位运算

删去二进制表示中最右侧1的方法：x = x & (x–1) ---(用于计算二进制中的1的位数)

举例：

x=010111000

x–1 = 010110111

容易发现

x & x–1 的最右侧的1变为0

二分计算二进制中1的个数

网上的回答讲的不好，并没有详细讲原理而是扔了个示例让你理解，以下是我自己的理解：（CSDN上找的，但CSDN很明显也抄的别人的，所以并不知道原始出处，姑且贴一下CSDN的链：计算一个 32 位无符号整数有多少个位为 1_写一个函数计算一个无符号整型有多少位被置1–CSDN博客）

C++

```
unsigned popcount (unsigned u)
{
    u = (u & 0x55555555) + ((u >> 1) & 0x55555555);
    u = (u & 0x33333333) + ((u >> 2) & 0x33333333);
    u = (u & 0x0F0F0F0F) + ((u >> 4) & 0x0F0F0F0F);
    u = (u & 0x00FF00FF) + ((u >> 8) & 0x00FF00FF);
    u = (u & 0x0000FFFF) + ((u >> 16) & 0x0000FFFF);
    return u;
}
```

这个二分法的原理：

> 第一段代码：u = (u & 0x55555555) + ((u >> 1) & 0x55555555);

0x55555555是（01010101010101010101010101010101），即取二进制的奇数位。

u右移一位再与0x55555555相与，即取二进制的偶数位，再将结果右移一位；

相加的结果就可以理解为将二进制数分为 $\frac{N}{2}$ 个两个bit位的片段，在**每两位里记录了对应两位里1的个数**；

其实可以理解为第一步已经统计好1的个数了，只是将计数分散在每个2bit位里，之后的操作就是利用位运算将他们加起来。

> 第二段代码：u = (u & 0x33333333) + ((u >> 2) & 0x33333333);

0x33333333 是（00110011001100110011001100110011）,即将二进制按4个bit位划分，取每四位里的低两位

u & 0x33333333 取每四位里的低两位（每四位的值为原值里每四位里低两位的1的个数）

(u >> 2) & 0x33333333 取每四位里的高两位再右移两位；（每四位的值为原值里每四位里高两位的1的个数）

相加的结果就是统计原值每四位里1的个数并存储到对应的四位里表示；

之后的代码类似第二段的理解，最后结果就是32位里1的个数。

用一个例子理解：

u = 11010110

第一阶段（统计1的个数，结果分散保存在每两位里）

取奇数位：(u')

u = 11–01–01–10

u' = 01–01–01–00

对比发现u'记录了**每两位里低位的1的个数**

取偶数位并右移一位：（u''）

10000010 –> 01000001

u = 11–01–01–10

u'' = 01–00–00–01

对比发现u''记录了**每两位里高一位的1的个数**

相加(u''')

u = 11–01–01–10

u''' = 10–01–01–01

可以发现u'与u''两数之和的**每两位里包含对应两位的1的个数**

第二阶段：（将第一阶段对每两位1的统计结果记录加起来，每个结果分散在每四位片段里）

取每四位里低两位（u1'）

u''' = 1001–0101（上阶段结果）

u1' = 0001–0001

每四位里保存了对应四位低两位的1个数记录

取每四位里高两位并右移两位（u1''）

1000–0100 –> 0010–0001

u''' = 1001–0101（上阶段结果）

u1'' = 0010–0001

可以发现**每四位里保存了对应四位高两位的1个数记录**

相加(u1''')

u = 1101–0110

u1''' = 0011–0010

可以发现，高四位值为3，低四位值为2，我们成功**将每四位的1的个数记录了下来，分散在每四位里。**

第三阶段：（将第二阶段的1记录结果加起来，每个结果分撒在每八位片段里）

取每八位里低四位（u2'）

u1''' = 0011–0010（上阶段结果）

u2' = 0000–0010

每八位里保存**对应八位低四位的1个数记录**

取每八位里高四位并右移四位（u2''）

0011–0000 –> 0000–0011

u1''' = 0011–0010（上阶段结果）

u2'' = 0000–0011

每八位里保存**对应八位里高四位的1个数记录**

相加（u2'''）

u = 1101–0110

u2''' = 0000–0101

可以发现我们成功将每八位里高、低四位的1的个数记录相加在一起，分散在每八位里。而本例为8位二进制数，因此u2'''的值即为最终结果5。

异或运算（有关重复值可以往这方面思考）

由于异或有着这样的性质：

1. $A \oplus A = 0$
2. $A \oplus 0 = A$

因此可以用于消除数据中的重复值

如：简单题/数组、顺序表/只出现一次的数字