第 396 场周赛

1. 100284. 有效单词

有效单词 需要满足以下几个条件：

- **至少** 包含 3 个字符。
- 由数字 0-9 和英文大小写字母组成。（不必包含所有这类字符。）
- **至少** 包含一个 **元音字母** 。
- **至少** 包含一个 **辅音字母** 。

给你一个字符串 `word` 。如果 `word` 是一个有效单词，则返回 `true` ，否则返回 `false` 。

注意：

- `'a'`、`'e'`、`'i'`、`'o'`、`'u'` 及其大写形式都属于 **元音字母** 。
- 英文中的 **辅音字母** 是指那些除元音字母之外的字母。

示例 1：

输入： word = "234Adas"

输出： true

解释：

这个单词满足所有条件。

示例 2：

输入： word = "b3"

输出： false

解释：

这个单词的长度少于 3 且没有包含元音字母。

示例 3：

输入： word = "a3$e"

输出： false

解释：

这个单词包含了 `'$'` 字符且没有包含辅音字母。

提示：

- `1 <= word.length <= 20`
- `word` 由英文大写和小写字母、数字、`'@'`、`'#'` 和 `'$'` 组成。

思路

按照题意依次遍历

1. Java

```java
class Solution {
    public boolean isValid(String word) {
        if (word.length() < 3) {
            return false;
        }
        boolean[] f = new boolean[2];
        Arrays.fill(f, false);
        for (char c : word.toCharArray()) {
            if (Character.isAlphabetic(c)) {
                c = Character.toLowerCase(c);
                f[c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u' ? 1 : 0] =
true;
            } else if (!Character.isDigit(c)) {
                // 既不是字母也不是数字直接 return false
                return false;
            }
        }
        return f[0] & f[1];
    }
}
```

2. Python

```python
class Solution:
    def isValid(self, word: str) -> bool:
        if len(word) < 3:
            return False
        flag1 = flag2 = False
        for c in word:
            if c in "@#$":
                return False
            # 这里也可以写 if c.isalpha():
            if (ord('a') <= ord(c) <= ord('z')) or (ord('A') <= ord(c) <= ord('Z')):
                if c.lower() in "aeiou":
                    flag1 = True
                else:
                    flag2 = True
        return flag1 and flag2
```

2. 100275. K 周期字符串需要的最少操作次数

给你一个长度为 `n` 的字符串 `word` 和一个整数 `k` ，其中 `k` 是 `n` 的因数。

在一次操作中，你可以选择任意两个下标 `i` 和 `j` ，其中 `0 <= i, j < n` ，且这两个下标都可以被 `k` 整除，然后用从 `j` 开始的长度为 `k` 的子串替换从 `i` 开始的长度为 `k` 的子串。也就是说，将子串 `word[i..i + k - 1]` 替换为子串 `word[j..j + k - 1]` 。

返回使 `word` 成为 **K 周期字符串** 所需的 **最少** 操作次数。

如果存在某个长度为 `k` 的字符串 `s`，使得 `word` 可以表示为任意次数连接 `s` ，则称字符串 `word` 是 **K 周期字符串** 。例如，如果 `word == "ababab"`，那么 `word` 就是 `s = "ab"` 时的 2 周期字符串 。

示例 1：

输入： word = "leetcodeleet", k = 4

输出： 1

解释： 可以选择 i = 4 和 j = 0 获得一个 4 周期字符串。这次操作后，word 变为 "leetleetleet" 。

示例 2：

输入： word = "leetcoleet", k = 2

输出： 3

解释： 可以执行以下操作获得一个 2 周期字符串。

i	j	word
0	2	etetcoleet
4	0	etetetleet
6	0	etetetetet

提示：

- `1 <= n == word.length <= 105`
- `1 <= k <= word.length`
- `k` 能整除 `word.length` 。
- `word` 仅由小写英文字母组成。

思路

根据题意，其实是用长度为k去分割这个字符串，然后去用分割完的字符串去替换其他的字符串，要想操作次数是最少的，显然要替换的次数是最少的，那么就要用出现次数最多的子串去替换其他的字符串

1. Java

```java
public class Solution {
    public int minimumOperationsToMakeKPeriodic(String word, int k) {
        int n = word.length();
        int mx = 0;
        HashMap<String, Integer> cnt = new HashMap<>();
        for (int i = k; i <= n; i += k) {
            String sub = word.substring(i - k, i);
            int c = cnt.merge(sub, 1, Integer::sum);
            mx = Math.max(mx, c);
        }
        return n / k - mx;
    }
}
```

2. Python

```python
class Solution:
    def minimumOperationsToMakeKPeriodic(self, word: str, k: int) -> int:
        n = len(word)
        cnt = Counter(word[i - k: i] for i in range(k, n + 1, k))
        mx = max(cnt.values())
        return n // k - mx
```

3. 3138. 同位字符串连接的最小长度

给你一个字符串 `s`，它由某个字符串 `t` 和它的 **同位字符串** 连接而成。

请你返回字符串 `t` 的 **最小** 可能长度。

同位字符串 指的是重新排列一个单词得到的另外一个字符串，原来字符串中的每个字符在新字符串中都恰好只使用一次。

示例 1：

输入： s = "abba"

输出： 2

解释：

一个可能的字符串 `t` 为 `"ba"` 。

示例 2：

输入： s = "cdef"

输出： 4

解释：

一个可能的字符串 `t` 为 `"cdef"`，注意 `t` 可能等于 `s` 。

提示：

- `1 <= s.length <= 105`
- `s` 只包含小写英文字母。

思路

题目的意思是可以把字符串 s 分割为几个子串 这些子串按照字典序排列是一样的 比如 "aab" "aba" "baa" 都是 "aab" 的同位字符串 观察案例可以发现 return 的同位字符串的长度一定是 s 长度的一个因子 那么只需要枚举 len(s) 的所有因子 然后根据因子 k 依次去分割字符串 s 得到子串 只需要统计每个子串中每个字母的出现次数 如果都相同 那么 k 就是合法的 如果有一个不同 代表 k 不合法 遍历完成之后如果都没有找到 k 那么返回的就是 s 自身的长度

1. Java

```java
class Solution {
    public int minAnagramLength(String S) {
        char[] s = S.toCharArray();
        int n = s.length;
        next:
        for (int k = 1; k <= n / 2; k++) {
            if (n % k > 0) {
                continue;
            }
            int[] cnt0 = new int[26];
            for (int j = 0; j < k; j++) {
                cnt0[s[j] - 'a']++;
            }
            for (int i = k * 2; i <= n; i += k) {
                int[] cnt = new int[26];
                for (int j = i - k; j < i; j++) {
                    cnt[s[j] - 'a']++;
                }
                if (!Arrays.equals(cnt, cnt0)) {
                    continue next;
                }
            }
            return k;
        }
        return n;
    }
}
```

2. Python

```python
class Solution:
    def minAnagramLength(self, s: str) -> int:
        n = len(s)
        for k in range(1, n // 2 + 1):
            if n % k:
                continue
            cnt0 = Counter(s[:k])
            flag = True
            for i in range(k * 2, n + 1, k):
                if Counter(s[i - k: i]) != cnt0:
                    flag = False
                    break
            if flag:
                return k
        return n
```