0131. 分割回文串

ITCharge 大约 1 分钟

- 标签：字符串、动态规划、回溯
- 难度：中等

题目链接

- 0131. 分割回文串 - 力扣

题目大意

给定一个字符串 `s` ，将 `s` 分割成一些子串，保证每个子串都是「回文串」。返回 `s` 所有可能的分割方案。

解题思路

回溯算法，建立两个数组 res、path。res 用于存放所有满足题意的组合，path 用于存放当前满足题意的一个组合。

在回溯的时候判断当前子串是否为回文串，如果不是则跳过，如果是则继续向下一层遍历。

定义判断是否为回文串的方法和回溯方法，从 `start_index = 0` 的位置开始回溯。

- 如果 `start_index >= len(s)` ，则将 path 中的元素加入到 res 数组中。
- 然后对 `[start_index, len(s) - 1]` 范围内的子串进行遍历取值。
  - 如果字符串 `s` 在范围 `[start_index, i]` 所代表的子串是回文串，则将其加入 path 数组。
  - 递归遍历 `[i + 1, len(s) - 1]` 范围上的子串。
  - 然后将遍历的范围 `[start_index, i]` 所代表的子串进行回退。
- 最终返回 res 数组。

代码

```py
class Solution:
    res = []
    path = []
    def backtrack(self, s: str, start_index: int):
        if start_index >= len(s):
            self.res.append(self.path[:])
            return
        for i in range(start_index, len(s)):
            if self.ispalindrome(s, start_index, i):
                self.path.append(s[start_index: i+1])
                self.backtrack(s, i + 1)
                self.path.pop()

    def ispalindrome(self, s: str, start: int, end: int):
        i, j = start, end
        while i < j:
            if s[i] != s[j]:
                return False
            i += 1
            j -= 1
        return True

    def partition(self, s: str) -> List[List[str]]:
        self.res.clear()
        self.path.clear()
        self.backtrack(s, 0)
        return self.res
```