(/)

8050. Count K-Subsequences of a String With Maximum Beauty

ıtest/biweekly-contest-112/problems/count-k-subsequences-of-a-string-with-maximum-beauty/submissions/)

test/biweekly-contest-112/)

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Hard

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

A **k-subsequence** is a **subsequence** of `s`, having length `k`, and all its characters are **unique**, **i.e.**, every character occurs once.

Let `f(c)` denote the number of times the character `c` occurs in `s`.

The **beauty** of a **k-subsequence** is the **sum** of `f(c)` for every character `c` in the k-subsequence.

For example, consider `s = "abbbdd"` and `k = 2`:

- `f('a') = 1`, `f('b') = 3`, `f('d') = 2`
- Some k-subsequences of `s` are:
  - `"abbbdd" -> "ab"` having a beauty of `f('a') + f('b') = 4`
  - `"abbbdd" -> "ad"` having a beauty of `f('a') + f('d') = 3`
  - `"abbbdd" -> "bd"` having a beauty of `f('b') + f('d') = 5`

Return *an integer denoting the number of k-subsequences whose **beauty** is the **maximum** among all **k-subsequences***. Since the answer may be too large, return it modulo $10^9 + 7$.

A subsequence of a string is a new string formed from the original string by deleting some (possibly none) of the characters without disturbing the relative positions of the remaining characters.

Notes

- `f(c)` is the number of times a character `c` occurs in `s`, not a k-subsequence.
- Two k-subsequences are considered different if one is formed by an index that is not present in the other. So, two k-subsequences may form the same string.

Example 1:

```
Input: s = "bcca", k = 2
Output: 4
Explanation: From s we have f('a') = 1, f('b') = 1, and f('c') = 2.
The k-subsequences of s are:
bcca having a beauty of f('b') + f('c') = 3
bcca having a beauty of f('b') + f('c') = 3
bcca having a beauty of f('b') + f('a') = 2
bcca having a beauty of f('c') + f('a') = 3
bcca having a beauty of f('c') + f('a') = 3
There are 4 k-subsequences that have the maximum beauty, 3.
Hence, the answer is 4.
```

Example 2:

```
Input: s = "abbcd", k = 4
Output: 2
Explanation: From s we have f('a') = 1, f('b') = 2, f('c') = 1, and f('d') = 1.
The k-subsequences of s are:
abbcd having a beauty of f('a') + f('b') + f('c') + f('d') = 5
abbcd having a beauty of f('a') + f('b') + f('c') + f('d') = 5
There are 2 k-subsequences that have the maximum beauty, 5.
Hence, the answer is 2.
```

Constraints:

- 1 <= s.length <= $2 * 10^5$
- 1 <= k <= s.length
- s consists only of lowercase English letters.

JavaScript

```
1  ////////////////////////////// Template ////////////////////////////////////
2  const ll = BigInt, mod = ll(1e9 + 7);
3  let N;
4
5  let fact, ifact, inv;
6  const comb_init = () => {
7      fact = Array(N), ifact = Array(N), inv = Array(N);
8      fact[0] = ifact[0] = inv[1] = 1n;
9      for (let i = 2; i < N; i++) inv[i] = (mod - mod / ll(i)) * inv[mod %
   ll(i)] % mod;
10     for (let i = 1; i < N; i++) {
11         fact[i] = fact[i - 1] * ll(i) % mod;
12         ifact[i] = ifact[i - 1] * inv[i] % mod;
13     }
14 };
```

```
15
16  const comb = (n, k) => {
17      if (n < k || k < 0) return 0n;
18      return fact[n] * ifact[k] % mod * ifact[n - k] % mod;
19  };
20  ////////////////////////////////////////////////////////////////////
21
22  const ord = (c) => c.charCodeAt();
23
24  const M = 1e9 + 7;
25  const countKSubsequencesWithMaxBeauty = (s, k) => {
26      N = s.length + 1;
27      comb_init();
28      let f = Array(26).fill(0), res = 1, pick = 0;
29      for (const c of s) f[ord(c) - 97]++;
30      f = f.sort((x, y) => y - x).filter(x => x > 0);
31      for (let i = 0; i < k; i++) {
32          if (f[i] >= f[k - 1]) {
33              res *= f[i];
34              res %= M;
35          }
36          if (f[i] == f[k - 1]) pick++;
37      }
38      let lastPick = comb(f.filter(x => x == f[k - 1]).length, pick);
39      res = ll(res) * lastPick % mod;
40      return res;
41  };
```

Custom Testcase Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/1039576414/)

More Details > (/submissions/detail/1039576414/)

Share your acceptance!