```swift
import Foundation
/// 剑指 Offer 40. 最小的k个数
/// 输入整数数组 arr ，找出其中最小的 k 个数。
/// 例如，输入4、5、1、6、2、7、3、8这8个数字，则最小的4个数字是1、2、3、4。
class Solution {
    /// 执行用时：280 ms, 在所有 Swift 提交中击败了 18.33% 的用户
    /// 内存消耗：14.6 MB, 在所有 Swift 提交中击败了 21.67% 的用户
    /// 通过测试用例：204 / 204
    func getLeastNumbersSolution1(_ arr: [Int], _ k: Int) -> [Int] {
        let length: Int = arr.count
        /// 比较剩下的元素与res列表里的元素
        var left: [Int] = arr.suffix(length-k)
        /// 从原始的数据中直接取出k个数作为结果
        var res: [Int] = arr.dropLast(length-k)
        for num in left {
            res = res.sorted(by: { $0 > $1 })
            if let _ = res.first {
                res[0] = res[0] > num ? num: res[0]
            }
        }
        return res
    }

    /// 执行用时：160 ms, 在所有 Swift 提交中击败了 35.00% 的用户
    /// 内存消耗：14.8 MB, 在所有 Swift 提交中击败了 6.67% 的用户
    /// 通过测试用例：204 / 204
    func quickSort(_ array: inout [Int], left: Int, right: Int) {
        if left >= right { return }
        var (i,j) = (left, right)
        while i < j {
            while (i < j) && (array[j] >= array[left]) { j -= 1 }
            while (i < j) && (array[i] <= array[left]) { i += 1 }
            (array[i], array[j]) = (array[j], array[i])
        }
        (array[left], array[i]) = (array[i], array[left])
        /// 此时`i`的位置与`j`相同，`i`为中间的元素，以`i`为中心将数组分为两个部分
        /// 利用`quickSort`递归分别排序左右两个数组即可
        quickSort(&array, left: left, right: i-1)
        quickSort(&array, left: i+1, right: right)
    }

    func getLeastNumbers(_ arr: [Int], _ k: Int) -> [Int] {
        var _array = arr
        let length = arr.count
        quickSort(&_array, left: 0, right: length-1)
        return _array.dropLast(length-k)
    }
}
```