巧妙的位运算

今日目标:

1：能说出常用的几个位运算技巧

2：完成今日面试题目

实战题目

191. 位1的个数

思科，字节，腾讯半年内面试题

思路1：位移+计数

对于学C++的人来说，代码可能是这样写

```cpp
class Solution {
public:
    int hammingWeight(uint32_t n) {
        unsigned int count = 0; // 计数器
        while (n > 0) {
            if ((n & 1) ==1) { //当前位为1
                count++;
            }
           n >>=1 ; //右移一位
        }
        return count;
    }
};
```

但如果这样的代码放到java中，可能是这样

```java
public class Solution {
    // you need to treat n as an unsigned value
    //位移+计数
    public int hammingWeight(int n) {
        int count = 0; // 计数器
        while (n > 0) {
            if ((n & 1) ==1) { //当前位为1
                count++;
            }
           n >>=1 ; //右移一位
        }
        return count;
    }
}
```

对于以下测试用例:

```
11111111111111111111111111111101
```

无法通过！原因是什么？

> `11111111111111111111111111111101`表示的是有符号整数`-3`

```
int unsignedInt =
Integer.parseUnsignedInt("11111111111111111111111111111101",2);
System.out.println(unsignedInt);
```

原因就在`while`循环的条件上

```
while (n > 0) { //该条件得不到通过，java中 >> 是有符号右移

}
```

java版AC代码：直接移位32次

```
public class Solution {
    // you need to treat n as an unsigned value
    //位移+计数
    public int hammingWeight(int n) {
        int count = 0; // 计数器
        for (int i=0;i<32;i++) {
            if ((n & 1) ==1) { //当前位为1
                count++;
            }
           n >>=1 ; //右移一位
        }
        return count;
    }
}
```

需要循环32次。

思路2：&运算消去最低位的1

```
public class Solution {
    // you need to treat n as an unsigned value
    //&运算消去最低位的1,执行次数少于32次
    public int hammingWeight(int n) {
        int count = 0;
        while (n !=0) {
            count++;
            n = n & (n-1);
        }
        return count;
    }
}
```

231. 2的幂

字节，小米，腾讯面试，231. 2的幂

知识点：如果是2的幂次方则只会有一个二进制位为1，则可以借助 n & (n-1) 的结果是否为0

```
class Solution {
    //如果是2的幂次方则只会有一个二进制位为1
    public boolean isPowerOfTwo(int n) { // 借助 n & (n-1)
        if (n <=0) {
            return false;
        }
        return (n & (n-1)) == 0;
    }
}
```

190. 颠倒二进制位

字节，三星，今日头条面试题，190. 颠倒二进制位

> 本题多解法，可参考题解：https://leetcode-cn.com/problems/reverse-bits/solution/zhi-qi-ran-zhi-qi-suo-yi-ran-wei-yun-suan-jie-fa-x/

1、取模求和

```
public class Solution {
    // you need treat n as an unsigned value
    public int reverseBits(int n) {
        int ret = 0 ;
        for (int i=0;i<32;i++) {
            ret = (ret<<1) + (n & 1);
            n = n>>1;
        }
        return ret;
    }
}
```

2、按位翻转

```java
public class Solution {
    // you need treat n as an unsigned value
    //按位翻转
    public int reverseBits(int n) {
        int res = 0;
        for (int i=0;i<=31;i++) {
            res ^= (n & (1<<i)) !=0 ? (1<<(31-i)) : 0;
        }
        return res;
    }
}
```

52. N皇后 II

腾讯，字节最近面试题，面试题 52. N皇后 II

位运算解法：此类问题的终极解法

```java
class Solution {

    int total = 0;
    //终极解法：位运算
    public int totalNQueens(int n) {
        dfs(n,0,0,0,0);
        return total;
    }

    public void dfs(int n,int row,int col,int pie,int na) {
        //terminal
        if (row == n) {
            total++;
            return;
        }
        //算出棋盘当前行还有哪些位置可以放置皇后由 0 变成 1，以便进行后续的位遍历
        int bits = ~(col | pie | na) & ((1<<n )-1);
        while (bits > 0) {
            int mask = bits & -bits;
            dfs (n,row+1,col | mask,(pie | mask )>>1,(na | mask) <<1) ;
            bits &= bits -1;
        }
    }
}
```

进阶题目：

51. N 皇后

面试题 08.12. 八皇后

338. 比特位计数