开始错误理解了log函数，它只能计算以e为底和以10为底的对数，以2 为底会出错，提交代码如下：

class Solution {

public:

int hammingWeight(uint32\_t n) {

if(n==0) return 0;

int ret=0;

while(n!=0)

{

if(ceil(log(n))==floor(log(n))) {return 1+ret;break;}

n=floor(log(n));

ret++;

}

return ret;

}

};

后来进入思维固定模式走不出来了，网上看博客说到的几种思想自己实现了一下，代码如下：

第一种：

class Solution {

public:

int hammingWeight(uint32\_t n) {

int res;

res=0;

while(n!=0)

{

if(n%2==1)

{

res++;

}

n=n/2;

}

return res;

}

};

这是最容易想到的，跟取一个整数的每一位的位数除10思想类似。

第二种：

class Solution {

public:

int hammingWeight(uint32\_t n) {

int res=0;

int temp;

while(n!=0)

{

**temp=n&0x1; //最后一位为1就加1，为0就加0**

res+=temp;

n=n>>1;

}

return res;

}

};

这种方法采用位操作，与16进制的1相与，再右移。

第二次尝试左移1导致出错：

class Solution {

public:

int hammingWeight(uint32\_t n) {

int count=0;

int temp=1;

for(int i=0;i<32;i++){

if((n&0x(temp))==1){

count++;

}

temp=temp<<1;

}

return count;

}

};

**第三种：**

class Solution {

public:

int hammingWeight(uint32\_t n) {

int res=0;

while(n!=0)

{

**n=n&(n-1);**

res++;

}

return res;

}

};

这种方法让人眼前一亮，它是从n的低位到高位一次消去1，直到n为0，消的次数即n里1的个数。