# 题目

给定一个非空的字符串，判断它是否可以由它的一个子串重复多次构成。给定的字符串只含有小写英文字母，并且长度不超过10000。

示例 1:

输入: "abab"

输出: True

解释: 可由子字符串 "ab" 重复两次构成。

示例 2:

输入: "aba"

输出: False

示例 3:

输入: "abcabcabcabc"

输出: True

解释: 可由子字符串 "abc" 重复四次构成。 (或者子字符串 "abcabc" 重复两次构成。)

# 分析

## 方法一：枚举

思路：

总体思路是遍历一半字符串长度，然后判断这半个字符串是否可以由某个子串重复为整个字符串s（注意这里的下标从1开始）。

这个问题的核心思路是尝试找到一个长度小于等于原始字符串长度一半的子串，使得原始字符串可以由这个子串重复多次构成。

具体实现步骤如下：

1、枚举所有可能的子串长度len，从1到原始字符串长度的一半。

2、对于每个长度为len的子串，检查是否能够构成原始字符串。如果原始字符串长度不能整除len，则跳过该子串长度。

3、构造一个重复拼接子串的目标字符串，并与原始字符串比较是否相等。如果相等，则返回 true。

4、如果遍历完所有可能的子串长度都没有找到合适的子串，则返回 false。

这样就能判断原始字符串是否可以由一个子串重复多次构成了。

代码：

class Solution {

public:

bool repeatedSubstringPattern(string s) {

int n = s.size();

// 枚举所有可能的子串长度，从1到原始字符串长度的一半

for (int len = 1; **len <= n / 2**; len++) {

if (n % len == 0) { // 如果原始字符串长度能整除当前长度

string sub = s.substr(0, len); // 获取当前长度的子串

string target;

// 构造一个重复拼接子串的目标字符串

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

target += sub;

}

// 如果目标字符串与原始字符串相等，则返回true

if (target == s) {

return true;

}

}

}

// 遍历完所有可能的子串长度都没有找到合适的子串，则返回false

return false;

}

};

## 方法二：搜索子字符串/字符串匹配/加倍字符串

思路：

这段代码的实现思路是将原始字符串s拼接两次，得到一个新的字符串 tmpStr。然后在tmpStr中查找s第二次出现的位置（从索引1开始查找）。如果这个位置不是s的长度（即不是从头开始重复的情况），则说明原始字符串可以由一个子串重复多次构成，返回 true；否则返回 false。

这个实现思路利用了字符串的拼接和查找操作，简洁地判断了原始字符串是否可以由一个子串重复多次构成。

假设原始字符串s为"abab"，按照上述代码的实现思路：

1、将原始字符串s拼接两次，得到tmpStr = "abababab"。

2、在tmpStr中查找s第二次出现的位置（从索引1开始查找），即查找 "abab"在"abababab"中的位置。

3、因为 "abab" 从索引1开始可以找到，所以返回 true，表示原始字符串 "abab" 可以由一个子串 "ab" 重复两次构成。

因此，对于原始字符串 "abab"，这段代码会返回 true。

与LeetCode 796思路一样。

代码：

class Solution {

public:

bool repeatedSubstringPattern(string s) {

string tmpStr = s+s;

if(tmpStr.find(s,1)!=s.length())

return true;

return false;

}

};

## 方法三：KMP