# 题目

给你一个字符串数组 queries，和一个表示模式的字符串 pattern，请你返回一个布尔数组 answer 。只有在待查项 queries[i] 与模式串 pattern 匹配时， answer[i] 才为 true，否则为 false。

如果可以将小写字母插入模式串 pattern 得到待查询项 query，那么待查询项与给定模式串匹配。可以在任何位置插入每个字符，也可以不插入字符。

示例 1：

输入：queries = ["FooBar","FooBarTest","FootBall","FrameBuffer","ForceFeedBack"], pattern = "FB"

输出：[true,false,true,true,false]

示例：

"FooBar" 可以这样生成："F" + "oo" + "B" + "ar"。

"FootBall" 可以这样生成："F" + "oot" + "B" + "all".

"FrameBuffer" 可以这样生成："F" + "rame" + "B" + "uffer".

示例 2：

输入：queries = ["FooBar","FooBarTest","FootBall","FrameBuffer","ForceFeedBack"], pattern = "FoBa"

输出：[true,false,true,false,false]

解释：

"FooBar" 可以这样生成："Fo" + "o" + "Ba" + "r".

"FootBall" 可以这样生成："Fo" + "ot" + "Ba" + "ll".

示例 3：

输入：queries = ["FooBar","FooBarTest","FootBall","FrameBuffer","ForceFeedBack"], pattern = "FoBaT"

输出：[false,true,false,false,false]

解释：

"FooBarTest" 可以这样生成："Fo" + "o" + "Ba" + "r" + "T" + "est".

提示：

1 <= pattern.length, queries.length <= 100

1 <= queries[i].length <= 100

queries[i] 和 pattern 由英文字母组成

# 分析

## 方法一：双指针/模拟

思路：

代码：

class Solution {

public:

vector<bool> camelMatch(vector<string>& queries, string pattern) {

vector<bool> result;

for (const string& query : queries) {

result.push\_back(match(query, pattern));

}

return result;

}

private:

bool match(const string& query, const string& pattern) {

int i = 0, j = 0;

while (i < query.length() && j < pattern.length()) {

if (query[i] == pattern[j]) {

i++;

j++;

} else if (islower(query[i]) && query[i] == pattern[j]) {

j++;

} else if (isupper(query[i])) {

return false;

} else {

i++;

}

}

if (j == pattern.length()) {

while (i < query.length() && islower(query[i])) {

i++;

}

return i == query.length();

}

return false;

}

};

或：

class Solution {

public:

vector<bool> camelMatch(vector<string>& queries, string pattern) {

int n = queries.size();

vector<bool> ans(n, true);

// 遍历queries中的每个字符串

for (int i = 0; i< n; i++) {

// pattern的指针

int j = 0;

// 遍历每个字符串的每个字符

for (auto &ch : queries[i]) {

// 如果j在合理值内并且当前的值等于queries中的字符，则递增j

if (j < pattern.size() && pattern[j] == ch) {

j++;

} else if (isupper(ch)) {

ans[i] = false; // 如果是大写字符则设置为false

break;

}

}

if (j < pattern.size()) {

ans[i] = false;

}

}

return ans;

}

};