位运算-分治

题目描述：反转一个无符号32位二进制数

若要翻转一个二进制串，可以将其均分成左右两部分，对每部分递归执行翻转操作，然后将左半部分拼在右半部分的后面，即完成了翻转。

对于递归的最底层，我们需要交换所有奇偶位：

1. 取出所有奇数位和偶数位；
2. 将奇数位移到偶数位上，偶数位移到奇数位上。

类似地，对于倒数第二层，每两位分一组，按组号取出所有奇数组和偶数组，然后将奇数组移到偶数组上，偶数组移到奇数组上。以此类推。

```
class Solution {
private:
    const uint32_t M1 = 0x55555555; // 01010101010101010101010101010101
    const uint32_t M2 = 0x33333333; // 00110011001100110011001100110011
    const uint32_t M4 = 0x0f0f0f0f; // 00001111000011110000111100001111
    const uint32_t M8 = 0x00ff00ff; // 00000000111111110000000011111111

public:
    uint32_t reverseBits(uint32_t n) {
        n = n >> 1 & M1 | (n & M1) << 1;//分治的底层-反转奇偶数位
        n = n >> 2 & M2 | (n & M2) << 2;
        n = n >> 4 & M4 | (n & M4) << 4;
        n = n >> 8 & M8 | (n & M8) << 8;
        return n >> 16 | n << 16;
    }
};
```

作者：LeetCode-Solution

链接：https://leetcode-cn.com/problems/reverse-bits/solution/dian-dao-er-jin-zhi-wei-by-leetcode-solu-yhxz/
来源：力扣（LeetCode）

求一个二进制数中1的个数：

方法二：位运算优化

思路及解法

观察这个运算：$n \ \& \ (n-1)$，其运算结果恰为把 n 的二进制位中的最低位的 1 变为 0 之后的结果。

如：$6 \ \& \ (6-1) = 4, 6 = (110)_2, 4 = (100)_2$，运算结果 4 即为把 6 的二进制位中的最低位的 1 变为 0 之后的结果。

这样我们可以利用这个位运算的性质加速我们的检查过程，在实际代码中，我们不断让当前的 n 与 $n-1$ 做与运算，直到 n 变为 0 即可。因为每次运算会使得 n 的最低位的 1 被翻转，因此运算次数就等于 n 的二进制位中 1 的个数。

观察这个运算：n & (n-1)，其运算结果恰为把 n的二进制位中的最低位的1变为0之后的结果。

```
class Solution {
public:
    int hammingWeight(uint32_t n) {
        int ret = 0;
        while (n) {
            n &= n - 1;
            ret++;
        }
        return ret;
    }
};
```

作者：LeetCode-Solution
链接：https://leetcode-cn.com/problems/number-of-1-bits/solution/wei-1de-ge-shu-by-leetcode-solution-jnwf/
来源：力扣（LeetCode）