Java方向编程题答案

day32

[编程题]24532-在霍格沃茨找零钱

https://www.nowcoder.com/questionTerminal/58e7779c9f4e413cb80792d33ba6acaf

【题目解析】：

其实是个进制转换的问题

【解题思路】：

可以看成特殊进制的数，统一转换为最小单位纳特后再计算即可

【示例代码】：

```
import java.io.IOException;
import java.util.Scanner;

/**
 * 在霍格沃茨找零钱
 * https://www.nowcoder.com/questionTerminal/58e7779c9f4e413cb80792d33ba6acaf
 */
public class Main {
    private static long toKnut(long galleon, long sickle, long knut) {
        return galleon * 17 * 29 + sickle * 29 + knut;
    }

    public static void main(String[] args) throws IOException {
        Scanner scanner = new Scanner(System.in);
        String[] strP = scanner.next().split("\\.");
        String[] strA = scanner.next().split("\\.");
        // 将字符串转换为数值类型，考虑到可能过大，用 long 类型
        long[] longP = {
                Long.parseLong(strP[0]),
                Long.parseLong(strP[1]),
                Long.parseLong(strP[2]),
        };
        long[] longA = {
                Long.parseLong(strA[0]),
                Long.parseLong(strA[1]),
                Long.parseLong(strA[2]),
        };
        // 将单位全部转换为 纳特(Knut)
        long pInKnut = toKnut(longP[0], longP[1], longP[2]);
        long aInKnut = toKnut(longA[0], longA[1], longA[2]);
        long changeInKnut = aInKnut - pInKnut;
        if (changeInKnut < 0) {
            System.out.print("-");
            changeInKnut = -changeInKnut;
```

```
        }
        // 打印时，将单位还原回去
        System.out.format("%d.%d.%d\n",
                changeInKnut / (17 * 29),
                (changeInKnut % (17 * 29)) / 29,
                ((changeInKnut % (17 * 29)) % 29));
    }
}
```

[编程题]25084-2的个数

https://www.nowcoder.com/questionTerminal/31a9495eb02844fb8c0e9ab101053f53

【题目解析】：

题目应该还是挺好懂得

【解题思路】：

这里千万不要愣着从 1 遍历到 n，去数 2 的个数，其实找找规律就好了

每十位数里会出现一个 2。但 2x 是个例外，会多出现 10 个 2 也就是每百位数里会出现 20 个 2。（2、12、22、32...92 + 20、21、22...29）。但 2xx 是个例外，会多出现 100 个 2 依次类推即可根据 n 的值直接得出 2 的个数。

【示例代码】：

```
import java.util.*;

public class Count2 {
    public int countNumberOf2s(int n) {
        int count = 0;      // 最终 2 的个数
        int factor = 1;     // 从低位到高位计算 2 的个数
        int low = 0;        // 屏蔽掉低位
        int current = 0;    // 当前计算位数的值
        int high = 0;       // 计算高位
        while(n / factor != 0) {
            low = n - (n / factor) * factor;
            current = (n / factor) % 10;
            high = n / (factor * 10);

            // 根据当前的
            switch(current){
                case 0:
                case 1:
                    count += high * factor;
                    break;
                case 2:
                    count += high * factor + low + 1;
                    break;
                default:
                    count += (high + 1) * factor;
                    break;
            }
            factor *= 10;
        }
```

```
            return count;
        }
    }
```