17. Letter Combinations of a Phone Number

Created	@July 15, 2020 10:24 AM
Difficulty	Medium
LC Url	https://leetcode.com/problems/letter-combinations-of-a-phone-number/
Importance	
Tag	Backtrack
Video	

Given a string containing digits from `2-9` inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

Example 1:

```
Input: digits = "23"
Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]
```

Example 2:

```
Input: digits = ""
Output: []
```

Example 3:

```
Input: digits = "2"
Output: ["a","b","c"]
```

Constraints:

- `0 <= digits.length <= 4`
- `digits[i]` is a digit in the range `['2', '9']`.

Solution

方法一：回溯

首先使用哈希表存储每个数字对应的所有可能的字母，然后进行回溯操作。

回溯过程中维护一个字符串，表示已有的字母排列（如果未遍历完电话号码的所有数字，则已有的字母排列是不完整的）。该字符串初始为空。每次取电话号码的一位数字，从哈希表中获得该数字对应的所有可能的字母，并将其中的一个字母插入到已有的字母排列后面，然后继续处理电话号码的后一位数字，直到处理完电话号码中的所有数字，即得到一个完整的字母排列。然后进行回退操作，遍历其余的字母排列。

回溯算法用于寻找所有的可行解，如果发现一个解不可行，则会舍弃不可行的解。在这道题中，由于每个数字对应的每个字母都可能进入字母组合，因此不存在不可行的解，直接穷举所有的解即可。

```
class Solution:
    phoneMap = {
        "2": "abc",
        "3": "def",
        "4": "ghi",
        "5": "jkl",
        "6": "mno",
        "7": "pqrs",
        "8": "tuv",
        "9": "wxyz",
    }

    def letterCombinations(self, digits: str) -> List[str]:
        if not digits:
            return []

        subset = []
        res = []
        self.backtrack(digits, 0, subset, res)

        return res

    def backtrack(self, digits, index, subset, res):
        if index == len(digits):
            res.append(''.join(subset))
        else:
            digit = digits[index]
            for c in self.phoneMap[digit]:
                subset.append(c)
                self.backtrack(digits, index + 1, subset, res)
                subset.pop()
```

```
class Solution:
    def letterCombinations(self, digits: str) -> List[str]:
        if not digits:
            return []

        phoneMap = {
            "2": "abc",
            "3": "def",
            "4": "ghi",
            "5": "jkl",
            "6": "mno",
            "7": "pqrs",
            "8": "tuv",
            "9": "wxyz",
        }

        def backtrack(index):
            if index == len(digits):
                combinations.append(''.join(combination))
            else:
                digit = digits[index]
                for c in phoneMap[digit]:
                    combination.append(c)
                    backtrack(index + 1)
                    combination.pop()

        combination = []
        combinations = []
        backtrack(0)
```

```
        return combinations

        # 链接：https://leetcode.cn/problems/letter-combinations-of-a-phone-number/solution/dian-hua-hao-ma-de-zi-mu-zu-he-by-leetcode-solut
```

复杂度分析

- 时间复杂度：$O(3^m \times 4^n)$，其中 m 是输入中对应 3 个字母的数字个数（包括数字 2、3、4、5、6、8），n 是输入中对应 4 个字母的数字个数（包括数字 7、9），$m+n$ 是输入数字的总个数。当输入包含 m 个对应 3 个字母的数字和 n 个对应 4 个字母的数字时，不同的字母组合一共有 $3^m \times 4^n$ 种，需要遍历每一种字母组合。
- 空间复杂度：$O(m+n)$，其中 m 是输入中对应 3 个字母的数字个数，n 是输入中对应 4 个字母的数字个数，$m+n$ 是输入数字的总个数。除了返回值以外，空间复杂度主要取决于哈希表以及回溯过程中的递归调用层数，哈希表的大小与输入无关，可以看成常数，递归调用层数最大为 $m+n$。