```swift
import Foundation
/// 剑指 Offer 15. 二进制中1的个数
/// 编写一个函数，输入是一个无符号整数(以二进制串的形式)
/// 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).）。
class Solution {
    /// 执行用时：0 ms, 在所有 Swift 提交中击败了 100.00% 的用户
    /// 内存消耗：13.1 MB, 在所有 Swift 提交中击败了 83.78% 的用户
    func hammingWeightSolution1(_ n: Int) -> Int {
        let bin_n = String(n, radix: 2)
        return bin_n.filter({ $0 == "1" }).count
    }
    /// 执行用时：4 ms, 在所有 Swift 提交中击败了 72.97% 的用户
    /// 内存消耗：13.4 MB, 在所有 Swift 提交中击败了 5.41% 的用户
    func hammingWeightSolution2(_ n: Int) -> Int {
        let bin_n = dec2Bin(n)
        return bin_n.filter({ $0 == "1" }).count
    }
    func dec2Bin(_ n: Int) -> String {
        var result = ""
        var _n = n
        while (_n != 0) {
            result += String(_n % 2)
            _n /= 2
        }
        return result
    }
    /// 执行用时：0 ms, 在所有 Swift 提交中击败了 100.00% 的用户
    /// 内存消耗：13.2 MB, 在所有 Swift 提交中击败了 35.14% 的用户
    func hammingWeightSolution3(_ n: Int) -> Int {
        var _n = n
        var counter: Int = 0
        while (_n != 0) {
            if (_n & 1 == 1) { counter += 1 }
            _n >>= 1
        }
        return counter
    }
    /// 执行用时：4 ms, 在所有 Swift 提交中击败了 72.97% 的用户
    /// 内存消耗：13.4 MB, 在所有 Swift 提交中击败了 5.41% 的用户
    func hammingWeight(_ n: Int) -> Int {
        var _n = n
        var counter: Int = 0
        while (_n != 0) {
            counter += 1
            _n &= (_n-1)
        }
        return counter
    }
}
```