```swift
//
//  main.swift
//  LeetCode
//
//  Created by Huang Runhua on 7/28/23.
//

import Foundation

class Solution {
    func mergeSort(_ nums: [Int]) -> [Int] {
        var _nums = nums
        divid(&_nums, 0, _nums.count-1)
        return _nums
    }
    /// 将序列从中间划分为子序列
    /// - Parameters:
    ///   - nums: 每次分割后的序列
    ///   - left: 序列的起始索引
    ///   - right: 序列的最终索引
    private func divid(_ nums: inout [Int], _ left: Int, _ right: Int) {
        if left >= right { return }
        /// 确认中心点索引
        let mid = left + (right - left)/2
        // MARK: 分割部分
        /// 将序列按中心点索引划分为左右两部分
        var leftSideNums: [Int] = Array(nums[left...mid])
        var rightSideNums: [Int] = Array(nums[mid+1...right])
        /// 对划分后的左右两部分再次进行划分
        divid(&leftSideNums, 0, leftSideNums.count-1)
        divid(&rightSideNums, 0, rightSideNums.count-1)
        // MARK: 合并部分
        /// 由于 divid 方法将序列划分为两部分，在调用 merge 函数后将两部分序列进行合并
        /// 需要将合并后的序列重新赋值给原来的 nums 序列
        nums = merge(leftSideNums, rightSideNums)
    }

    /// 将两个有序的数组进行合并元素，类似于将两个有序链表进行合并。
    /// - Parameters:
    ///   - list1: 第一个待合并的数组
    ///   - list2: 第二个待合并的数组
    /// - Returns: 合并后的数组
    private func merge(_ list1: [Int], _ list2: [Int]) -> [Int] {
        var (id1, id2, _list1, _list2): (Int, Int, [Int], [Int]) = (0, 0,
         list1,list2)
        var res: [Int] = []
        while id1 < list1.count && id2 < list2.count {
            if list1[id1] <= list2[id2] {
                res.append(list1[id1])
```

```
                _list1.remove(at: 0)
                id1 += 1
            } else {
                res.append(list2[id2])
                _list2.remove(at: 0)
                id2 += 1
            }
        }
        res.append(contentsOf: _list1.isEmpty ? _list2 : _list1)
        return res
    }
}

let solution = Solution()
var nums: [Int] = [9,8,7,6,5,4,3,2,1]
print(solution.mergeSort(nums))
```