Space complexity:**

The space complexity is O(1) as the additional space used in the function is not dependent on the input size.