```swift
import Foundation
/// 剑指 Offer 61. 扑克牌中的顺子
/// 从若干副扑克牌中随机抽 5 张牌，判断是不是一个顺子，即这5张牌是不是连续的。
/// 2～10为数字本身，A为1，J为11，Q为12，K为13，而大、小王为 0 ，可以看成任意数字。
/// A 不能视为 14。
class Solution {
    /// 执行用时：4 ms, 在所有 Swift 提交中击败了 77.14% 的用户
    /// 内存消耗：13.9 MB, 在所有 Swift 提交中击败了 48.57% 的用户
    /// 通过测试用例：204 / 204
    func isStraightSolution1(_ nums: [Int]) -> Bool {
        /// 先给数组排序
        let sortedNums: [Int] = nums.sorted(by: { $0 < $1 })
        /// 记录非0元素，0的个数最多为5
        var zeroNumberCount: Int = nums.filter({ $0 == 0 }).count
        if zeroNumberCount >= 4 { return true }
        /// 将非0元素取出
        var nonZeroNums: [Int] = sortedNums.filter({ $0 != 0 })
        let nonZeroNumsCount = nonZeroNums.count
        /// 遍历找寻相邻的数字
        for (id1, id2) in zip(0...nonZeroNumsCount-2, 1...nonZeroNumsCount-1) {
            /// 如果数字相同则不是顺子
            if nonZeroNums[id2] == nonZeroNums[id1] {
                return false
            }
            /// 计算相邻两个数的间隔，从0的个数中扣除该个数
            if nonZeroNums[id2] - nonZeroNums[id1] > 1 {
                zeroNumberCount -= (nonZeroNums[id2] - nonZeroNums[id1] - 1)
            }
            /// 若0不够了则说明无法形成顺子
            if zeroNumberCount < 0 { return false }
        }
        return true
    }

    func isStraight(_ nums: [Int]) -> Bool {
        var zeroNumCount: Int = 0
        let numLength = nums.count
        let _nums = nums.sorted(by: { $0 < $1 })
        for id in 0..<numLength-1 {
            if _nums[id] == 0 { zeroNumCount += 1 }
            else if _nums[id] == _nums[id+1] { return false }
        }
        /// 最大牌 - 最小牌小于 5 则可构成顺子
        return _nums[4] - _nums[zeroNumCount] < 5
    }
}
```