user: sakura

第11题——二进制中1的个数

请实现一个函数，输入一个整数（以二进制串形式），输出该数二进制表示中 1 的个数。例如，把 9 表示成二进制是 1001，有 2 位是 1。因此，如果输入 9，则该函数输出 2。

示例 1：

```
输入：00000000000000000000000000001011
```

```
输出：3
```

解释：输入的二进制串 00000000000000000000000000001011 中，共有三位为 '1'。 示例 2：

```
输入：00000000000000000000000010000000
```

```
输出：1
```

解释：输入的二进制串 00000000000000000000000010000000 中，共有一位为 '1'。 示例 3：

```
输入：11111111111111111111111111111101
```

```
输出：31
```

解释：输入的二进制串 11111111111111111111111111111101 中，共有 31 位为 '1'。

提示：

输入必须是长度为 32 的 二进制串 。

位运算+右移（1）

分析： 很简单，每次循环先判断N是否合法，再判断 n&1 是否合法，如果是，那么count++; 并且，每一次循环n都右移一位；

```
int hammingWeight(uint32_t n) {
    int count=0;
    while(n)//
    {
        if(n&1)
```

```
        {
            count++;
        }
        n>>=1;
    }
    return count;
}
```

另一种位运算

算法核心:

n &= (n - 1);

分析:

得到数据 n = 001011 于是 n-1=001010

第一次

```
count=1;
001011
001010      结果为001010，  循环继续
```

第二次

```
count=2;
001010
001001        结果为001000，循环继续
```

第三次

```
count=3;
001000
000100          结果为0，跳出循环
```

最后返回count即可。

如此一来，大大减少了循环次数，NB！！！

```
int hammingWeight(uint32_t n)
{
    int count = 0;
    while (n) {
        count++;
        n &= (n - 1); //每次 N 的值-1，再和自己与运算
    }
```

```
    return count;
}
```