131. Palindrome Partitioning

Created	@September 21, 2022 3:40 PM
Difficulty	Medium
LC Url	https://leetcode.com/problems/palindrome-partitioning/
Importance	
Tag	Backtrack DFS
Video	https://www.youtube.com/watch?v=3jvWodd7ht0

Given a string `s`, partition `s` such that every substring of the partition is a **palindrome**. Return all possible palindrome partitioning of `s`.

A **palindrome** string is a string that reads the same backward as forward.

Example 1:

```
Input: s = "aab"
Output: [["a","a","b"],["aa","b"]]
```

Example 2:

```
Input: s = "a"
Output: [["a"]]
```

Constraints:

- `1 <= s.length <= 16`
- `s` contains only lowercase English letters.

Solution

Backtrack

```
class Solution:
    def partition(self, s: str) -> List[List[str]]:
        res = []
        part = []

        def dfs(i):
            if i >= len(s):
                res.append(part.copy())
                return

            for j in range(i, len(s)):
                if self.isPali(s, i, j):
                    part.append(s[i:j + 1])
                    dfs(j + 1)
                    part.pop()
        dfs(0)
        return res

    def isPali(self, s, left, right):
        while left < right:
            if s[left] != s[right]:
                return False
            left, right = left + 1, right - 1
        return True
```

复杂度分析

- 时间复杂度：$O(n \cdot 2^n)$，其中 n 是字符串 s 的长度，与方法一相同。
- 空间复杂度：$O(n^2)$，与方法一相同。