<https://leetcode.com/problems/shifting-letters/description/>

You are given a string s of lowercase English letters and an integer array shifts of the same length.

Call the shift() of a letter, the next letter in the alphabet, (wrapping around so that 'z' becomes 'a').

* For example, shift('a') = 'b', shift('t') = 'u' and shift('z') = 'a'

Now for each shifts[i] = x, we want to shift the first i + 1 letters of s, x times.

Return **the final string after all such shifts to s are applied**.

**Example 1:**

**Input:** s = "abc", shifts = [3,5,9]

**Output:** "rpl"

**Explanation:** We start with "abc".

After shifting the first 1 letters of s by 3, we have "dbc".

After shifting the first 2 letters of s by 5, we have "igc".

After shifting the first 3 letters of s by 9, we have "rpl", the answer.

**Example 2:**

**Input:** s = "aaa", shifts = [1,2,3]

**Output:** "gfd"

**Constraints:**

* 1 <= s.length <= 10^5
* s consists of lowercase English letters.
* shifts.length == s.length
* 0 <= shifts[i] <= 10^9

**Attempt 1: 2023-12-22**

**Solution 1: Prefix Sum (30min)**

**Style 1: Create prefix count sum array for each char, but from backwards**

class Solution {

    public String shiftingLetters(String s, int[] shifts) {

        char[] chars = s.toCharArray();

        int n = shifts.length;

        long[] count = new long[n];

        // Trick: Create prefix count sum array for each char,

        // but from backwards

        // e.g shifts = {26,9,17} means count = {52,26,17}

        // since it sum up from backwards

        count[n - 1] = (long)shifts[n - 1];

        for(int i = n - 2; i >= 0; i--) {

            count[i] = count[i + 1] + (long)shifts[i];

        }

        for(int i = 0; i < n; i++) {

            // Formula need to cover '(chars[i] + count[i] % 26) > 26' case

            // e.g s = "ruu", shifts = {26,9,17}, for the 3rd one 'u' shift

            // 17 times will over 'z', we have to covert 'u' to 'a' based

            // digit number representation then take model on 26(% 26) again

            chars[i] = (char)((chars[i] - 'a' + count[i] % 26) % 26 + 'a');

        }

        return new String(chars);

    }

}

Time Complexity: O(N)

Space Complexity: O(N)

**Style 2: Merge two for loop**

class Solution {

    public String shiftingLetters(String s, int[] shifts) {

        char[] chars = s.toCharArray();

        int n = shifts.length;

        long[] count = new long[n];

        count[n - 1] = (long)shifts[n - 1];

        for(int i = n - 1; i >= 0; i--) {

            if(i != n - 1) {

                count[i] = count[i + 1] + (long)shifts[i];

            }

            chars[i] = (char)((chars[i] - 'a' + count[i] % 26) % 26 + 'a');

        }

        return new String(chars);

    }

}

========================================================================

Or same way as below

class Solution {

    public String shiftingLetters(String s, int[] shifts) {

        char[] chars = s.toCharArray();

        int n = shifts.length;

        long[] count = new long[n];

        count[n - 1] = (long)shifts[n - 1];

        chars[n - 1] = (char)((chars[n - 1] - 'a' + count[n - 1] % 26) % 26 + 'a');

        for(int i = n - 2; i >= 0; i--) {

            count[i] = count[i + 1] + (long)shifts[i];

            chars[i] = (char)((chars[i] - 'a' + count[i] % 26) % 26 + 'a');

        }

        return new String(chars);

    }

}

Time Complexity: O(N)

Space Complexity: O(N)