日报格式:

主 题：日报

汇报人：杨明

日 期：2018年7月27日 星期五

内 容：

1. 今日学习了的内容：
2. shiro项目连接上了数据库，可以和数据库连接进行身份验证了
3. 日常一道牛客算法题，重建二叉树思想确实不好想。Java基础选择题大概四十道左右
4. Linux基础指令，touch指令、cp指令、mv指令
5. 没有解决的问题：

1）好像没啥没解决的问题呢

1. 明天的学习计划：
2. Shiro项目的身份验证再完善
3. 继续学习Linux指令，尽量提高速率，同时保证效率

3）牛客网算法题日常一道，加选择题若干

4.学习任务完成度： （90）%[注：0% ~ 100%]

5.对自己今天学习的满意度：（8.5）分 [注：1-10分]

6.对今天的学习做一句话的总结：

笔记整理得不及时

附录：

算法1：

题目：*\* 输入某二叉树的前序遍历和中序遍历的结果，请重建出该二叉树。假设输入的前序遍历和中序遍历的结果中都不含重复的数字。  
\* 例如输入前序遍历序列{1,2,4,7,3,5,6,8}和中序遍历序列{4,7,2,1,5,3,8,6}，则重建二叉树并返回。*

{

*\* 思路：  
 \* 前序遍历顺序为根结点->左子树->右子树；中序遍历顺序为左子树->根结点->右子树  
 \* 所以可以根据前序遍历结果在中序遍历结果中找到根结点的位置，从而将中序遍历数组分为两部分，根结点左边为左子树  
 \* 根结点右边为右子树，那么如果将这两边看成两个独立的二叉树，那么它们的重建过程是相同的。  
 \* 所以只需设计一个递归函数来完成，并且计算出每次要传入的两个子树的起点和终点即可。  
 \*  
 \*/*class TreeNode {  
 int val;  
 TreeNode left;  
 TreeNode right;  
 TreeNode(int x) {  
 val = x;  
 }  
}  
  
public class RebuildBinaryTree {  
 public TreeNode reConstructBinaryTree(int[] pre, int[] in) {  
 if (pre.length==0||in.length==0)//如果数组长度为0，直接返回空值  
 return null;  
 //调用递归函数，起点为数组开头到数组结尾  
 TreeNode root = reConstructBinaryTree(pre,0,pre.length-1,in,0,in.length-1);  
 return root;  
 }  
  
 */\*\*  
 \* 重建二叉树函数  
 \** ***@param*** *pre 是每次递归时新的子树对应的前序遍历数组  
 \** ***@param*** *startPre 前序遍历数组起点  
 \** ***@param*** *endPre 前序遍历数组终点 起点和终点一起决定了新的前序遍历子数组的长度  
 \** ***@param*** *in 是每次递归时新的子树对应的中序遍历数组  
 \** ***@param*** *startIn 中序遍历数组起点  
 \** ***@param*** *endIn 中序遍历数组终点 起点和终点一起决定了新的中序遍历子数组的长度  
 \** ***@return*** *返回重建好的二叉树的根结点  
 \*/* private TreeNode reConstructBinaryTree(int[] pre,int startPre,int endPre,int[] in,int startIn,int endIn){  
 if (startPre>endPre||startIn>endIn)  
 return null;  
 TreeNode root = new TreeNode(pre[startPre]); //因为前序遍历的第一个元素一定是根结点，所以每次先确定这一个结点  
 for (int i=0;i<in.length;i++){ // 遍历中序遍历子数组  
 if (in[i]==pre[startPre]){ // 在中序遍历子数组中找到前序遍历的根结点  
 // 调用递归函数，完成左子树和右子树的重建  
 root.left = reConstructBinaryTree(pre,startPre+1,startPre+i-startIn,in,startIn,i-1);  
 root.right = reConstructBinaryTree(pre,i-startIn+startPre+1,endPre,in,i+1,endIn);  
 break;  
 }  
 }  
 return root;  
 }  
 // 前序递归输出函数，为了验证重建后的二叉树是否正确  
 public void printBinaryTree(TreeNode root) {  
 TreeNode p = root;  
 System.*out*.print(p.val + " ");  
 if (p.left != null)  
 printBinaryTree(p.left);  
 if (p.right != null)  
 printBinaryTree(p.right);  
 }  
  
 public static void main(String[] args) {  
 int[] pre = {1, 2, 4, 7, 3, 5, 6, 8};  
 int[] in = {4, 7, 2, 1, 5, 3, 8, 6};  
 TreeNode root = null;  
 RebuildBinaryTree binaryTree = new RebuildBinaryTree();  
 root = binaryTree.reConstructBinaryTree(pre, in);  
 binaryTree.printBinaryTree(root);  
 }  
}

【学习内容真实可靠，给自己的评分和完成度客观真实，不谦虚，不夸大】