赖凯庭-华为实验班作业（20200925）

一、代码实践

leetcode501 二叉搜索树中的众数

1、空间复杂度 ≠ O（1）

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
    private int count = 1;
    private int max = 1;
    private TreeNode preNode;
    public int[] findMode(TreeNode root) {
        if(root == null){
            return new int[0];
        }
        List<Integer> tmp = new ArrayList<>();
        inorder(root,tmp);
        int[] result = new int[tmp.size()];
        for(int i=0;i<tmp.size();i++){
            result[i] = tmp.get(i);
        }
        return result;
    }
    private void inorder(TreeNode root,List<Integer> temp){
        if(root == null){
            return ;
        }
        inorder(root.left,temp);
// ------------
        if(preNode != null){
            if(preNode.val == root.val){
                count++;
            }
            else{
                count = 1;
            }
        }
        if(count > max){
            max = count;
            temp.clear();
            temp.add(root.val);
        }else if(count == max){
            temp.add(root.val);
```

```
        }
        preNode = root;
// ------------
        inorder(root.right,temp);
    }
}
```

2、空间复杂度为O（1）、

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
    int base,count,maxCount;
    List<Integer> answer = new ArrayList<Integer>();

    public int[] findMode(TreeNode root) {
        TreeNode cur = root,pre = null;
        while(cur != null){
            if(cur.left == null){
                update(cur.val);
                cur = cur.right;
                continue;
            }
            pre = cur.left;
            while(pre.right != null && pre.right != cur){
                pre = pre.right;
            }
            if(pre.right == null){
                pre.right = cur;
                cur = cur.left;
            }else{
                pre.right = null;
                update(cur.val);
                cur = cur.right;
            }
        }
        int[] mode = new int[answer.size()];
        for(int i=0;i<answer.size();++i){
            mode[i] = answer.get(i);
        }
        return mode;
    }

    public void update(int x){
        if(x == base){
            ++count;
        }else{
            count = 1;
            base = x;
        }
```

```
            if(count == maxCount){
                answer.add(base);
            }
            if(count > maxCount){
                maxCount = count;
                answer.clear();
                answer.add(base);
            }
        }
    }
```

二、计算机基础知识

1、Java中Static关键字的用法

（1）static修饰成员方法

static修饰的方法一般称作静态方法，由于静态方法不依赖于任何对象就可以进行访问，因此对于静态方法来说，是没有this的，因为它不依附于任何对象，既然都没有对象，就谈不上this了。并且由于这个特性，在静态方法中不能访问类的非静态成员变量和非静态成员方法，因为非静态成员方法/变量都必须依赖具体的对象才能够被调用。

但是要注意的是，虽然在静态方法中不能访问非静态成员方法和非静态成员变量，但是在非静态成员方法中是可以访问静态成员方法/变量的。

（2）static修饰成员变量

static修饰的变量也称为静态变量，静态变量和非静态变量的区别是：静态变量被所有对象共享，在内存中只有一个副本，它当且仅当在类初次加载时会被初始化。而非静态变量是对象所拥有的，在创建对象的时候被初始化，存在多个副本，各个对象拥有的副本互不影响。

static成员变量的初始化顺序按照定义的顺序进行初始化。

（3）static修饰代码块

static关键字还有一个比较重要的作用就是用来形成静态代码块以优化程序性能。static块可以置于类中的任何地方，类中可以有多个static块。在类初次被加载的时候，会按照static块的顺序来依次执行每个static块，并且只会执行一次。

static块可以优化程序性能，是因为它的特性：只会在类被初次加载的时候执行一次

2、Java中path与classpath的区别

（1）根据环境变量中的path值，找到对应的指令可执行文件进行执行

（2）classpath的作用就是通知JVM用户类的存放路径，加载Java能够识别的自解码文件

三、实验班小组项目

1、实验班学习任务

（1）根据第一篇文档完成python环境+anaconda+jupyter notebook+pycharm community的搭建并在上面跑通第一个程序

（2）根据博客学习线性回归模型，修改并debug博客中的代码

（3）查阅资料学习线性回归等机器学习常见模型，学习SVM模型及其原理推导

2、小组项目

（1）新建分支Statistical-Learning学习笔记及代码，并上传这几天总结的前四章学习笔记及重要算法模型代码实现

（2）添加第一个test-issue，小组内商讨issue提交事项及规范