21 电话号码的字母组合

更新时间：2019-09-03 09:34:44

> 学习这件事不在乎有没有人教你，最重要的是在于你自己有没有觉悟和恒心。
>
> —— 法布尔

刷题内容

> 难度: Medium
>
> 原题链接：https://leetcode.com/problems/letter-combinations-of-a-phone-number/description/

内容描述

给定一个仅包含数字 2-9 的字符串，返回所有它能表示的字母组合。

给出数字到字母的映射如下（与电话按键相同）。注意 1 不对应任何字母。

示例：
输入："23"
输出：["ad", "ae", "af", "bd", "be", "bf", "cd", "ce", "cf"].

说明:
尽管上面的答案是按字典序排列的，但是你可以任意选择答案输出的顺序。

> 其实就是手机键盘的 9 键输入

输入 “23”，返回 abcdef 所有能表示的字母组合。

解题方案

思路 1：

这道题我们首先将数字与其对应字母放到一个字典中去：

```
lookup = {
    '2':['a','b','c'],
    '3':['d','e','f'],
    '4':['g','h','i'],
    '5':['j','k','l'],
    '6':['m','n','o'],
    '7':['p','q','r','s'],
    '8':['t','u','v'],
    '9':['w','x','y','z']
}
```

定义一个数组 res = [] 来接收最后结果，如果没有参数 digits 或 digits 为 0，直接返回 [] 。

helper() 方法接受两个参数 s 和 digits ， s 用来记录结果。cur_digit 为 digit 第一个字符，根据cur_digit 在 lookup 中找到对应的值。

遍历 lookup[cur_digit] ， helper() 方法递归调用自身，将每一种可能都尝试一遍，最后如果 digit 长度为 0 ，将 s 放入最后结果 res 中去。

下面来看具体代码实现：

Python beats 95.07%

```python
class Solution(object):
    def letterCombinations(self, digits):
        """
        :type digits: str
        :rtype: List[str]
        """
        # 将数字与其对应字母放到一个字典中去
        lookup = {
            '2':['a','b','c'],
            '3':['d','e','f'],
            '4':['g','h','i'],
            '5':['j','k','l'],
            '6':['m','n','o'],
            '7':['p','q','r','s'],
            '8':['t','u','v'],
            '9':['w','x','y','z']
        }
        res = []

        def helper(s, digits):
            # 如果digit长度为0，将s放入最后结果res中去。
            if len(digits) == 0:
                res.append(s)
            else:
                cur_digit = digits[0]
                for char in lookup[cur_digit]:
                    helper(s+char, digits[1:])

        if not digits or len(digits) == 0:
            return res
        helper('', digits)
        return res
```

Java beats 70.86%

```java
class Solution {
    public List<String> letterCombinations(String digits) {
        String[] lookup = {"", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};
        if (digits == null || digits.length() == 0) {
            return new ArrayList<>();
        }
        // 用队列来保存满足要求的字符串
        Queue<String> queue = new LinkedList();
        queue.add("");
        for (int i = 0; i < digits.length(); i++) {
            // - '0' 为了转化为整型
            String letter = lookup[digits.charAt(i) - '0'];
            int n = queue.size();
            for (int j = 0; j < n; j++) {
                // 弹出队列头
                String s = queue.poll();
                for (int k = 0; k < letter.length(); k++) {
                    // 每次入队满足要求的字符串
                    queue.add(s + letter.charAt(k));
                }
            }
        }
        List<String> ret = new ArrayList<>();
        // 将满足条件的字符串逐步出队
        while (!queue.isEmpty()) {
            ret.add(queue.poll());
        }
        return ret;
    }
}
```

c++ beats 100%

```
const string letters[] = {"", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};
class Solution {
public:
    vector<string> letterCombinations(string digits) {
        if (digits == "") {
            return vector<string>();
        }
        //n个字母的字符串，要再加上一个字母变成n+1个字符的字符串
        //而这个字母有k种选择，这里的数据结构选用队列
        //每次需要加上一个字母时，一个个出列，丢进队列k个处理后的字符串
        queue<string> Q;
        Q.push("");
        for (int i = 0;i < digits.size();i++) {
            string letter = letters[digits[i] - '0'];
            int n = Q.size();
            for (int j = 0;j < n;j++) {
                string s = Q.front();
                Q.pop();
                for (int k = 0;k < letter.size();k++) {
                    Q.push(s + letter[k]);
                }
            }
        }
        vector<string> ret;
        while (!Q.empty()) {
            ret.push_back(Q.front());
            Q.pop();
        }
        return ret;
    }
};
```

go beats 100%

```go
import (
    "container/list"
)

func letterCombinations(digits string) []string {
    letters := []string{"", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};
    if len(digits) == 0 {
        return make([]string, 0)
    }
    list := list.New()
    list.PushBack("")
    //n个字母的字符串，要再加上一个字母变成n+1个字符的字符串
    //而这个字母有k种选择，这里的数据结构选用队列
    //每次需要加上一个字母时，一个个出列，丢进队列k个处理后的字符串
    for _, digit := range(digits) {
        letter := letters[digit - '0']
        n := list.Len()
        for j := 0;j < n;j++ {
            s := list.Front()
            list.Remove(s)
            for _, c := range(letter) {
                list.PushBack(s.Value.(string) + string(c))
            }
        }
    }
    ret := make([]string, 0)
    for {
        ret = append(ret, list.Front().Value.(string))
        list.Remove(list.Front())
        if list.Len() == 0 {
            break
        }
    }
    return ret
}
```

小结

- 这道题最难的是数字和字母的对应，拿出手机打字键盘看看哈哈哈；
- 剩下的就是一个递归调用，自减长度直到为0放入结果中，最后返回即可。

}

20 最接近的三数之和

22 删除链表的倒数第 n 个节点