2223. Sum of Scores of Built Strings

Leetcode Link

Problem Description:

In this problem, we are given a string s of length n. Our task is to build the string iteratively one character at a time by prepending each new character to the front of the string. The strings are labeled from 1 to n, where the string with length i is labeled as s_i.

The score of s_i is the length of the longest common prefix between s_i and s (where s = s_n). Given the final string s, our task is to return the sum of the score of every s_i.

Example:

Consider the following example with s = "codeforces":

- 1. We start building the string from the first character, s_1 = "c".
- 2. Then, we prepend the second character, s₂ = "oc".
- 3. Continuing in this manner, s_3 = "doc", s_4 = "edoc", and so on.

The sum of scores for each s_i in this example can be computed as follows:

- 1. The score for s_c ("c") is 0 (since no common prefix in "c" and "codeforces").
- 2. s_oc ("oc") and "codeforces" have a common prefix of length 1 (since the first characters match), so the score is 1.
- 3. s_doc ("doc") and "codeforces" have no common prefix, so the score is 0.
- 4. s_edoc ("edoc") and "codeforces" have no common prefix, so the score is 0.

5. ...

The sum of all scores is 0 + 1 + 0 + 0 + ... = 1.

Solution Approach:

The solution is based on the Z-Algorithm for finding the longest common prefix in a string. The Z-Algorithm computes the length of the longest common prefix between the current substring (starting from every position) and the original string. The Z array z is initialized to the length of the given string s.

We traverse the string s from the 1st index to the end (ignoring the 0th index), calculating the z-values for each index. After calculating the z-array, we find the sum of all the elements in the array z and add the length of the string (n) to the result.

Python Solution:

```
2 python
   class Solution:
       def sumScores(self, s: str) -> int:
           n = len(s)
           z = [0] * n
           l, r = 0, 0
 8
           for i in range(1, n):
 9
10
               if i <= r:
                   z[i] = min(r - i + 1, z[i - l])
11
               while i + z[i] < n and s[z[i]] == s[i + z[i]]:
12
                   z[i] += 1
               if i + z[i] - 1 > r:
14
15
                    l, r = i, i + z[i] - 1
16
            return sum(z) + n
17
```

Java Solution:

```
2 java
   class Solution {
        public long sumScores(String s) {
            int n = s.length();
            int[] z = new int[n];
            int l = 0, r = 0;
            for (int i = 1; i < n; i++) {
 9
                if (i \ll r)
10
                    z[i] = Math.min(r - i + 1, z[i - l]);
11
                while (i + z[i] < n \&\& s.charAt(z[i]) == s.charAt(i + z[i]))
12
13
                    z[i]++;
14
                if (i + z[i] - 1 > r) {
15
                    l = i;
                    r = i + z[i] - 1;
16
17
18
19
20
            long sum = 0;
21
            for (int value : z)
                sum += value;
23
            return sum + n;
24
25 }
```

JavaScript Solution:

```
2 javascript
   class Solution {
        sumScores(s) {
            const n = s.length;
            const z = new Array(n).fill(0);
            let l = 0, r = 0;
 9
            for (let i = 1; i < n; i++) {
                if (i \ll r)
10
                    z[i] = Math.min(r - i + 1, z[i - l]);
11
12
                while (i + z[i] < n \&\& s[z[i]] === s[i + z[i]])
13
                    z[i]++;
14
                if (i + z[i] - 1 > r) {
15
                    l = i;
16
                    r = i + z[i] - 1;
17
18
19
20
            return z.reduce((sum, value) => sum + value, 0) + n;
21
22 }
```

C++ Solution:

```
cpp
 3 class Solution {
   public:
        long long sumScores(string s) {
            const int n = s.length();
            vector<int> z(n);
            int l = 0, r = 0;
10
            for (int i = 1; i < n; ++i) {
                if (i \ll r)
12
                    z[i] = min(r - i + 1, z[i - l]);
13
                while (i + z[i] < n \&\& s[z[i]] == s[i + z[i]])
14
                    ++z[i];
                if (i + z[i] - 1 > r) {
                   l = i;
                    r = i + z[i] - 1;
18
19
20
21
            return accumulate(begin(z), end(z), 0LL) + n;
22
23 };
C# Solution:
```

```
csharp
   public class Solution {
       public long SumScores(string s) {
            int n = s.Length;
            int[] z = new int[n];
            int l = 0, r = 0;
            for (int i = 1; i < n; i++) {
9
10
                if (i \ll r)
11
                    z[i] = Math.Min(r - i + 1, z[i - l]);
                while (i + z[i] < n \&\& s[z[i]] == s[i + z[i]])
12
13
                    z[i]++;
                if (i + z[i] - 1 > r) {
14
                    l = i;
                    r = i + z[i] - 1;
16
17
18
19
            long sum = 0;
20
            foreach (int value in z)
                sum += value;
            return sum + n;
```

22 23 24 25 } In conclusion, the above-discussed solutions use a smart approach that is Z-Algorithm to build the string iteratively and find the

length of the longest common prefix. This can be easily implemented in different programming languages like Python, Java,

JavaScript, C++, and C#. The overall time complexity of the above solutions is O(n), where n is the length of the input string.



Get Premium