131. Palindrome Partitioning

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"]]
```

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



    def partitionUtil(self,s,ssf,res):
        if len(s)==0:
            res.append(ssf[:])
            return


        for i in range(len(s)):
            subString = s[0:i+1]
            remain = s[i+1:]
            if subString == subString[::-1]:
                self.partitionUtil(remain,ssf+[subString],res)
```