**《算法设计与分析》周练记录表-第5周**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **班学号** | **姓名** | **OJ账号** | **练习平台** | **完成题目数量** | **题号** |
| 11117233 | 徐鸿飞 | X-cool | LeetCode | 8 | 93，51，52，401，77，17，78，90 |
| Xhofe | LintCode |  |  |
|  | 牛客网 |  |  |
|  | CCF-CSP |  |  |

LeetCode：https://leetcode.com/ 或https://leetcode-cn.com/

LintCode：https://www.lintcode.com/

牛客网（剑指Offer）：https://www.nowcoder.com/ta/coding-interviews?page=1

CCF-CSP：http://www.cspro.org/lead/application/ccf/login.jsp

\*注：本课程作业报告每周提交一次，请所有同学每周日晚21:00前交给各班学习委员，由学习委员统计并上传至微云，课堂上将随机抽取同学讲解自己的练习（5分钟）。

## 一、练习题目及代码

#### 【题目1】LeetCode93

给定一个只包含数字的字符串，复原它并返回所有可能的 IP 地址格式。

**示例:**

**输入:** "25525511135"

**输出:** ["255.255.11.135", "255.255.111.35"]

**【代码】**

**package** LeetCode;

**import** java.util.ArrayList;

**import** java.util.List;

**public class** leetcode93 {

**private** List<String> **res**;

**private** StringBuffer **str**;

**public** List<String> restoreIpAddresses(String s) {

**res**=**new** ArrayList<>();

**int** len=s.length();

**str**=**new** StringBuffer();

ip(0,0,len,s);

**return res**;

}

**private void** ip(**int** i,**int** index,**int** n,String s){

**if** (i==3){

**if** (!is(s.substring(index,s.length()))){

**return**;

}

**str**.append(s.substring(index,s.length()));

**res**.add(**str**.toString());

**str**.delete(**str**.length()-s.length()+index,**str**.length());

**return**;

}

**for** (**int** j=index+1;j<n-(2-i);j++){

**if** (!is(s.substring(index,j))||!(n-j<=3\*(3-i)))**continue**;

**str**.append(s.substring(index,j));

**str**.append(**'.'**);

ip(i+1,j,n,s);

**str**.delete(**str**.length()-j+index-1,**str**.length());

}

}

**private boolean** is(String s){

**if** (s.length()>3)**return false**;

**int** num=Integer.*valueOf*(s);

**if** (num<10)**return** s.length()==1;

**if** (num<100)**return** s.length()==2;

**if** (num<=255)**return** s.length()==3;

**return false**;

}

}

**【提交记录截图】**



**【遇到的问题及解决办法】**

递归分为4段，每次都检测每段是否符合ip的规则。

问题一：分段问题，被index整疯。。慢慢调试修改

问题二：Integer.*valueOf*(String s)中s会超范围。。

**【对比学习心得】**

代码有点繁琐。。

还可以剪枝。在检测s是否符合的时候只要s的长度超过3，就不在继续。

#### 【题目2】LeetCode51

*n*皇后问题研究的是如何将 *n* 个皇后放置在 *n*×*n* 的棋盘上，并且使皇后彼此之间不能相互攻击。



上图为 8 皇后问题的一种解法。

给定一个整数 *n*，返回所有不同的 *n*皇后问题的解决方案。

每一种解法包含一个明确的 *n* 皇后问题的棋子放置方案，该方案中 'Q' 和 '.' 分别代表了皇后和空位。

**示例:**

**输入:** 4

**输出:** [

[".Q..", // 解法 1

"...Q",

"Q...",

"..Q."],

["..Q.", // 解法 2

"Q...",

"...Q",

".Q.."]

]

**解释:** 4 皇后问题存在两个不同的解法。

**【代码】**

**package** LeetCode;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**public class** leetcode51 {

**private** List<List<String>> **res**;

**private** List<String> **map**;

**private boolean**[] **col**;

**private boolean**[] **diagonal1**;

**private boolean**[] **diagonal2**;

**public** List<List<String>> solveNQueens(**int** n) {

**res**=**new** ArrayList<>();

StringBuffer tmp=**new** StringBuffer();

**for** (**int** i=0;i<n;i++){

tmp.append(**'.'**);

}

**map**=**new** ArrayList<>();

**for** (**int** i=0;i<n;++i){

**map**.add(tmp.toString());

}

**col**=**new boolean**[n];

**diagonal1**=**new boolean**[2\*n-1];

**diagonal2**=**new boolean**[2\*n-1];

Arrays.*fill*(**col**,**false**);

Arrays.*fill*(**diagonal1**,**false**);

Arrays.*fill*(**diagonal2**,**false**);

nqueens(n,0);

**return res**;

}

**private boolean** is(**int** x,**int** y,**int** n){

**return** !**col**[x]&&!**diagonal1**[x+y]&&!**diagonal2**[x-y+n-1];

}

**private void** updatemap(**int** x,**int** y,**int** n,**boolean** is){

**col**[x]=is;**diagonal1**[x+y]=is;**diagonal2**[x-y+n-1]=is;

StringBuffer tmp=**new** StringBuffer(**map**.get(y));

**char** a=is?**'Q'**:**'.'**;

tmp.setCharAt(x,a);

**map**.set(y,tmp.toString());

}

**private void** nqueens(**int** n,**int** y){

**if** (y==n){

List<String> tmp=**new** ArrayList<>();

tmp.addAll(**map**);

**res**.add(tmp);

**return**;

}

**for** (**int** x=0;x<n;++x){

**if** (!is(x,y,n))**continue**;

updatemap(x,y,n,**true**);

nqueens(n,y+1);

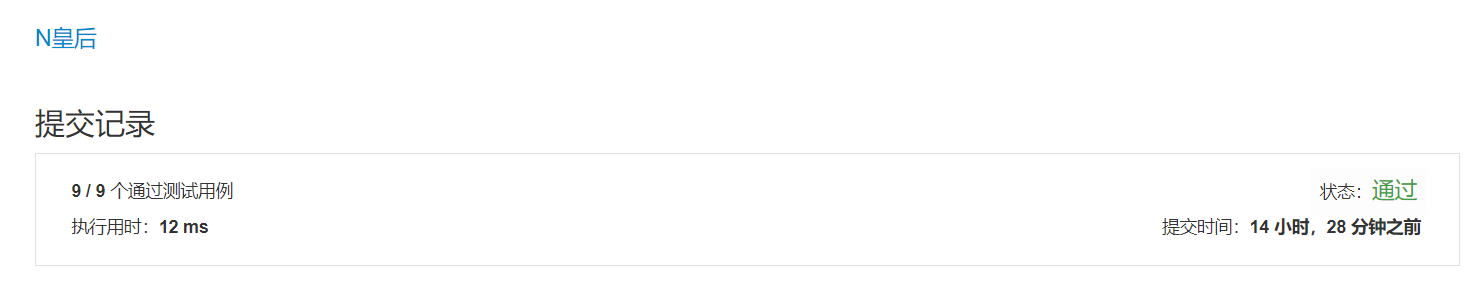
updatemap(x,y,n,**false**);

}

}

}

**【提交记录截图】**



**【遇到的问题及解决办法】**

按照老师上课说的思路依次在每一行下皇后，然后用三个数组纪录不能再放置皇后的位置，在每次防止皇后后更新map，递归到完全防止后将解加入结果中。

问题：List的拷贝问题，浅拷贝导致所有结果一样。String没有浅拷贝。

**【对比学习心得】**

#### 【题目3】LeetCode52

*n*皇后问题研究的是如何将 *n* 个皇后放置在 *n*×*n* 的棋盘上，并且使皇后彼此之间不能相互攻击。



上图为 8 皇后问题的一种解法。

给定一个整数 *n*，返回 *n* 皇后不同的解决方案的数量。

**示例:**

**输入:** 4

**输出:** 2

**解释:** 4 皇后问题存在如下两个不同的解法。

[

 [".Q..",  // 解法 1

  "...Q",

  "Q...",

  "..Q."],

 ["..Q.",  // 解法 2

  "Q...",

  "...Q",

  ".Q.."]

]

**【代码】**

**package** LeetCode;

**import** java.util.Arrays;

**public class** leetcode52 {

**private int res**;

**private boolean**[] **col**;

**private boolean**[] **diagonal1**;

**private boolean**[] **diagonal2**;

**public int** totalNQueens(**int** n) {

**res**=0;

**col**=**new boolean**[n];

**diagonal1**=**new boolean**[2\*n-1];

**diagonal2**=**new boolean**[2\*n-1];

Arrays.*fill*(**col**,**false**);

Arrays.*fill*(**diagonal1**,**false**);

Arrays.*fill*(**diagonal2**,**false**);

nqueens(n,0);

**return res**;

}

**private boolean** is(**int** x,**int** y,**int** n){

**return** !**col**[x]&&!**diagonal1**[x+y]&&!**diagonal2**[x-y+n-1];

}

**private void** updatemap(**int** x,**int** y,**int** n,**boolean** is){

**col**[x]=is;**diagonal1**[x+y]=is;**diagonal2**[x-y+n-1]=is;

}

**private void** nqueens(**int** n,**int** y){

**if** (y==n){

**res**++;

}

**for** (**int** x=0;x<n;++x){

**if** (!is(x,y,n))**continue**;

updatemap(x,y,n,**true**);

nqueens(n,y+1);

updatemap(x,y,n,**false**);

}

}

}

**【提交记录截图】**



**【遇到的问题及解决办法】**

在上一题的基础上稍微修改一下，去掉了纪录map，直接纪录解的个数就行了。

**【对比学习心得】**

#### 【题目4】LeetCode401

二进制手表顶部有 4 个 LED 代表**小时（0-11）**，底部的 6 个 LED 代表**分钟（0-59）**。

每个 LED 代表一个 0 或 1，最低位在右侧。



例如，上面的二进制手表读取 “3:25”。

给定一个非负整数 *n*代表当前 LED 亮着的数量，返回所有可能的时间。

**案例:**

输入: n = 1

返回: ["1:00", "2:00", "4:00", "8:00", "0:01", "0:02", "0:04", "0:08", "0:16", "0:32"]

**注意事项:**

* 输出的顺序没有要求。
* 小时不会以零开头，比如 “01:00” 是不允许的，应为 “1:00”。
* 分钟必须由两位数组成，可能会以零开头，比如 “10:2” 是无效的，应为 “10:02”。

**【代码】**

**package** LeetCode;

**import** java.util.ArrayList;

**import** java.util.List;

**public class** leetcode401 {

**private** List<String> **res**;

**private int**[] **light**;*//纪录每个灯是否亮*

**public** List<String> readBinaryWatch(**int** num) {

**res**=**new** ArrayList<String>();

**light**=**new int**[10];

**return res**;

}

**private boolean** is(){

**return** 1\***light**[0]+2\***light**[1]+4\***light**[2]+8\***light**[3]<12&&1\***light**[4]+2\***light**[5]+4\***light**[6]+8\***light**[7]+16\***light**[8]+32\***light**[9]<60;

}

**private void** watch(**int** i,**int** num){

**if** (num==0){

**int** h=1\***light**[0]+2\***light**[1]+4\***light**[2]+8\***light**[3];

**int** m=1\***light**[4]+2\***light**[5]+4\***light**[6]+8\***light**[7]+16\***light**[8]+32\***light**[9];

**if** (m>9)**res**.add(h+**":"**+m);

**else res**.add(h+**":0"**+m);

**return**;

}

**if** (i>9)**return**;

**light**[i]=1;

**if** (is()){

watch(i+1,num-1);

}

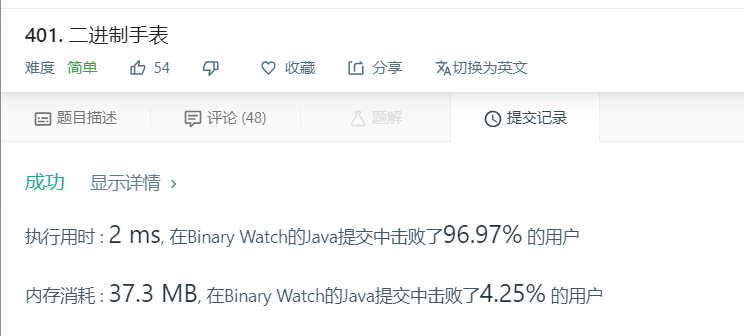
**light**[i]=0;

watch(i+1,num);

}

}

**【提交记录截图】**



**【遇到的问题及解决办法】**

Light[10]纪录十个灯的情况，递归遍历每种情况，符合情况则加入结果。

**【对比学习心得】**

#### 【题目5】LeetCode77

给定两个整数 *n* 和 *k*，返回 1 ... *n*中所有可能的 *k* 个数的组合。

**示例:**

**输入:** n = 4, k = 2

**输出:**

[

[2,4],

[3,4],

[2,3],

[1,2],

[1,3],

[1,4],

]

**【代码】**

**package** LeetCode;

**import** java.util.ArrayList;

**import** java.util.List;

**public class** leetcode77 {

**private int**[] **num**;

**private** List<List<Integer>> **res**;

**public** List<List<Integer>> combine(**int** n, **int** k) {

**num**=**new int**[k];

**res**=**new** ArrayList<>();

Backtracking(0,k,k);

**return res**;

}

**private void** Backtracking(**int** i,**int** k,**int** n){

**if** (k==0){

insert(n);

**return**;

}

**if** (i>n-1)**return**;

**num**[i]=1;

Backtracking(i+1,k-1,n);

**num**[i]=0;

Backtracking(i+1,k,n);

}

**private void** insert(**int** n){

List<Integer> tmp=**new** ArrayList<>();

**for** (**int** i=0;i<n;i++){

**if** (**num**[i]==1){

tmp.add(Integer.*valueOf*(i));

}

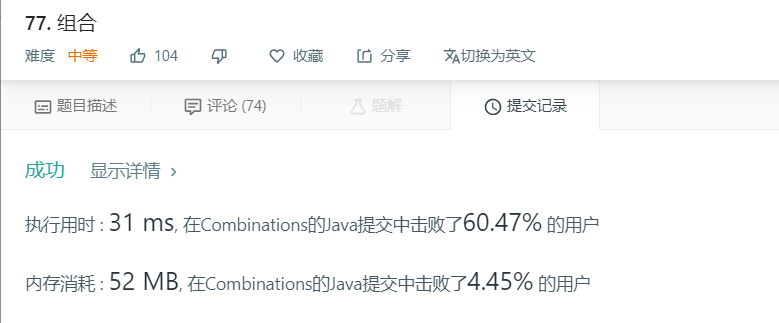
}

**res**.add(tmp);

}

}

**【提交记录截图】**



**【遇到的问题及解决办法】**

与上一题思想类似，纪录1…n选择的情况，依次递归第i个选与不选，选择k个之后，将结果加入解。

**【对比学习心得】**

#### 【题目6】LeetCode17

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。

**【代码】**

**package** LeetCode;

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.List;

**import** java.util.Map;

**public class** leetcode17 {

**private** List<String> **res**;

**private** Map<String,String> **map**;

**private** StringBuffer **str**;

**public** List<String> letterCombinations(String digits) {

**res**=**new** ArrayList<>();

**if** (digits.length()==0||digits==**null**)**return res**;

**map**=**new** HashMap<>();

**map**.put(**"2"**,**"abc"**);

**map**.put(**"3"**,**"def"**);

**map**.put(**"4"**,**"ghi"**);

**map**.put(**"5"**,**"jkl"**);

**map**.put(**"6"**,**"mno"**);

**map**.put(**"7"**,**"pqrs"**);

**map**.put(**"8"**,**"tuv"**);

**map**.put(**"9"**,**"wxyz"**);

**int** len=digits.length();

**str** =**new** StringBuffer();

bt(0,digits,len);

**return res**;

}

**private void** bt(**int** i,String digits,**int** n){

**if** (i==n){

**res**.add(**str**.toString());

**return**;

}

String tmp=**map**.get(String.*valueOf*(digits.charAt(i)));

**for** (**int** j=0;j<tmp.length();j++){

**str**.append(tmp.charAt(j));

bt(i+1,digits,n);

**str**.deleteCharAt(**str**.length()-1);

}

}

}

**【提交记录截图】**



**【遇到的问题及解决办法】**

直接套就行。

第一次没有加digits为空情形。

**【对比学习心得】**

#### 【题目7】LeetCode78

给定一组**不含重复元素**的整数数组 *nums*，返回该数组所有可能的子集（幂集）。

**说明：**解集不能包含重复的子集。

**示例:**

**输入:** nums = [1,2,3]

**输出:**

[

[3],

  [1],

  [2],

  [1,2,3],

  [1,3],

  [2,3],

  [1,2],

  []

]

**【代码】**

**private** List<List<Integer>> **res**;

**private boolean**[] **pd**;

**public** List<List<Integer>> subsets(**int**[] nums) {

**res**=**new** ArrayList<>();

**if** (nums.**length**==0||nums==**null**)**return res**;

**pd**=**new boolean**[nums.**length**];

bt(0,nums,nums.**length**);

**return res**;

}

**private void** bt(**int** i,**int**[] nums,**int** n){

**if** (i==n){

List<Integer> tmp=**new** ArrayList<>();

**for** (**int** j=0;j<nums.**length**;j++){

**if** (**pd**[j])tmp.add(Integer.*valueOf*(nums[j]));

}

**res**.add(tmp);

**return**;

}

**pd**[i]=**true**;

bt(i+1,nums,n);

**pd**[i]=**false**;

bt(i+1,nums,n);

}

**【提交记录截图】**



**【遇到的问题及解决办法】**

与leetcode77类似，纪录某个数选与不选，递归到终点将这时的结果加入res。没有发送错误。

**【对比学习心得】**

#### 【题目8】LeetCode90

给定一个可能包含重复元素的整数数组 ***nums***，返回该数组所有可能的子集（幂集）。

**说明：**解集不能包含重复的子集。

**示例:**

**输入:** [1,2,2]

**输出:**

[

[2],

[1],

[1,2,2],

[2,2],

[1,2],

[]

]

**【代码】**

**package** LeetCode;

**import** java.util.\*;

**public class** leetcode90 {

**private** Set<List<Integer>> **res**;

**private boolean**[] **pd**;

**public** List<List<Integer>> subsetsWithDup(**int**[] nums) {

**res**=**new** HashSet<>();

**if** (nums.**length**==0||nums==**null**)**return new** ArrayList<List<Integer>>(**res**);

**pd**=**new boolean**[nums.**length**];

bt(0,nums,nums.**length**);

**return new** ArrayList<List<Integer>>(**res**);

}

**private void** bt(**int** i,**int**[] nums,**int** n){

**if** (i==n){

List<Integer> tmp=**new** ArrayList<>();

**for** (**int** j=0;j<nums.**length**;j++){

**if** (**pd**[j])tmp.add(Integer.*valueOf*(nums[j]));

}

Collections.*sort*(tmp);

**res**.add(tmp);

**return**;

}

**pd**[i]=**true**;

bt(i+1,nums,n);

**pd**[i]=**false**;

bt(i+1,nums,n);

}

}

**【提交记录截图】**



**【遇到的问题及解决办法】**

与上一题一样，把List换成了Set避免加入重复元素，第一次因为未排序导致还是有重复。

可以优化：在刚开始对数组排序，大概减少5-6ms。

**【对比学习心得】**

## 二、本周小结

本周主要写回溯的题，刚开始接触回溯的题特别迷，感觉无从下手（上课没认真听）；认真做题之后，主要有两种：①遍历，某个选项选与不选，两个分治②遍历，加入这个值递归下一层。

不足之处：对剪枝理解不够，时间复杂度较高。

-------------------------------

成绩评定：

教师签名：

批改日期：