828. Unique Letter String

DescriptionHintsSubmissionsDiscussSolution

A character is unique in string `S` if it occurs exactly once in it.

For example, in string `S = "LETTER"`, the only unique characters are `"L"` and `"R"`.

Let's define `UNIQ(S)` as the number of unique characters in string `S`.

For example, `UNIQ("LETTER") =  2`.

Given a string `S` with only uppercases, calculate the sum of `UNIQ(substring)` over all non-empty substrings of `S`.

If there are two or more equal substrings at different positions in `S`, we consider them different.

Since the answer can be very large, retrun the answer modulo `10 ^ 9 + 7`.

Example 1:

```
Input: "ABC"
Output: 10
Explanation: All possible substrings are: "A","B","C","AB","BC" and "ABC".
Evey substring is composed with only unique letters.
Sum of lengths of all substring is 1 + 1 + 1 + 2 + 2 + 3 = 10
```

Example 2:

```
Input: "ABA"
Output: 8
Explanation: The same as example 1, except uni("ABA") = 1.
```

Note: `0 <= S.length <= 10000`.

Intuition:

Let's think about how a character can be found as a unique character.

Think about string `"XAXAXXAX"` and focus on making the second `"A"` a unique character.
We can take `"XA(XAXX)AX"` and between `"()"` is our substring.
We can see here, to make the second "A" counted as a uniq character, we need to:

1. insert `"("` somewhere between the first and second `A`
2. insert `")"` somewhere between the second and third `A`

For step 1 we have `"A(XA"` and `"AX(A"`, 2 possibility.
For step 2 we have `"A)XXA"`, `"AX)XA"` and `"AXX)A"`, 3 possibilities.

So there are in total `2 * 3 = 6` ways to make the second `A` a unique character in a substring.
In other words, there are only 6 substring, in which this `A` contribute 1 point as unique string.

Instead of counting all unique characters and struggling with all possible substrings,
we can count for every char in S, how many ways to be found as a unique char.
We count and sum, and it will be out answer.

Explanation:

1. `index[26][2]` record last two occurrence index for every upper characters.
2. Initialise all values in `index` to `-1`.
3. Loop on string S, for every character `c`, update its last two occurrence index to `index[c]`.
4. Count when loop. For example, if "A" appears twice at index 3, 6, 9 seperately, we need to count:
    - For the first "A": (6-3) * (3-(-1))"
    - For the second "A": (9-6) * (6-3)"
    - For the third "A": (N-9) * (9-6)"

```
//python
def uniqueLetterString(self, S):
        index = {c: [-1, -1] for c in string.ascii_uppercase}
        res = 0
        for i, c in enumerate(S):
            k, j = index[c]
            res += (i - j) * (j - k)
            index[c] = [j, i]
        for c in index:
            k, j = index[c]
            res += (len(S) - j) * (j - k)
        return res % (10**9 + 7)
```

```
//C++
int uniqueLetterString(string S) {
        long mod = pow(10,9)+7;
        int index[26][2];
        memset(index,-1,sizeof(index));
        int N = S.size(); int res = 0;
        for(int i=0;i<N;i++)
        {
            int c = S[i] - 'A';

            res = (res+(i-index[c][1])*(index[c][1]-index[c][0]) %
mod)%mod;
            index[c][0] = index[c][1];
            index[c][1] = i;

        }
        for(int i=0;i<26;++i)
        {
```

```
            res = (res + ((N-index[i][1])*(index[i][1]-index[i][0])
%mod))%mod;
        }

        return res;
    }
```