

917 · Palindrome Permutation II

Description

Given a string `s`, return all the palindromic permutations (**without duplicates**) of it. Return an empty list if no palindromic permutation could be form.

Example

Example1

Input: `s = "aabb"`

Output: `["abba", "baab"]`

Example2

Input: `"abc"`

Output: `[]`

```
class Solution:
    """
    @param s: the given string
    @return: all the palindromic permutations (without duplicates) of it
    """
    def generatePalindromes(self, s):
        # write your code here
        freqmap = {}
        for ele in s:
            if ele in freqmap:
                freqmap[ele] = freqmap[ele]+1
            else:
                freqmap[ele]=1
        oddChar = ''
        odd = 0
        count=0
        for key in freqmap.keys():
            if freqmap[key]%2!=0:
                oddChar = key
                freqmap[key] = freqmap[key]//2
                if freqmap[key]!=0:
                    count = freqmap[key]+count
                odd = odd+1
            else:
```

```

        freqmap[key] = freqmap[key]//2
        count = freqmap[key]+count
    if odd>1:
        return []
    res =[]
    self.permuteUtil(freqmap, '', oddChar, res, count)
    return res

def permuteUtil(self, freqmap, ssf, oddChar, res, count):
    if len(ssf)==count:
        temp = ssf[::-1]
        result = ssf+oddChar+temp
        res.append(result)
        return
    for key in freqmap.keys():
        if freqmap[key]>0:
            freqmap[key]= freqmap[key]-1
            self.permuteUtil(freqmap, ssf+key, oddChar, res, count)
            freqmap[key] = freqmap[key] + 1

```