#5714. 替换字符串中的括号内容

题目描述：

给你一个字符串 `s` ，它包含一些括号对，每个括号中包含一个 **非空** 的键。

- 比方说，字符串 `"(name)is(age)yearsold"` 中，有 **两个** 括号对，分别包含键 `"name"` 和 `"age"` 。

你知道许多键对应的值，这些关系由二维字符串数组 `knowledge` 表示，其中 `knowledge[i] = [keyi, valuei]` ，表示键 `keyi` 对应的值为 `valuei` 。

你需要替换 **所有** 的括号对。当你替换一个括号对，且它包含的键为 `keyi` 时，你需要：

- 将 `keyi` 和括号用对应的值 `valuei` 替换。
- 如果从 `knowledge` 中无法得知某个键对应的值，你需要将 `keyi` 和括号用问号 `"?"` 替换（不需要引号）。

`knowledge` 中每个键最多只会出现一次。 `s` 中不会有嵌套的括号。

请你返回替换 **所有** 括号对后的结果字符串。

示例 1：

```
输入：s = "(name)is(age)yearsold", knowledge = [["name","bob"],["age","two"]]
输出："bobistwoyearsold"
解释：
键 "name" 对应的值为 "bob" ，所以将 "(name)" 替换为 "bob" 。
键 "age" 对应的值为 "two" ，所以将 "(age)" 替换为 "two" 。
```

示例 2：

```
输入：s = "hi(name)", knowledge = [["a","b"]]
输出："hi?"
解释：由于不知道键 "name" 对应的值，所以用 "?" 替换 "(name)" 。
```

示例 3：

```
输入：s = "(a)(a)(a)aaa", knowledge = [["a","yes"]]
输出："yesyesyesaaa"
解释：相同的键在 s 中可能会出现多次。
键 "a" 对应的值为 "yes" ，所以将所有的 "(a)" 替换为 "yes" 。
注意，不在括号里的 "a" 不需要被替换。
```

示例 4：

```
输入：s = "(a)(b)", knowledge = [["a","b"],["b","a"]]
输出："ba"
```

我的做法：（超时）

```
class Solution {
public:
    string evaluate(string s, vector<vector<string>>& knowledge) {
        string res;//结果文件
        for(int i=0;i<s.size();i++){
            if(s[i]!='(')res.push_back(s[i]);
            else{
                string temp;
                while(i<s.size()-1&&s[++i]!=')'){
                    temp.push_back(s[i]);
                }
                int k=0;
                while(k<knowledge.size()&&knowledge[k][0]!=temp)k++;//此
处多次从左向右遍历导致超时
                if(k==knowledge.size()){
                    res.append("?");
                }
                else if(knowledge[k][0]==temp){
                    res.append(knowledge[k][1]);
```

```
                }

            }
        }
         return res;
    }
};
```

使用模板map构建哈希函数以提高搜索的速度

```
class Solution {
public:
    string evaluate(string s, vector<vector<string>>& knowledge) {
        string res;//结果文件
        map<string,string>my_map;//建立哈希映射
        for(auto x:knowledge){
            my_map[x[0]]=x[1];
        }

        for(int i=0;i<s.size();i++){
            if(s[i]!='(')res.push_back(s[i]);
            else{
                string temp;
                while(i<s.size()-1&&s[++i]!=')'){
                    temp.push_back(s[i]);
                }
                /*
                int k=0;
                while(k<knowledge.size()&&knowledge[k][0]!=temp)k++;
                */
                auto flag=my_map.find(temp);
                if(flag==my_map.end()){
                    res.append("?");
                }else{
                    res.append(flag->second);
                }
                /*
                if(k==knowledge.size()){
                    res.append("?");
                }
                else if(knowledge[k][0]==temp){
                    res.append(knowledge[k][1]);
                }
                */
            }
        }
         return res;
    }
```

```
};
```

反思总结：

1. 在处理大型数据的时候，可以通过优化数据结构来提高运行的速度
2. 哈希不失为一个比较好的选择