**河南开封科技传媒学院实验报告**

••••••••••••••••••••••••••••••••• 密 ••••••••••••••••••••••••••••••••• 封 ••••••••••••••••••••••••••••••••• 线 •••••••••••••••••••••••••••••••••

**2023-2024学年第二学期**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名： | 刘东 | | 学号： | 2136101045 | 专业： | 2021软件工程 | |
| 所在学院： | | 信息工程学院 | | | 指导老师： | | 翟俊武 |
| 实验时间： | | 2024.4.10 | | | 实验地点： | | 创客中心 |
| 课程名称： | | 算法分析与设计 | | | | | |
| 实验题目： | | 回溯法2 | | | | | |
| 实验目的：   1. 回顾基本算法原理； 2. 提升学生分析问题的能力，以及巩固在求解组合问题时回溯法的主要思想； 3. 提高学生的编程能力。 | | | | | | | |
| 1. 实验内容：   LeetCode习题：17、LCR 082 | | | | | | | |
| 实验代码及通过率截图：    17题：  class Solution {  public:      vector<string> result;      string path;      void getCombinations(string digits, int number) {          if (number == digits.size()) {              result.push\_back(path);              return;          }          int digit = digits[number] - '0';          string strLetter = letterMap[digit];          for (int i = 0; i < strLetter.size(); i++) {              path.push\_back(strLetter[i]);              getCombinations(digits, number + 1);              path.pop\_back();          }      }      vector<string> letterCombinations(string digits) {          if (digits.size() == 0) return result;          getCombinations(digits, 0);          return result;      }  private :      const string letterMap[10] = {          "", // 0          "", // 1          "abc", // 2          "def", // 3          "ghi", // 4          "jkl", // 5          "mno", // 6          "pqrs", // 7          "tuv", // 8          "wxyz", // 9      };  };    LCR 082  class Solution {  public:      /\*\*      \*思路是通过数组排序进行dfs，其中进行剪枝      \*/      vector<vector<int>> res;  // 结果      vector<int> temp; // 每次存储的数组，用于深搜数组添加      vector<vector<int>> combinationSum2(vector<int>& candidates, int target) {          sort(candidates.begin(), candidates.end());          findTargetByDeep(candidates, target, 0);          return res;      }        void findTargetByDeep(vector<int>& candidates, int target, int k) {          if (target == 0) {              res.push\_back(temp);              return;          }          for (int i = k; i < candidates.size(); i ++) {              if (candidates[i] > target) break; // 剪枝              if ( i > k && candidates[i] == candidates[i - 1]) continue; // 遇见相同数就跳过                temp.push\_back(candidates[i]);              target -= candidates[i];              findTargetByDeep(candidates, target, i + 1);              target += candidates[i];              temp.pop\_back();          }      }  }; | | | | | | | |
| 实验心得： | | | | | | | |