3106. Lexicographically Smallest String After Operations With Constraint

Description

You are given a string `s` and an integer `k`.

Define a function `distance(s_1, s_2)` between two strings `s_1` and `s_2` of the same length `n` as:

- The **sum** of the **minimum distance** between `s_1[i]` and `s_2[i]` when the characters from `'a'` to `'z'` are placed in a **cyclic** order, for all `i` in the range `[0, n - 1]`.

For example, `distance("ab", "cd") == 4`, and `distance("a", "z") == 1`.

You can **change** any letter of `s` to **any** other lowercase English letter, **any** number of times.

Return a string denoting the **lexicographically smallest** string `t` you can get after some changes, such that `distance(s, t) <= k`.

Example 1:

Input: s = "zbbz", k = 3

Output: "aaaz"

Explanation:

Change `s` to `"aaaz"`. The distance between `"zbbz"` and `"aaaz"` is equal to `k = 3`.

Example 2:

Input: s = "xaxcd", k = 4

Output: "aawcd"

Explanation:

The distance between "xaxcd" and "aawcd" is equal to k = 4.

Example 3:

Input: s = "lol", k = 0

Output: "lol"

Explanation:

It's impossible to change any character as `k = 0`.

Constraints:

- `1 <= s.length <= 100`
- `0 <= k <= 2000`
- `s` consists only of lowercase English letters.