# 题目

无重复字符串的排列组合。编写一种方法，计算某字符串的所有排列组合，字符串每个字符均不相同。

示例1:

输入：S = "qwe"

输出：["qwe", "qew", "wqe", "weq", "ewq", "eqw"]

示例2:

输入：S = "ab"

输出：["ab", "ba"]

提示:

字符都是英文字母。

字符串长度在[1, 9]之间。

# 分析

## 方法一：回溯法

思路：

要计算一个字符串的所有排列组合，可以使用回溯法。具体步骤如下：

1、定义一个字符串result用来存储所有排列组合的结果。

2、定义一个布尔型数组visited来标记每个字符是否已经被访问过。

3、定义一个递归函数permute，该函数接收一个当前排列字符串current和一个当前位置pos，表示当前正在考虑第pos个字符。

4、在permute函数中，如果current的长度等于字符串S的长度，则将current添加到result中，并返回。

5、遍历字符串S，如果当前字符没有被访问过，则将其加入current中，标记为已访问，并递归调用permute函数考虑下一个位置。

6、在递归调用返回后，将当前字符标记为未访问，从current中移除该字符，以便考虑其他排列。

7、最终返回result。

下面是具体的实现：

class Solution {

public:

vector<string> permutation(string S) {

vector<string> result;

string current = "";

vector<bool> visited(S.length(), false);

permute(S, current, visited, result);

return result;

}

private:

void permute(const string& S, string& current, vector<bool>& visited, vector<string>& result) {

if (current.length() == S.length()) {

result.push\_back(current);

return;

}

for (int i = 0; i < S.length(); ++i) {

if (!visited[i]) {

current += S[i];

visited[i] = true;

permute(S, current, visited, result);

visited[i] = false;

current.pop\_back();

}

}

}

};

在这个实现中，我们首先定义了一个result数组来存储结果，然后定义了一个visited数组来标记每个字符是否已经被访问过。接着，我们定义了一个递归函数permute，在该函数中，我们首先判断当前排列字符串current的长度是否等于字符串S的长度，如果是则将current添加到result中并返回。然后，我们遍历字符串S，对于每个未被访问过的字符，我们将其加入current中，并递归调用permute函数考虑下一个位置。在递归调用返回后，我们将当前字符标记为未访问，并从current中移除该字符，以便考虑其他排列。最终，我们返回result数组即为所有排列组合的结果。