1. 题目
2. 题干

数字 n 代表生成括号的对数，请你设计一个函数，用于能够生成所有可能的并且 **有效的**括号组合。

1. 示例

输入：n = 3

输出：[

"((()))",

"(()())",

"(())()",

"()(())",

"()()()"

]

1. 题解
2. 思路

输入数字n，生成2n长度的字符串，字符串中有n个左括号和n个右括号，生成所有括号的排列即可，再通过判断序列的有效性来生成最终的结果序列。对于这种生成所有序列的问题，从空字符串开始，一个一个地往字符串中添加左括号或者右括号，可以使用递归来解决。

如，先往字符串中添加左括号，再不断进入新的递归中，直到生成了2n长度的左括号，继而判断字符串是否合法；然后回退到上一个递归中，删除最后一个左括号，添加右括号，进入新的递归中，继而判断字符串是否合法。这样，就能够生成所有2n长度的字符串了，通过判断字符串的合法性，生成最终的结果字符串。

总而言之，递归就是不断地添加左括号，删除左括号，添加右括号，删除右括号。

判断字符串有效性是通过比较左右括号的个数来实现的，若两者个数不相同，那么说明该字符串不合法。

1. 代码实现
2. C++

class Solution {

//判断字符串有效性

bool valid(const string& str) {

int balance = 0;

for (char c : str) {

if (c == '(') {

++balance;

}

else {

--balance;

}

if (balance < 0)

return false;

}

return balance == 0;

}

//生成所有的字符串

void generate\_all(string& current, int n, vector<string>& result) {

if (n == current.size()) {

if (valid(current)) {

result.push\_back(current);

}

return;

}

current += '(';

generate\_all(current, n, result);

current.pop\_back();

current += ')';

generate\_all(current, n, result);

current.pop\_back();

}

public:

vector<string> generateParenthesis(int n) {

vector<string> result;

string current;

generate\_all(current, n \* 2, result);

return result;

}

};

1. Java

import java.util.ArrayList;  
import java.util.List;  
  
public class All22 {  
 public List<String> generateParenthesis(int n){  
 List<String> combinations = new ArrayList<String>();  
 generateAll(new char[2\*n], 0, combinations);  
 return combinations;  
 }  
  
 public void generateAll(char[] current, int pos, List<String> result){  
 if(pos == current.length){  
 if(valid(current)){  
 result.add(new String(current));  
 }  
 }else{  
 current[pos] = '(';  
 generateAll(current, pos+1, result);  
 current[pos] = ')';  
 generateAll(current, pos +1, result);  
 }  
 }  
  
 public boolean valid(char[] current){  
 int balance = 0;  
 for(char c : current){  
 if(c == '('){  
 ++balance;  
 }else{  
 --balance;  
 }  
 if(balance<0){  
 return false;  
 }  
 }  
 return balance==0;  
 }  
}