位运算

程序中的所有数在计算机内存中都是以二进制的形式储存的。位运算就是直接对整数在内存中的二进制位进行操作。

位运算符号的解析

异或的常用操作

重要的位运算符

优点：

1、直接对整数在内存中的二进制位进行操作

2、处理速度快

案例：

编写一个函数，输入是一个无符号整数，返回其二进制表达式中数字位数为 ‘1’ 的个数编写一个函数，输入是一个无符号整数，返回其二进制表达式中数字位数为 ‘1’ 的个数

时间复杂度：0（ n ）缺点：容易超时

```
public class Solution {
// you need to treat n as an unsigned value
public int hammingWeight(int n) {
int count=0;
for(int i=0;i<32;i++){ //不能用while，因为最高位的补位不一定是0，请注意，在某些语言（如 Java）中，没有无符号整数类型。在这种情况下，输入和输出都将被指定为有符号整数类型，并且不应影响您的实现，因为无论整数是有符号的还是无符号的，其内部的二进制表示形式都是相同的；
java中的都是有符号位的；
、、注意负数是以补码的形式呈现的
if((n&1)==1){
count+=1;
}
n=n>>1;
}
return count;
}
```

方法二：位运算

时间复杂度：o(n)

```
public class Solution {
// you need to treat n as an unsigned value
public int hammingWeight(int n) {
int count=0;
while(n!=0){
count+=1;
```

```
n=n&(n-1); //
}
return count;
}
}
```

例题：给定一个整数，编写一个函数来判断它是否是 2 的幂次方。

方法一：时间复杂度o(logn)

```
class Solution {
public boolean isPowerOfTwo(int n) {
if(n<=0) return false;
while(n>1){
if(n%2!=0){
return false;
}else{
n=n/2;
}
}
return true;
}
}
```

方法二：位运算

若 n = 2^x

且 x 为自然数（即 n 为 2 的幂），则一定满足以下条件：

恒有 n & (n - 1) == 0，这是因为：

n 二进制最高位为 1，其余所有位为 0；

n - 1 二进制最高位为 0，其余所有位为 1；

一定满足 n > 0。

因此，通过 n > 0 且 n & (n - 1) == 0 即可判定是否满足 n = 2^x

```
class Solution {
public boolean isPowerOfTwo(int n) {
return n > 0 && (n & (n - 1)) == 0;
}
}
```

原文链接：https://blog.csdn.net/weixin_44625138/article/details/101226529