# 题目

给定一个布尔表达式和一个期望的布尔结果result，布尔表达式由0 (false)、1 (true)、& (AND)、| (OR)和^ (XOR)符号组成。实现一个函数，算出有几种可使该表达式得出result值的括号方法。

示例 1:

输入: s = "1^0|0|1", result = 0

输出: 2

解释: 两种可能的括号方法是

1^(0|(0|1))

1^((0|0)|1)

示例 2:

输入: s = "0&0&0&1^1|0", result = 1

输出: 10

提示：

运算符的数量不超过 19 个

# 分析

## 方法一：回溯法

思路：

这个算法使用了递归和记忆化搜索的思想，避免了重复计算相同子问题，从而提高了效率。

1、定义一个哈希表unordered\_map<string, int> memo来存储已经计算过的子问题的结果，其中键为表达式字符串和期望结果的组合，值为该组合下的计算结果。

2、在递归函数countEvalHelper中，如果当前的表达式字符串和期望结果已经在哈希表中存在，直接返回哈希表中存储的结果，避免重复计算。

3、在每次递归过程中，根据运算符将表达式分为左右两部分，并递归计算左右两部分得到期望结果的方式数。

4、根据运算符进行组合计算，分别处理AND、OR、XOR运算符，并根据结果值将计算结果加到count变量中。

5、最后将当前表达式字符串和期望结果的组合及计算结果存入哈希表，返回计算结果。

这样可以避免重复计算相同子问题，提高了算法的效率。

代码：

class Solution {

public:

int countEval(string s, int result) {

unordered\_map<string, int> memo;

return countEvalHelper(s, result, memo);

}

private:

int countEvalHelper(string s, int result, unordered\_map<string, int>& memo) {

if (s.empty()) {

return 0;

}

if (s.length() == 1) {

return (s[0] - '0' == result) ? 1 : 0;

}

string key = s + to\_string(result);

if (memo.find(key) != memo.end()) {

return memo[key];

}

int count = 0;

for (int i = 1; i < s.length(); i += 2) {

char op = s[i];

string left = s.substr(0, i);

string right = s.substr(i + 1);

int leftTrue = countEvalHelper(left, 1, memo);

int leftFalse = countEvalHelper(left, 0, memo);

int rightTrue = countEvalHelper(right, 1, memo);

int rightFalse = countEvalHelper(right, 0, memo);

if (op == '&') {

if (result == 1) {

count += leftTrue \* rightTrue;

} else {

count += leftFalse \* rightFalse;

count += leftTrue \* rightFalse;

count += leftFalse \* rightTrue;

}

} else if (op == '|') {

if (result == 1) {

count += leftTrue \* rightTrue;

count += leftTrue \* rightFalse;

count += leftFalse \* rightTrue;

} else {

count += leftFalse \* rightFalse;

}

} else if (op == '^') {

if (result == 1) {

count += leftTrue \* rightFalse;

count += leftFalse \* rightTrue;

} else {

count += leftTrue \* rightTrue;

count += leftFalse \* rightFalse;

}

}

}

memo[key] = count;

return count;

}

};