0022. 括号生成

ITCharge 大约 3 分钟

- 标签：字符串、回溯算法
- 难度：中等

题目链接

- 0022. 括号生成 - 力扣

题目大意

描述：给定一个整数 n，代表生成括号的对数。

要求：生成所有有可能且有效的括号组合。

说明：

- $1 \leq n \leq 8$。

示例：

- 示例 1：

```py
输入：n = 3
输出：["((()))","(()())","(())()","()(())","()()()"]
```

- 示例 2：

```py
输入：n = 1
输出：["()"]
```

解题思路

思路 1：回溯算法

为了生成的括号组合是有效的，回溯的时候，使用一个标记变量 `symbol` 来表示是否当前组合是否成对匹配。

如果在当前组合中增加一个 `(` ，则令 `symbol` 加 `1` ，如果增加一个 `)` ，则令 `symbol` 减 `1` 。

显然只有在 `symbol < n` 的时候，才能增加 `(` ，在 `symbol > 0` 的时候，才能增加 `)` 。

如果最终生成 $2 \times n$ 的括号组合，并且 `symbol == 0` ，则说明当前组合是有效的，将其加入到最终答案数组中。

下面我们根据回溯算法三步走，写出对应的回溯算法。

1. **明确所有选择**：$2 \times n$ 的括号组合中的每个位置，都可以从 `(` 或者 `)` 中选出。并且，只有在 `symbol < n` 的时候，才能选择 `(` ，在 `symbol > 0` 的时候，才能选择 `)` 。
2. **明确终止条件**：
   - 当遍历到决策树的叶子节点时，就终止了。即当前路径搜索到末尾时，递归终止。
3. **将决策树和终止条件翻译成代码：**
   1. 定义回溯函数：
      - `backtracking(symbol, index)`：函数的传入参数是 `symbol` （用于表示是否当前组合是否成对匹配），`index` （当前元素下标），全局变量是 `parentheses` （用于保存所有有效的括号组合），`parenthesis` （当前括号组合），。
      - `backtracking(symbol, index)` 函数代表的含义是：递归根据 `symbol` ，在 `(` 和 `)` 中选择第 `index` 个元素。
   2. 书写回溯函数主体（给出选择元素、递归搜索、撤销选择部分）。
      - 从当前正在考虑元素，到第 $2 \times n$ 个元素为止，枚举出所有可选的元素。对于每一个可选元素：
        - 约束条件：`symbol < n` 或者 `symbol > 0` 。
        - 选择元素：将其添加到当前括号组合 `parenthesis` 中。

- 递归搜索：在选择该元素的情况下，继续递归选择剩下元素。
- 撤销选择：将该元素从当前括号组合 `parenthesis` 中移除。

```py
if symbol < n:
    parenthesis.append('(')
    backtrack(symbol + 1, index + 1)
    parenthesis.pop()
if symbol > 0:
    parenthesis.append(')')
    backtrack(symbol - 1, index + 1)
    parenthesis.pop()
```

3. 明确递归终止条件（给出递归终止条件，以及递归终止时的处理方法）。
   - 当遍历到决策树的叶子节点时，就终止了。也就是当 `index == 2 * n` 时，递归停止。
   - 并且在 `symbol == 0` 时，当前组合才是有效的，此时将其加入到最终答案数组中。

思路 1：代码

```py
class Solution:
    def generateParenthesis(self     int) -> List[str]:
        parentheses = []            # 存放所有括号组合
        parenthesis = []            # 存放当前括号组合
        def backtrack(symbol, index):
            if n * 2 == index:
                if symbol == 0:
                    parentheses.append("".join(parenthesis))
            else:
                if symbol < n:
                    parenthesis.append('(')
                    backtrack(symbol + 1, index + 1)
                    parenthesis.pop()
                if symbol > 0:
                    parenthesis.append(')')
                    backtrack(symbol - 1, index + 1)
                    parenthesis.pop()
        backtrack(0, 0)
        return parentheses
```

思路 1：复杂度分析

- **时间复杂度**：$O(\frac{2^{2\times n}}{\sqrt{n}})$，其中 n 为生成括号的对数。
- **空间复杂度**：$O(n)$。

参考资料

- 【题解】22. 括号生成 - 力扣（Leetcode）