482 License Key Formatting

2018年4月12日 21:02

You are given a license key represented as a string S which consists only alphanumeric character and dashes. The string is separated into N+1 groups by N dashes.
Given a number K, we would want to reformat the strings such that each group contains *exactly* K characters, except for the first group which could be shorter than K, but still must contain at least one character. Furthermore, there must be a dash inserted between two groups and all lowercase letters should be converted to uppercase.
Given a non-empty string S and a number K, format the string according to the rules described above.

Example 1:
Input: S = "5F3Z-2e-9-w", K = 4
Output: "5F3Z-2E9W"
Explanation: The string S has been split into two parts, each part has 4 characters.
Note that the two extra dashes are not needed and can be removed.

Example 2:
Input: S = "2-5g-3-J", K = 2
Output: "2-5G-3J"
Explanation: The string S has been split into three parts, each part has 2 characters except the first part as it could be shorter as mentioned above.

Note:

1. The length of string S will not exceed 12,000, and K is a positive integer.
2. String S consists only of alphanumerical characters (a-z and/or A-Z and/or 0-9) and dashes(-).
3. String S is non-empty.

来自 <https://leetcode.com/problems/license-key-formatting/description/>

Solution for Python3:

```
1  class Solution:
2      def licenseKeyFormatting(self, S, K):
3          """
4          :type S: str
5          :type K: int
6          :rtype: str
7          """
```

```
        S = S.replace('-', '').upper()
        st = len(S) % K or K
        s = S[:st]
        while st < len(S):
            s += '-' + S[st:st+K]
            st += K
        return s
```

Solution for C++:

```
class Solution {
public:
    string licenseKeyFormatting(string S, int K) {
        string res;
        for (auto i = S.rbegin(); i < S.rend(); i++) {
            if (*i != '-') {
                if (res.size() % (K + 1) == K)
                    res += '-';
                res += toupper(*i);
            }
        }
        reverse(res.begin(), res.end());
        return res;
    }
};
```