LeetCode 1408. 数组中的字符串匹配

1408. 数组中的字符串匹配

难度 简单 4 收藏 分享 切换为英文 关注 反馈

给你一个字符串数组 `words` ，数组中的每个字符串都可以看作是一个单词。请你按 **任意** 顺序返回 `words` 中是其他单词的子字符串的所有单词。

如果你可以删除 `words[j]` 最左侧和/或最右侧的若干字符得到 `word[i]` ，那么字符串 `words[i]` 就是 `words[j]` 的一个子字符串。

示例 1：

```
输入：words = ["mass","as","hero","superhero"]
输出：["as","hero"]
解释："as" 是 "mass" 的子字符串，"hero" 是 "superhero" 的子字符串。
["hero","as"] 也是有效的答案。
```

示例 2：

```
输入：words = ["leetcode","et","code"]
输出：["et","code"]
解释："et" 和 "code" 都是 "leetcode" 的子字符串。
```

示例 3：

```
输入：words = ["blue","green","bu"]
输出：[]
```

提示：

- `1 <= words.length <= 100`
- `1 <= words[i].length <= 30`
- `words[i]` 仅包含小写英文字母。
- 题目数据 **保证** 每个 `words[i]` 都是独一无二的。

```cpp
class Solution {
public:
    bool check(const vector<string> &words, int i) {
        int n = words.size();
        for (int j = 0; j < n; j++)
            if (j != i)
                if (words[j].find(words[i]) != string::npos)
                    return true;

        return false;
    }

    vector<string> stringMatching(vector<string>& words) {
```

```
        int n = words.size();
        vector<string> ans;
        for (int i = 0; i < n; i++)
            if (check(words, i))
                ans.push_back(words[i]);

        return ans;
    }
};
```

LeetCode 1409. 查询带键的排列

1409. 查询带键的排列

难度 中等 7 收藏 分享 切换为英文 关注 反馈

给你一个待查数组 `queries` ，数组中的元素为 `1` 到 `m` 之间的正整数。 请你根据以下规则处理所有待查项 `queries[i]` （从 `i=0` 到 `i=queries.length-1` ）：

- 一开始，排列 `P=[1,2,3,...,m]` 。
- 对于当前的 `i` ，请你找出待查项 `queries[i]` 在排列 `P` 中的位置（**下标从 0 开始**），然后将其从原位置移动到排列 `P` 的起始位置（即下标为 0 处）。注意， `queries[i]` 在 `P` 中的位置就是 `queries[i]` 的查询结果。

请你以数组形式返回待查数组 `queries` 的查询结果。

示例 1：

```
输入：queries = [3,1,2,1], m = 5
输出：[2,1,2,1]
解释：待查数组 queries 处理如下：
对于 i=0: queries[i]=3, P=[1,2,3,4,5], 3 在 P 中的位置是 2，接着我们把 3 移动到 P 的起始位置，得到 P=[3,1,2,4,5] 。
对于 i=1: queries[i]=1, P=[3,1,2,4,5], 1 在 P 中的位置是 1，接着我们把 1 移动到 P 的起始位置，得到 P=[1,3,2,4,5] 。
对于 i=2: queries[i]=2, P=[1,3,2,4,5], 2 在 P 中的位置是 2，接着我们把 2 移动到 P 的起始位置，得到 P=[2,1,3,4,5] 。
对于 i=3: queries[i]=1, P=[2,1,3,4,5], 1 在 P 中的位置是 1，接着我们把 1 移动到 P 的起始位置，得到 P=[1,2,3,4,5] 。
因此，返回的结果数组为 [2,1,2,1] 。
```

示例 2：

```
输入：queries = [4,1,2,2], m = 4
输出：[3,1,2,0]
```

```
class Solution {
public:
    vector<int> processQueries(vector<int>& queries, int m) {
        vector<int> p(m);
        for (int i = 0; i < m; i++)
            p[i] = i + 1;

        vector<int> ans;
        for (int q : queries) {
```

```
            for (int i = 0; i < m; i++)
                if (q == p[i]) {
                    ans.push_back(i);
                    int x = p[i];
                    for (int j = i - 1; j >= 0; j--)
                        p[j + 1] = p[j];
                    p[0] = x;
                    break;
                }
        }

        return ans;
    }
};
```

1410. HTML 实体解析器

1410. HTML 实体解析器

难度 中等 5 收藏 分享 切换为英文 关注 反馈

「HTML 实体解析器」 是一种特殊的解析器，它将 HTML 代码作为输入，并用字符本身替换掉所有这些特殊的字符实体。

HTML 里这些特殊字符和它们对应的字符实体包括：

- **双引号：** 字符实体为 `&quot;` ，对应的字符是 `"` 。
- **单引号：** 字符实体为 `&apos;` ，对应的字符是 `'` 。
- **与符号：** 字符实体为 `&` ，对应对的字符是 `&` 。
- **大于号：** 字符实体为 `>` ，对应的字符是 `>` 。
- **小于号：** 字符实体为 `<` ，对应的字符是 `<` 。
- **斜线号：** 字符实体为 `&frasl;` ，对应的字符是 `/` 。

给你输入字符串 `text` ，请你实现一个 HTML 实体解析器，返回解析器解析后的结果。

示例 1：

```
输入：text = "&amp; is an HTML entity but &ambassador; is not."
输出："& is an HTML entity but &ambassador; is not."
解释：解析器把字符实体 &amp; 用 & 替换
```

示例 2：

```
输入：text = "and I quote: &quot;...&quot;"
输出："and I quote: \"...\""
```

示例 3：

```
输入：text = "Stay home! Practice on Leetcode :)"
输出："Stay home! Practice on Leetcode :)"
```

```
class Solution {
public:
    bool check(const string &s, int l, const string &t) {
        if (l < t.length())
```

```
            return false;

        for (int i = l - 1, j = t.length() - 1; j >= 0; i--, j--)
            if (s[i] != t[j])
                return false;

        return true;
    }

    string entityParser(string text) {
        vector<pair<string, char>> p;
        p.emplace_back("&quot;", '"');
        p.emplace_back("&apos;", '\'');
        p.emplace_back("&amp;", '&');
        p.emplace_back("&gt;", '>');
        p.emplace_back("&lt;", '<');
        p.emplace_back("&frasl;", '/');

        string s(text.size(), ' ');
        int l = 0;

        for (char c : text) {
            s[l++] = c;
            if (c == ';') {
                for (const auto &t : p)
                    if (check(s, l, t.first)) {
                        l -= t.first.length();
                        s[l++] = t.second;
                        break;
                    }
            }
        }

        return s.substr(0, l);
    }
};
```

1411. 给 N x 3 网格图涂色的方案数