```
                }
            }
            return false;
        }
```

数字的英文表达和中文表达

【题目】

给定一个 32 位整数 num，写两个函数分别返回 num 的英文与中文表达字符串。

【举例】

num=319
英文表达字符串为：Three Hundred Nineteen
中文表达字符串为：三百一十九

num=1014
英文表达字符串为：One Thousand, Fourteen
中文表达字符串为：一千零十四

num=-2147483648
英文表达字符串为：Negative, Two Billion, One Hundred Forty Seven Million, Four Hundred Eighty Three Thousand, Six Hundred Forty Eight
中文表达字符串为：负二十一亿四千七百四十八万三千六百四十八

num=0
英文表达字符串为：Zero
中文表达字符串为：零

【难度】

校 ★★★☆

【解答】

本题的重点是考查面试者分析业务场景并实际解决问题的能力。本题实现的方式当然是多种多样的，本书提供的方法仅是作者的实现，希望读者也能写出自己的实现。

英文表达的实现。英文的表达是以三个数为单位成一组的，所以先要解决数字 1~999

的表达问题。首先看数字 1~19 的表达问题，具体过程请参看如下代码中的 num1To19 方法。

```
public String num1To19(int num) {
        if (num < 1 || num > 19) {
                return "";
        }
        String[] names = { "One ", "Two ", "Three ", "Four ", "Five ", "Six ",
                "Seven ", "Eight ", "Nine ", "Ten ", "Eleven ", "Twelve",
                "Thirteen ", "Fourteen ", "Fifteen ", "Sixteen","Sixteen",
                "Eighteen ", "Nineteen " };
        return names[num - 1];
}
```

然后利用 num1To99 函数来解决数字 1~99 的表达问题。具体参看如下的 num1To99 方法。

```
public String num1To99(int num) {
        if (num < 1 || num > 99) {
                return "";
        }
        if (num < 20) {
                return num1To19(num);
        }
        int high = num / 10;
        String[] tyNames = { "Twenty ", "Thirty ", "Forty ", "Fifty ",
                        "Sixty ", "Seventy ", "Eighty ", "Ninety " };
        return tyNames[high - 2] + num1To19(num % 10);
}
```

有以上两个函数，再解决数字 1~999。具体请参看如下代码中的 num1To999 方法。

```
public String num1To999(int num) {
        if (num < 1 || num > 999) {
                return "";
        }
        if (num < 100) {
                return num1To99(num);
        }
        int high = num / 100;
        return num1To19(high) + "Hundred " + num1To99(num % 100);
}
```

最后可以解决最终的问题，需要注意如下几个特殊情况：

- num 为 0 的情况要单独处理。
- num 为负的处理，对于负数，一律以处理其绝对值的方式来得到表达字符串，然后加上“Negative.”的前缀，所以 num 为 Integer.MIN_VALUE 时，也是特殊情况。

- 把 32 位整数分解成十亿组、百万组、千组、1~999 组。对每个组的表达利用 num1To999 方法，再把组与组之间各自的表达字符串连接起来即可。

最后是英文表达的主方法，参见如下代码中的 getNumEngExp 方法。

```
public String getNumEngExp(int num) {
        if (num == 0) {
                return "Zero";
        }
        String res = "";
        if (num < 0) {
                res = "Negative, ";
        }
        if (num == Integer.MIN_VALUE) {
                res += "Two Billion, ";
                num %= -2000000000;
        }
        num = Math.abs(num);
        int high = 1000000000;
        int highIndex = 0;
        String[] names = { "Billion", "Million", "Thousand", "" };
        while (num != 0) {
                int cur = num / high;
                num %= high;
                if (cur != 0) {
                        res += num1To999(cur);
                        res += names[highIndex] + (num == 0 ? " " : ", ");
                }
                high /= 1000;
                highIndex++;
        }
        return res;
}
```

中文表达的实现。与英文表达的处理过程类似，都是由小范围的数向大范围的数扩张的过程，这个过程有非常不同的处理细节。

首先解决数字 1~9 的中文表达问题，具体参看如下代码中的 num1To9 方法

```
public String num1To9(int num) {
        if (num < 1 || num > 9) {
                return "";
        }
        String[] names = { "一", "二", "三", "四", "五", "六", "七", "八", "九" };
        return names[num - 1];
}
```

利用 num1To9 方法，我们来看看数字 1~99 如何表达。其中有一个很值得注意的细节，16 的表达是十六，116 的表达是一百一十六，1016 的表达可以是一千零十六，也可以是一

千零一十六。这个细节说明，对 10~19 来说，如果其前一位（也就是百位）有数字，则表达该是一十～一十九。如果百位上没数字，则表达应该一律规定为十～十九。具体过程请参看如下代码中的 num1To99 方法，boolean 型参数 hasBai 表示是否其前一位（百位）有数字。

```
public String num1To99(int num, boolean hasBai) {
        if (num < 1 || num > 99) {
                return "";
        }
        if (num < 10) {
                return num1To9(num);
        }
        int shi = num / 10;
        if (shi == 1 && (!hasBai)) {
                return "十" + num1To9(num % 10);
        } else {
                return num1To9(shi) + "十" + num1To9(num % 10);
        }
}
```

利用 num1To9 与 num1To99 方法后，接下来解决数字 1~999 的表达，具体过程请参看如下代码中的 num1To999 方法。

```
public String num1To999(int num) {
        if (num < 1 || num > 999) {
                return "";
        }
        if (num < 100) {
                return num1To99(num, false);
        }
        String res = num1To9(num / 100) + "百";
        int rest = num % 100;
        if (rest == 0) {
                return res;
        } else if (rest >= 10) {
                res += num1To99(rest, true);
        } else {
                res += "零" + num1To9(rest);
        }
        return res;
}
```

然后是数字 1~9999 的表达问题，见如下代码中的 num1To9999 方法。

```
public String num1To9999(int num) {
        if (num < 1 || num > 9999) {
                return "";
```

```
        }
        if (num < 1000) {
                return num1To999(num);
        }
        String res = num1To9(num / 1000) + "千";
        int rest = num % 1000;
        if (rest == 0) {
                return res;
        } else if (rest >= 100) {
                res += num1To999(rest);
        } else {
                res += "零" + num1To99(rest, false);
        }
        return res;
}
```

接下来是数字 1~99999999 的表达问题，见如下代码中的 num1To99999999 方法。

```
public String num1To99999999(int num) {
        if (num < 1 || num > 99999999) {
                return "";
        }
        int wan = num / 10000;
        int rest = num % 10000;
        if (wan == 0) {
                return num1To9999(rest);
        }
        String res = num1To9999(wan) + "万";
        if (rest == 0) {
                return res;
        } else {
                if (rest < 1000) {
                        return res + "零" + num1To999(rest);
                } else {
                        return res + num1To9999(rest);
                }
        }
}
```

最后是中文表达的主方法，参见如下代码中的 getNumChiExp 方法。

```
public String getNumChiExp(int num) {
        if (num == 0) {
                return "零";
        }
        String res = num < 0 ? "负" : "";
        int yi = Math.abs(num / 100000000);
        int rest = Math.abs((num % 100000000));
        if (yi == 0) {
                return res + num1To99999999(rest);
        }
```

```
            res += num1To9999(yi) + "亿";
            if (rest == 0) {
                    return res;
            } else {
                    if (rest < 10000000) {
                            return res + "零" + num1To99999999(rest);
                    } else {
                            return res + num1To99999999(rest);
                    }
            }
    }
```

该类型的代码面试题目实际上是相当棘手的。通常是由小的、简单的场景出发，把复杂的事情拆解成简单的场景，最终得到想要的结果。

分糖果问题

【题目】

一群孩子做游戏，现在请你根据游戏得分来发糖果，要求如下：

1．每个孩子不管得分多少，起码分到 1 个糖果。

2．任意两个相邻的孩子之间，得分较多的孩子必须拿多一些的糖果。

给定一个数组 arr 代表得分数组，请返回最少需要多少糖果。

例如：arr=[1,2,2]，糖果分配为[1,2,1]，即可满足要求且数量最少，所以返回 4。

【进阶题目】

原题目中的两个规则不变，再加一条规则：

3．任意两个相邻的孩子之间如果得分一样，糖果数必须相同。

给定一个数组 arr 代表得分数组，返回最少需要多少糖果。

例如：arr=[1,2,2]，糖果分配为[1,2,2]，即可满足要求且数量最少，所以返回 5。

【要求】

arr 长度为 N，原题与进阶题都要求时间复杂度为 $O(N)$，额外空间复杂度为 $O(1)$。

【难度】

校 ★★★☆