

```
/*  
编写一个函数，输入是一个无符号整数，返回其二进制表达式中数字位数为‘1’的个数（也被称为汉明重量）。
```

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: 000000000000000000000000010000000

输出: 1

解释: 输入的二进制串 000000000000000000000000010000000 中，共有一位为 '1'。

示例 3:

输入: 1111111111111111111111111111101

输出: 31

解释: 输入的二进制串 1111111111111111111111111111101 中，共有 31 位为 '1'。

提示:

请注意，在某些语言（如 Java）中，没有无符号整数类型。在这种情况下，输入和输出都将被指定为有符号整数类型，并且不应影响您的实现，因为无论整数是有符号的还是无符号的，其内部的二进制表示形式都是相同的。

在 Java 中，编译器使用二进制补码记法来表示有符号整数。因此，在上面的 示例 3 中，输入表示有符号整数 -3。

进阶:

如果多次调用这个函数，你将如何优化你的算法？

来源: 力扣 (LeetCode)

链接: <https://leetcode-cn.com/problems/number-of-1-bits>

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

```
*/
```

分析:

- 直接的逻辑当然是数1的个数，所以我想到了方法一，利用数组一个一个数。
- 利用与逻辑的特性可以知道 $n = n \& (n - 1)$ 在 $n \neq 0$ 的情况下，能从低位到高位消掉原 n 中的1，对应方法二。
- 扩展一下， $n = n | (n + 1)$ 会在原始 n 中增加一个1(n 不为 $0\text{xf}\text{f}\text{f}\text{f}\text{f}\text{f}\text{f}$ 时)，而且是从低位到高位，依次把0变成1。

方法一:C_Solution

```
int hammingweight(uint32_t n)
{
    int count = 0;
    int i     = 0;
    const uint32_t arr[32] =
    {
        0x00000001,0x00000002,0x00000004,0x00000008,
        0x00000010,0x00000020,0x00000040,0x00000080,
        0x00000100,0x00000200,0x00000400,0x00000800,
        0x00001000,0x00002000,0x00004000,0x00008000,
        0x00010000,0x00020000,0x00040000,0x00080000,
        0x00100000,0x00200000,0x00400000,0x00800000,
        0x01000000,0x02000000,0x04000000,0x08000000,
        0x10000000,0x20000000,0x40000000,0x80000000,
    };

    for(i=0;i<32;i++)
    {
        if(n&arr[i])
        {
            count++;
        }
    }

    return count;
}

/*
执行结果:
通过
显示详情
执行用时 :4 ms, 在所有 C 提交中击败了86.99% 的用户
内存消耗 :6.8 MB, 在所有 C 提交中击败了6.02%的用户
*/
```

方法二: C 与操作特性。

```
int hammingweight(uint32_t n)
{
    int count = 0;
    while(n)
    {
        count++;
        n = n&(n-1);
    }

    return count;
}
```

```
/*
```

执行结果:

通过

[显示详情](#)

执行用时 :0 ms, 在所有 C 提交中击败了100.00% 的用户

内存消耗 :6.9 MB, 在所有 C 提交中击败了6.02%的用户

```
*/
```

AlimyBreak

2019.07.21