LeetCode 第 131 号问题：分割回文串

> *本文首发于公众号「图解面试算法」，是 图解 LeetCode 系列文章之一。*
>
> *同步博客：https://www.algomooc.com*

题目来源于 LeetCode 上第 131 号问题：分割回文串。题目难度为 Medium，目前通过率为 45.8% 。

题目描述

给定一个字符串 *s*，将 *s* 分割成一些子串，使每个子串都是回文串。

返回 *s* 所有可能的分割方案。

示例:

```
输入: "aab"
输出:
[
  ["aa","b"],
  ["a","a","b"]
]
```

题目解析

首先，对于一个字符串的分割，肯定需要将所有分割情况都遍历完毕才能判断是不是回文数。不能因为 **abba** 是回文串，就认为它的所有子串都是回文的。

既然需要将所有的分割方法都找出来，那么肯定需要用到DFS（深度优先搜索）或者BFS（广度优先搜索）。

在分割的过程中对于每一个字符串而言都可以分为两部分：左边一个回文串加右边一个子串，比如 "abc" 可分为 "a" + "bc" 。 然后对"bc"分割仍然是同样的方法，分为"b"+"c"。

在处理的时候去优先寻找更短的回文串，然后回溯找稍微长一些的回文串分割方法，不断回溯，分割，直到找到所有的分割方法。

举个🌰：分割"aac"。

1. 分割为 a + ac
2. 分割为 a + a + c，分割后，得到一组结果，再回溯到 a + ac
3. a + ac 中 ac 不是回文串，继续回溯，回溯到 aac
4. 分割为稍长的回文串，分割为 aa + c 分割完成得到一组结果，再回溯到 aac
5. aac 不是回文串，搜索结束

动画描述

代码实现

```
class Solution {
    List<List<String>> res = new ArrayList<>();
```

```
    public List<List<String>> partition(String s) {
        if(s==null||s.length()==0)
            return res;
        dfs(s,new ArrayList<String>(),0);
        return res;
    }

    public void dfs(String s,List<String> remain,int left){
        if(left==s.length()){  //判断终止条件
            res.add(new ArrayList<String>(remain));  //添加到结果中
            return;
        }
        for(int right=left;right<s.length();right++){  //从left开始，依次判断left->right是不是回文串
            if(isPalindroom(s,left,right)){  //判断是否是回文串
                remain.add(s.substring(left,right+1));    //添加到当前回文串到list中
                dfs(s,remain,right+1);  //从right+1开始继续递归，寻找回文串
                remain.remove(remain.size()-1);  //回溯，从而寻找更长的回文串
            }
        }
    }
    /**
    * 判断是否是回文串
    */
    public boolean isPalindroom(String s,int left,int right){
        while(left<right&&s.charAt(left)==s.charAt(right)){
            left++;
            right--;
        }
        return left>=right;
    }
}
```