Day01复杂度分析

注意：课堂练习的解题过程课上都有，我就不多说了，主要记录一下课上没讲的一些题哈。

1.设某算法的时间复杂度为递推关系式T(n)=T(n-1)+n(n为正整数)及T(0)=1，则该算法的时间复杂度为多少？（课堂练习）

参考答案：

```
T(n)=T(n-1)+n
=T(n-2)+(n-1)+n
=T(n-3)+(n-2)+(n-1)+n
=T(n-4)+(n-3)+(n-2)+(n-1)+n
=T(2)+3+…..+(n-2)+(n-1)+n
=T(1)+2+3+…..+(n-2)+(n-1)+n
=T(0)+1+2+3+……+(n-2)+(n-1)+n
=1+1+2+3+…..+(n-2)+(n-1)+n
=1+n*(1+n)/2
=O(n^2)
```

2.求数字n的所有约数。（课后练习）

例如：

输入n=6，则输出1，2，3，6

输入n=10，则输出1，2，5，10

输入n=16，则输出1，2，4，8，16

参考答案及解析过程：

一般大家都能想到普通的for循环：

```
    public static void getDivisor01(int n) {
        for (int i = 1; i <= n; i++) {
            if (n % i == 0) {
                System.out.println(i);
            }
        }
    }
```

上面这个for循环完全没有问题，能实现我们的要求，但大家想一下有没有更优解？

因为约数是成对出现的，当一个（小的）约数出现的时候，与之对应的（大的）约数也就出现了，比如当n=16时，i为2时，16/2=8，这个8也就确定了，所以没有必要遍历到n，那遍历到n/2是不是就可以了呢？于是乎，有同学将代码优化如下：

```java
    public static void getDivisor02(int n) {
        // for 循环结束的条件，当 i * i 大于n的时候就可以结束循环了，这一点不太容易想到，需要大家好好想想
        for (int i = 1; i <= n / 2; i++) {
            if (n % i == 0) {
                // 约数是成对出现的，所以当 n%i==0时，i 和(n/i)都是约数
                System.out.println(i);
                System.out.println(n / i);
            }
        }
    }
```

写到这里有的同学可能决定没问题了，那我再问一句：还有没有更优解？一定要遍历到n的一半吗？

变量i从小到大增加，什么时候就可以认为该遍历的已经遍历完了，后面无需遍历了呢？稍微思考一下就可以发现 当 i * i>n 的时候就已经遍历完了，抽象一点说就是当i与（n/i）相等的时候，就已经遍历完了。这里可能有点难理解，需要你多琢磨一会。

那有人优化代码如下：

```java
    public static void getDivisor03(int n) {
        // for 循环结束的条件，当 i * i 大于n的时候就可以结束循环了，这一点不太容易想到，需要大家好好想想
        for (int i = 1; i * i <= n; i++) {
            if (n % i == 0) {
                // 约数是成对出现的，所以当 n%i==0时，i 和(n/i)都是约数
                System.out.println(i);
                System.out.println(n / i);
            }
        }
    }
```

这么是不是可以了呢？这里需要注意特殊值的处理，比如n=16,i=4时，n/i也是4，所以需要做个特殊值的去重处理，代码如下：

```java
    /**
     * 这道题我主要想考察两个点：
     * 1.for循环结束条件
     * 2.特殊值的一个考虑（就是当i==n/i时，去重问题）
     * 至于查找到约数后，是打印到控制台还是存储到数组中，这都无所谓，都行
     */
    public static void getDivisor04(int n) {
        // for 循环结束的条件，当 i * i 大于n的时候就可以结束循环了，这一点不太容易想到，需要大家好好想想
        for (int i = 1; i * i <= n; i++) {
            // 如果 n模i为0   并且  i等于(n/i)
            if (n % i == 0 && i == n / i) {
                // 控制台打印 i ，只打印i,不打印（n/i）目的是为了去重
                // 比如 n为 16时，当i=4时，16/4也是4，会出现重复值，所以在这里去重
                System.out.println(i);
            } else if (n % i == 0) {
                // 约数是成对出现的，所以当n%i==0时，i 和(n/i)都是约数
                System.out.println(i);
                System.out.println(n / i);
            }
        }
```

```
    }
```

上面的代码应该学任何语言的同学都能看懂，也有学java的同学给我们写出了对应的java版本的代码实现（不会java的同学可忽略哈）：

```java
    public static void getDivisor05(int n) {
        //在java中HashSet有自动去重功能
        HashSet<Integer> set = new HashSet<Integer>();
        for (int i = 1; i <= Math.sqrt(n); i++) {
            // 所以这里不需要更多判断，直接添加即可
            if (n % i == 0) {
                set.add(i);
                set.add(n / i);
            }
        }
        System.out.println(set);
    }
```

如果你有更好的想法欢迎告诉我哈。

Day02数组

1.移动零（课堂练习） https://leetcode-cn.com/problems/move-zeroes/

2.找出数组中重复的数字。（课后练习） https://leetcode-cn.com/problems/shu-zu-zhong-zhong-fu-de-shu-zi-lcof/