位运算技巧集合，用位运算可以大大提高程序的性能，但是很多技巧在实际编程中接触较少，这里做一个相关记录。

一个结论：位运算均满足交换律、结合律。

1. 位操作符

&与，|或，~取反，^异或，<<左移，向高位移动，>>右移，向地位移动

1. 常见位运算问题
2. 实现乘除法

左移或右移实现2的整次幂的乘除，例如左移一位为乘2.

a = 2

a << 1 == 4

1. 交换两数

普通操作

a = a + b

b = a – b

a = a – b

位操作

a ^= b

b ^= a

a ^= b

解释：第一步a=(a^b)，第二步b=b^(a^b)=(b^b)^a=a，第三步a=(a^b)^a=(a^a)^b=b。

1. 判断奇偶数

a & 1 == 1则为奇数

1. 交换符号

return ~a + 1

正数取反加1，刚好是对应的负数；负数取反加1，变为原码。

1. 求绝对值

先判断正负，正数是本身，负数是取反加1.判断符号可以将其右移31位（正数右移31位得到0，负数得到-1，即0xffffffff）

i = a >> 31

return i == 0 ? a : (~a + 1)

优化：0与任何数异或保持不变，-1即0xffffffff与任何数异或都是取反，所以(a^i)可以实现取反或不变，然后(a^i)-i可根据i是0或-1实现不变或+1效果

1. 高低位变换

给定一个16位的无符号整数，将其高8位和低8位交换，求出交换后的值

右移8位得到低8位，且高8位为0，左移8位得到高8位，且低8位为0，将二者求或即可得到目标结果

1. 二进制逆序

给定无符号数，返回二进制求逆序后的结果

10010逆序后得到01001

逆序操作是从首尾开始进行互换。由于二进制中使用位的高低位交换会更方便处理，这里分组进行处理

第一步：每2位为一组，组内进行高低位交换

第二步：每4位一组，组内高低位进行变换

第三步：每8位一组，组内高低位互换

第四步：每16位一组，组内高低位互换（这里就完成互换，因为无符号整数是32位存储）

对于第一步交换繁琐，可以先取出原数的奇数位和偶数位，再将奇数位右移一位，偶数位左移一位，二者求或即可得到交换效果。取奇数位可以表示为a&0xAAAA，取偶数位可以表示为a&0x5555，所以第一步修改为a = ((a&0xAAAA) >> 1) | ((a&0x5555) << 1)

同理，每4位一组就是22交换，即a = ((a&0xCCCC) >> 2) | ((a&0x3333) << 2)

每8位就是44交换，即a=((a&0xF0F0) >> 4) | ((a&0x0F0F) << 4)

每16位就是88交换，即a=((a&0xFF00) >> 8) | ((a&0x00FF) << 8)

所以整个代码是

a = ((a & 0xAAAA) >> 1) | ((a & 0x5555) << 1);

a = ((a & 0xCCCC) >> 2) | ((a & 0x3333) << 2);

a = ((a & 0xF0F0) >> 4) | ((a & 0x0F0F) << 4);

a = ((a & 0xFF00) >> 8) | ((a & 0x00FF) << 8);

1. 统计二进制中1的个数

a & (a – 1)可以将a的二进制中最低位的1变为0，只需要统计变0的次数即为1的个数

while a:

a = a & (a – 1)

cnt += 1

1. 来自ACM中枚举集合的技巧
2. 要求集合中不能有两个相邻的元素，相邻指的是索引位置的相邻，比如选了2就不能选3，二进制表示选取2和3分别为010,100（对应位1表示选，0表示未选），所以不相邻的判断方式

if ((mask >> 1) & mask) continue

使用位运算的一些技巧：

如果要求解的是有限状态，可以通过二进制表示不同的状态，并写出真值表表示状态转移过程，写出对应逻辑表达式，作为状态迁移过程。并最终返回目标状态结果。