Week 1

软73 沈冠霖 2017013569

March 15, 2019

1 题目思路与证明

1.1 BitAnd 因为对于整数x,y的任意一个二进制位,设为a,b,根据摩根定律,有$(a\&b) = \sim((\sim a)|(\sim b))$,且其每一位运算相互独立,因此 $x\&y = \sim((\sim x)|(\sim y))$ 成立

1.2 GetByte 只需要做两件事:
首先,把x右移8n位,让你想要的8个bit落在x的最后8位.
其次,用和0xff按位且运算提取这8位

1.3 LogicalShift 问题核心:构造前n位为1,其余为0的数
对于非负数,直接右移就是正确结果,而对于负数,右移n位后要消去前n位的1
因此,我先获取x的第一位,signal,再将其右移n-1位
如果x为非负数,则signal = 0,右移n-1位后还是0;
如果x为负数,则signal第一位是1,其余为0,右移n-1位后,其前n位是1,其余是0
将x与signal做异或,就可以保留后面的信息,同时消除前n位可能的1

1.4 BitCount 借鉴了 https://www.douban.com/note/274239939/ 的一种思路
先考虑2位的情况:a[0]+a[1] = a[0] + (a >> 1)[0] = 2位里1的个数
计算出每个2位数字中1的个数之后,可以把每4位中的左两位右移2位,和右两位相加,得到每4位数字中1的个数
以此类推,计算出每2^n位数字中1的个数之后,可以把前2^n位右移n位,和后2^n位相加,就可以得到$2^{(n+1)}$位中的个数

1.5 Bang 问题的核心:0的相反数还是0,其符号位都是0;
其余$\forall x, x != TMIN$,有$sign(x) != sign(-x)$
TMIN和-TMIN都是0x80000000,符号都是1
因此只需要求出x和$-x = \sim x + 1$两个数的符号,做或运算就可以了

2.1 tmin 如果x是长度为n字节的two's complement integer,则 $-2^{n-1} \leq x \leq 2^{n-1}-1$, 对于32位的int,最小的这个数显然是$INT_MIN$,也就是$1 << 31$

2.2 fitsBits 如果n=32,那么任何int范围内的数都是
如果其他情况,那么只要$0 \leq x + 2^{n-1} \leq 2^n - 1$,x也是.
所以我先判断n是否为32,如果不是的话,就判断x加上2^{n-1}这个偏置之后是否在此范围内

2.3 divpwr2 为了实现向0取整,应该这样:负数向上偏移$2^n - 1$,非负数不变,也就是向上偏移$2^0 - 1$
因此,我提取了符号位并且让字符signal的每一位都是符号位,然后偏移$2^{n\&signal} - 1$
这样如果是非负数,则n&signal = 0,否则n&signal = n,就能实现了

2.4 negate 因为每个int都是补码存储,所以其逐位取反再+1就是其相反数

2.5 IsPositive 若x是负数,则首位是1,!x = 0;
若x = 0,!x = 1,首位是0;
若x是正数,则首位是0,!x = 0
因此我计算 !x + 首位的结果,然后取非就可以判断了

2.6 IsLessOrEqual 先单独考虑x=y,此时x-y=0,!(x-y)=1 而再考虑$x < y$,此时,如果x与y同为非负数或者同为负数,x - y必定不溢出,结果正负反映了其大小关系
如果x,y符号位不同,则可以之间判断其符号位.
因此我设x的符号位为二进制变量a,y的符号位为b,$x - y$的符号位为c,则对于x!=y的情况,可以画出如Table.1的卡诺图:
根据卡诺图,可以化简得到:结果$ans = (a\&(\sim b))|(a\&c)|((\sim b)\&c)$

Table 1: 2.6题的卡诺图求解

	00	01	11	10
0	0	0	0	1
1	1	0	1	1

注：第一行代表了ab的二进制取值,第一列代表了c的二进制取值,表中其他数据代表了abc不同取值情况下的二进制输出

2.7 ILog2 这道题的实质是求从左到右数第一个1的位置,和1.5一样,这道题还是要看每一位,因此也模仿1.5使用递归.
思路如下:如果前16位有1(不是0),只需要在前16位找1就可以了,否则只在后16位找
当前的16位,如果前8位有1,只需要在前8位找,否则只在后8位找
以此类推,宏观上来说,可以使用5层递归来完成任务.
微观上来说,需要一个二进制变量w记录前一半位置是否为0,并且找到一个函数f(w),使得当w=1时,使用后一半的数字,并且,w=0时,使用前一半的数字
因此,我用current来记录当前需要处理的一串数字,用place记录那串数字最右一位的位置(比如说,我处理前16位,current就是$x >> 16$,place就是16).
令f(w)=$\sim w + 1$,这样当w=1时,f(w)全是1;w=0时,f(w)全是0
$new\_current$ = current前半部分 + current后半部分&f(w),这样,如果w=1,前半部分是0,后半部分留了下来;如果w=0,前半部分留了下来,后半部分被消除
place += $\sim$f(w)&当前一半的位数,这样,如果w=0,则place向左进一半的位数,否则不动

3.1 FloatNeg 只需要判断是否是NaN,如果是NaN,也就是exp位都是1,后23位不都是0,则直接输出
否则,翻转符号后输出

3.2 FloatI2f 首先,要提取x的符号,把x转化为其绝对值.其中,0和MIN_INT为特例,需要提前识别输出.
之后,要用移位法计算出x绝对值最高为1的位的位置(对应2的几次方),再加上偏置变成指数
最后,需要提取出x的小数部分,也就是x绝对值排除掉最高位的其他位,移到对应位置.
如果x的小数部分有需要舍弃的位的话,需要将其提取出来.
设x小数部分保留的最低位是a,被x舍弃的小数部分是b,根据保留小数的规则,分三种情况:
1:$b < 0.5$,小数部分不变
2:$b > 0.5$,小数部分+1
3:b = 0.5,此时如果a = 1,则小数部分+1,来向偶数取整;否则小数部分不变.

3.3 FloatTwice 首先,先用位运算提取符号位,指数位,小数位
之后,根据当前的指数位分为三种情况:1:这个数是无穷或NaN,直接输出
2:这个数是normalized,将指数位+1,如果此时指数位变成全1,则输出无穷
3:这个数是denormalized,指数位都是0的数,小数位需要左移一位,如果原来的小数位第一位是1,则这个数变为正规数,指数位变为1

2 总结

这些题目大概能归纳出如下的思路：

1.提取特征 **类**似1.5,2.6题，都有明显的特征：输出是0或1，都是把情况进行分类。这种题我的思路通常是提取值的特征，比如正负数的符号，0与0x80000000与其相反数完全相同之类的，然后用这些特征的逻辑关系求解。对于较少的特征，可以直接列出表达式，如果像2.6这种三四个特征，可以使用数电课上学习过的卡诺图法来化简逻辑表达式，以最小化算符。

2.提取位置 **像**1.2这种题，还有很多题的中间步骤需要提取一个数的某些位置，提取的方法通常是：先把这个数右移到0，再用一个辅助数（要的位是1，否则是0）和这个数做与运算，以排除干扰。

3.排除负数干扰 **我**总结出，这些题中，负数有两种干扰方式。首先，负数的右移会让前几位都是1（1.3,1.4,2.3）.其次，浮点数的运算算的是绝对值，和负数关系不大。
我的解决方案有三种：首先，如果要进行不断的移位处理，我会先求出符号位，消除符号位，然后再操作。其次，对于除法取整问题，可以向上偏移一个单位。最后，对于float操作，需要将其符号位求出来，转化成绝对值。

4.递归/分治/二分法 **有**些问题，比如1.4,2.6，需要操作一个二进制数的每一位才行，这种情况下，操作时间复杂度很高，需要优先考虑分治/二分法，把$\theta(n)$的问题转化为$\theta(lgn)$

5.浮点数的问题 **浮**点数严格分为三部分，因此，对于浮点数，我的一般处理方法是严格分为三个部分进行处理。而浮点数还有两种特殊情况：指数位全1的情况（无穷与溢出），指数位全0的非正规情况以及其和正规情况的相互转化。