1. 题目
2. 题干

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。

1. 示例

**输入：**"23"

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

1. 题解
2. 思路

首先，从题干中可以看出，这是一个要枚举所有的可能性的问题，这里我们会联想到深度优先搜索，不到最后的叶子节点，搜索不会停下来。当到达最后的叶子节点后，若满足条件，则记录该次搜索的全路径；若不满足条件，则退回到最邻近的一个节点，继续搜索另一个叶子节点。通过这种方式不断地搜索，退回，直到搜索完所有的可能结果。

对于这个题目，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组合的后面，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，得到一个完整的字母排列。在这之后，从最后一个字母开始回溯，遍历其余的字母组合。

1. 代码实现
2. C++

class Solution {

public:

    vector<string> letterCombinations(string digits) {

        vector<string> combinations;

        if (digits.empty())

            return combinations;

        unordered\_map<char, string> phoneMap{

            {'2', "abc"},

            {'3', "def"},

            {'4', "ghi"},

            {'5', "jkl"},

            {'6', "mno"},

            {'7', "pqrs"},

            {'8', "tuv"},

            {'9', "wxyz"},

        };

        string combination;

        backtrack(combinations, phoneMap, digits, 0, combination);

        return combinations;

    }

    void backtrack(vector<string>& combinations, const unordered\_map<char, string>& phoneMap, const string& digits, int index, string& combination) {

        if (index == digits.length())

            combinations.push\_back(combination);

        else {

            char digit = digits[index];

            const string& letters = phoneMap.at(digit);

            for (const char& letter : letters) {

                combination.push\_back(letter);

                backtrack(combinations , phoneMap, digits, index + 1, combination);

                combination.pop\_back();

            }

        }

    }

};

1. Java

import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
class Solution17Phone {  
 public List<String> letterCombinations(String digits){  
 //最终结果集  
 List<String> combinations = new ArrayList<String>();  
 if(digits.length()==0)  
 return combinations;  
  
 //电话号码map  
 HashMap<Character, String> phoneMap = new HashMap<Character, String>() {{  
 put('2', "abc");  
 put('3', "def");  
 put('4', "ghi");  
 put('5', "jkl");  
 put('6', "mno");  
 put('7', "pqrs");  
 put('8', "tuv");  
 put('9', "wxyz");  
 }};  
 //回溯函数  
 backtrack(combinations,phoneMap,digits, 0, new StringBuffer() );  
 return combinations;  
 }  
  
 //回溯函数  
 public void backtrack(List<String> combinations, Map<Character,String> phoneMap, String digits, int index, StringBuffer combination){  
 //枚举了所有的数字,添加到输出结果集中  
 if(index==digits.length())  
 combinations.add(combination.toString());  
 //不然，继续枚举  
 else{  
 char digit = digits.charAt(index);  
 String letters = phoneMap.get(digit);  
 int lettersCount = letters.length();  
 //对数字对应的所有字母一一遍历，添加到之前的字符串中，进行回溯，再删除该字符  
 for(int i = 0; i < lettersCount; ++i){  
 combination.append(letters.charAt(i));  
 backtrack(combinations, phoneMap, digits, index+1, combination );  
 combination.deleteCharAt(index);  
 }  
 }  
 }  
}