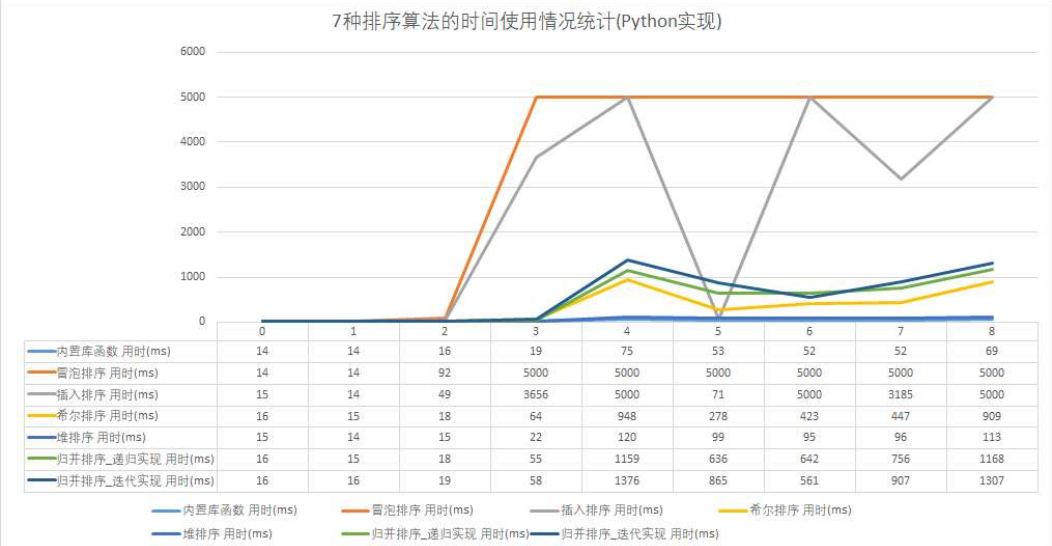
24 第九讲 作业记录（一）

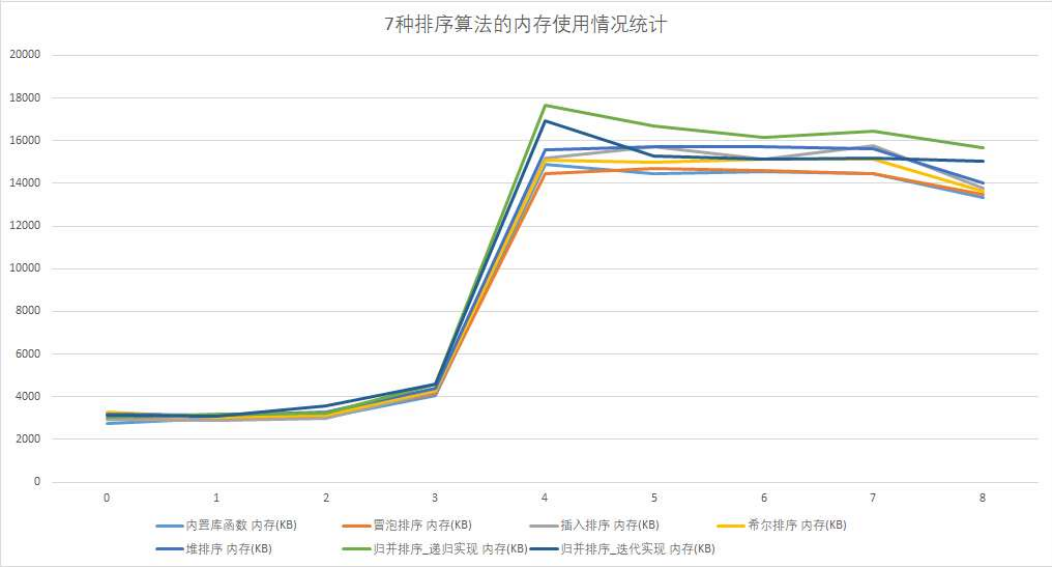
笔记本：	浙江大学《数据结构》		
创建时间：	2025/5/24 14:05	**更新时间：**	2025/5/24 18:17
作者：	panhengye@163.com		
URL：	https://pintia.cn/problem-sets/1873565885118418944/exam/problems/typ...		

09-排序1 排序
注意：在main函数中，只有归并排序_递归版本的接口与其他算法不同，这是由于递归函数特有性质决定的，并非设计不周

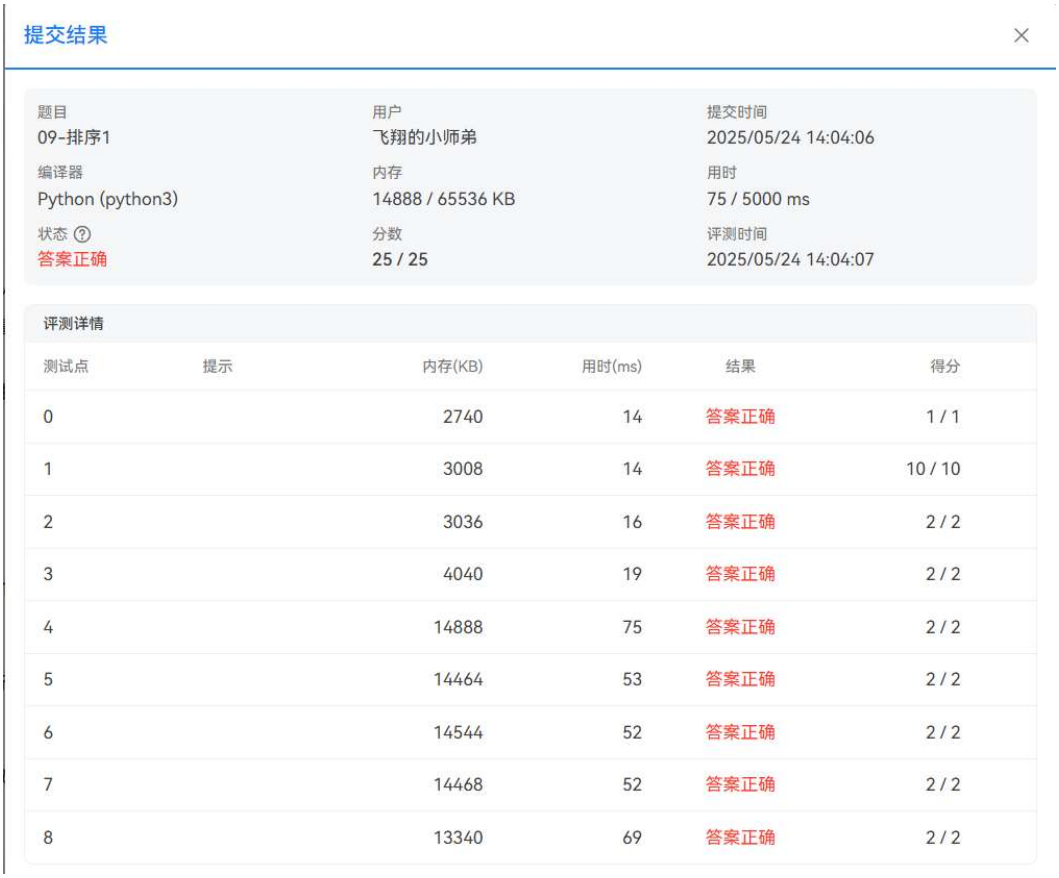
7种算法的性能比较



	0	1	2	3	4	5	6	7	8
内置库函数 用时(ms)	14	14	16	19	75	53	52	52	69
冒泡排序 用时(ms)	14	14	92	5000	5000	5000	5000	5000	5000
插入排序 用时(ms)	15	14	49	3656	5000	71	5000	3185	5000
希尔排序 用时(ms)	16	15	18	64	948	278	423	447	909
堆排序 用时(ms)	15	14	15	22	120	99	95	96	113
归并排序_递归实现 用时(ms)	16	15	18	55	1159	636	642	756	1168
归并排序_迭代实现 用时(ms)	16	16	19	58	1376	865	561	907	1307



算法1：利用Python内置库函数

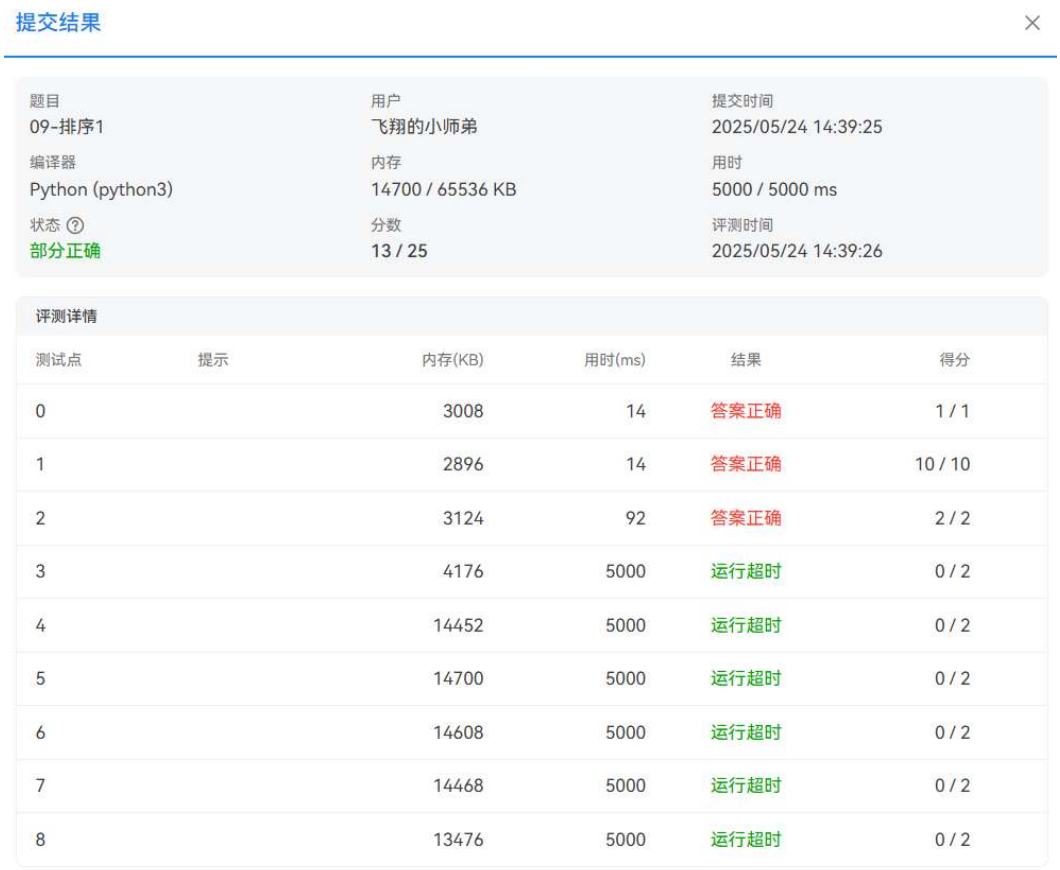


代码记录

```
def main():     # 处理输入     n = int(input())     nums = list(map(int,
input().split()))     # 排序     nums.sort()     # 输出     print('
'.join(map(str, nums))) if __name__ == '__main__':     main()
```

由于输入输出部分是一样的，所以为了节省篇幅，下面只呈现算法实现

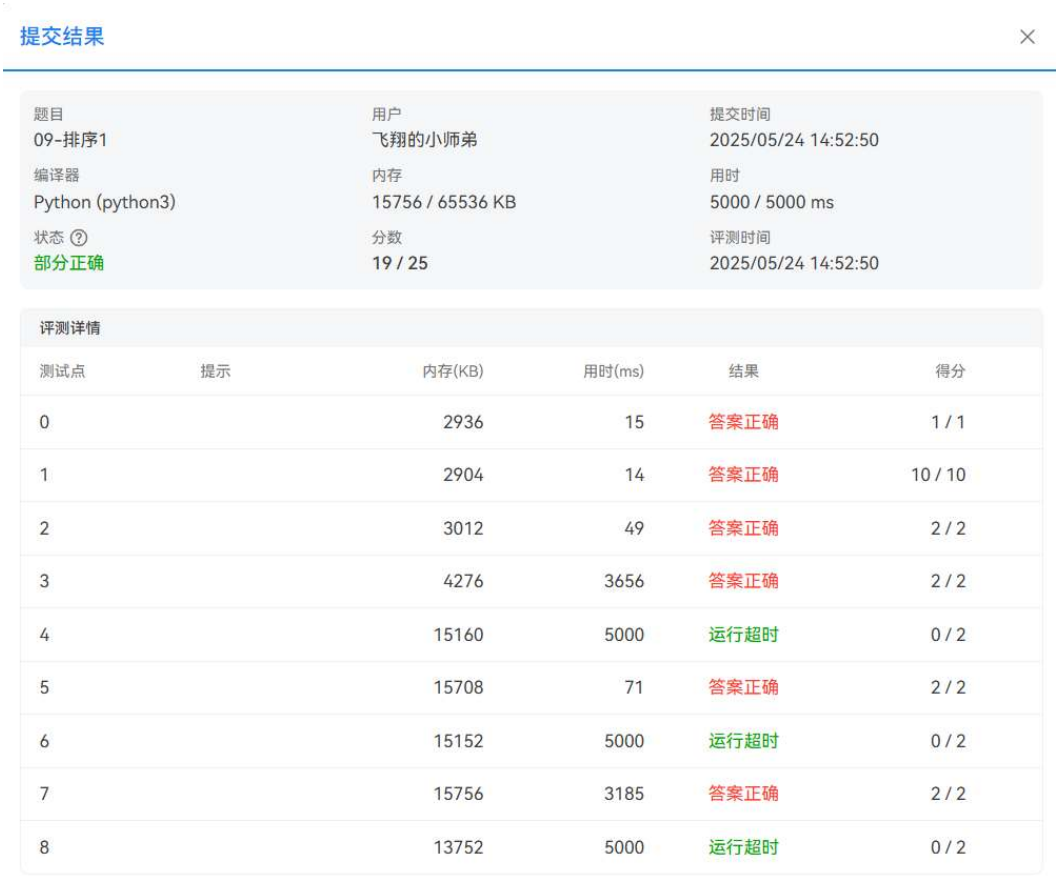
冒泡排序



代码记录

```
# 冒牌排序 def bubble_sort(nums):     """     :param nums: 待排序列。
:return: 返回排好的序列。     """     n = len(nums)     for i in range(n):
    for j in range(0, n - i -1):             if nums[j] > nums[j + 1]:
          nums[j], nums[j + 1] = nums[j + 1], nums[j]     return nums
```

插入排序



提交结果

题目	用户	提交时间
09-排序1	飞翔的小师弟	2025/05/24 14:52:50
编译器	内存	用时
Python (python3)	15756 / 65536 KB	5000 / 5000 ms
状态	分数	评测时间
部分正确	19 / 25	2025/05/24 14:52:50

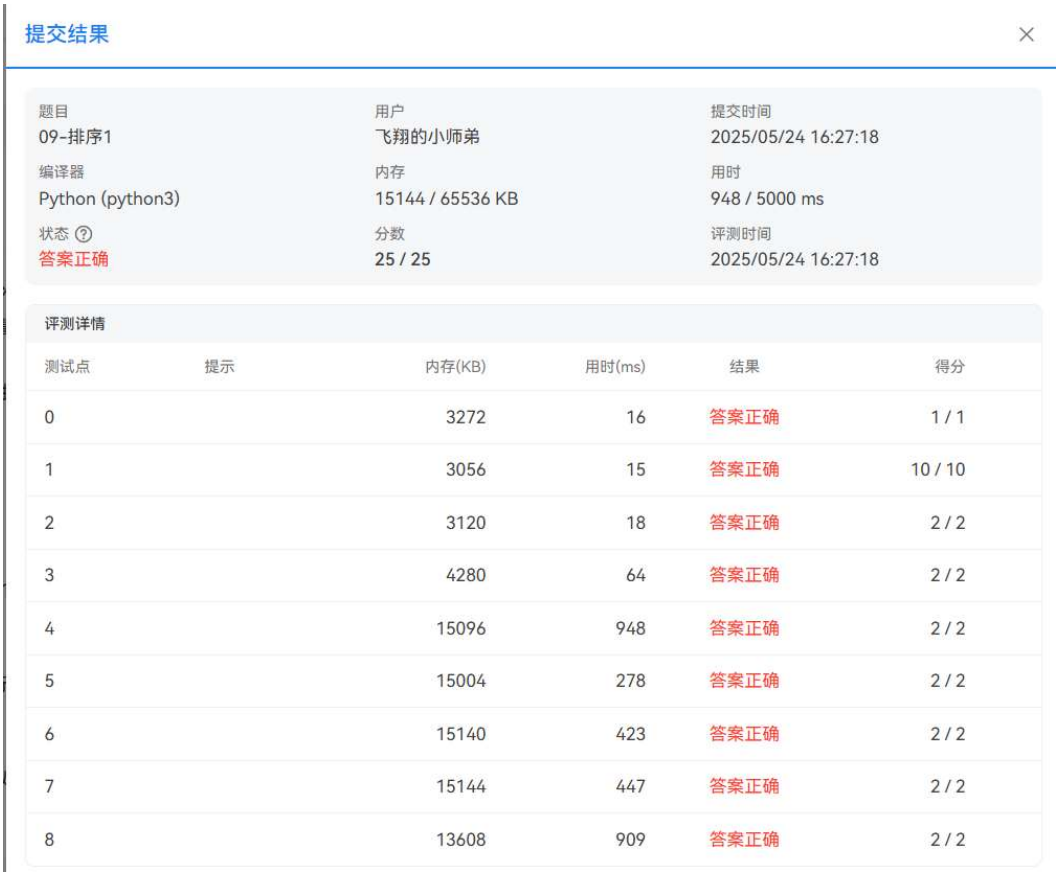
评测详情

测试点	提示	内存(KB)	用时(ms)	结果	得分
0		2936	15	答案正确	1 / 1
1		2904	14	答案正确	10 / 10
2		3012	49	答案正确	2 / 2
3		4276	3656	答案正确	2 / 2
4		15160	5000	运行超时	0 / 2
5		15708	71	答案正确	2 / 2
6		15152	5000	运行超时	0 / 2
7		15756	3185	答案正确	2 / 2
8		13752	5000	运行超时	0 / 2

代码记录

```
# 插入排序 def insertion_sort(nums):     """     :param nums: 待排序列。
:return: 返回排好的序列。      """     for i in range(1, len(nums)):
key = nums[i]         j = i - 1         while j >= 0 and key < nums[j]:
        nums[j + 1] = nums[j]             j = j - 1         nums[j + 1] =
key     return nums
```

希尔排序



提交结果

题目	用户	提交时间
09-排序1	飞翔的小师弟	2025/05/24 16:27:18
编译器	内存	用时
Python (python3)	15144 / 65536 KB	948 / 5000 ms
状态	分数	评测时间
答案正确	25 / 25	2025/05/24 16:27:18

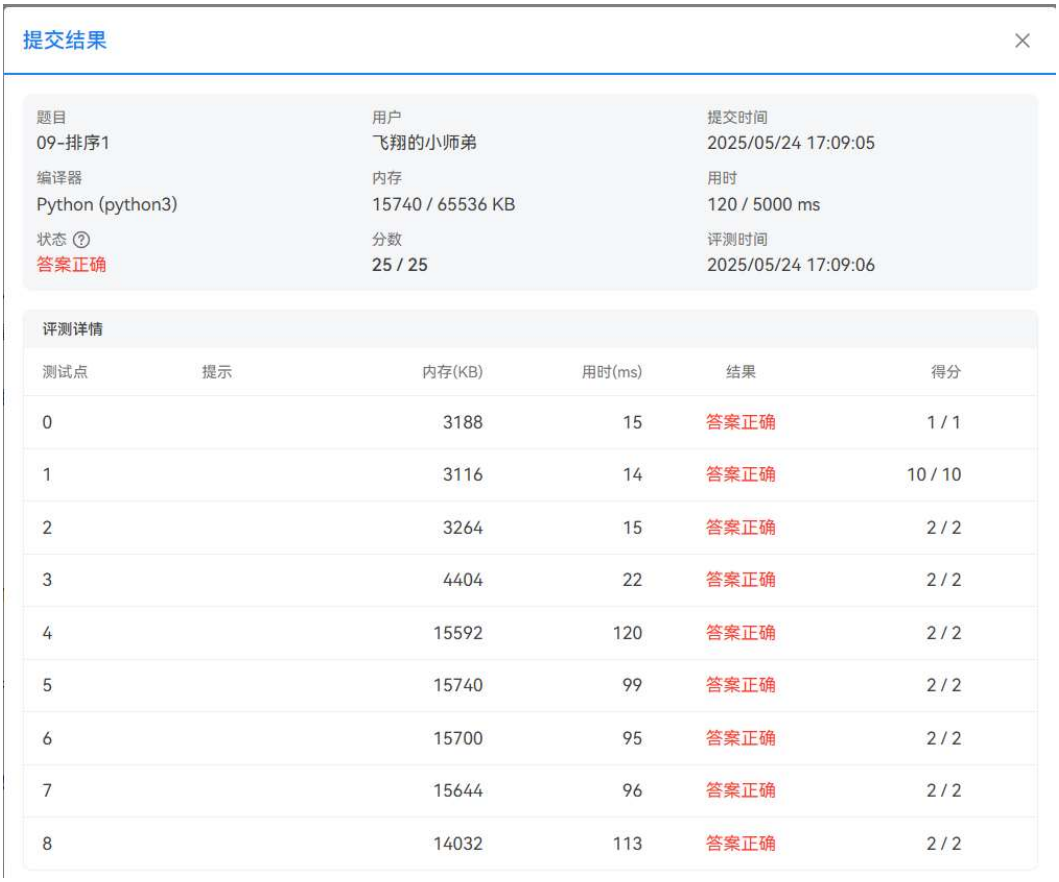
评测详情

测试点	提示	内存(KB)	用时(ms)	结果	得分
0		3272	16	答案正确	1 / 1
1		3056	15	答案正确	10 / 10
2		3120	18	答案正确	2 / 2
3		4280	64	答案正确	2 / 2
4		15096	948	答案正确	2 / 2
5		15004	278	答案正确	2 / 2
6		15140	423	答案正确	2 / 2
7		15144	447	答案正确	2 / 2
8		13608	909	答案正确	2 / 2

代码记录

```
# 希尔排序 def shell_sort(nums):     """     :param nums: 待排序列。
:return: 返回排好的序列。     """     n = len(nums)     gap = n // 2 # 初始步长
    while gap > 0:         for i in range(gap, n):             temp =
nums[i]             j = i             # 对每个步长进行排序             while j
>= gap and nums[j - gap] > temp:                 nums[j] = nums[j - gap] # 后
移元素                 j -= gap # 按步长回退             nums[j] = temp
  gap = gap // 2 # 减小步长         return nums
```

堆排序



提交结果

题目	用户	提交时间
09-排序1	飞翔的小师弟	2025/05/24 17:09:05
编译器	内存	用时
Python (python3)	15740 / 65536 KB	120 / 5000 ms
状态	分数	评测时间
答案正确	25 / 25	2025/05/24 17:09:06

评测详情

测试点	提示	内存(KB)	用时(ms)	结果	得分
0		3188	15	答案正确	1 / 1
1		3116	14	答案正确	10 / 10
2		3264	15	答案正确	2 / 2
3		4404	22	答案正确	2 / 2
4		15592	120	答案正确	2 / 2
5		15740	99	答案正确	2 / 2
6		15700	95	答案正确	2 / 2
7		15644	96	答案正确	2 / 2
8		14032	113	答案正确	2 / 2

代码记录

```
# 堆排序 def heap_sort(nums):     """     :param nums: 待排序列。     :return: 返回排好的序列。     """     # 构建最小堆     heapq.heapify(nums)  # 原地转换为最小堆，时间复杂度O(N)     # 逐个弹出最小元素     sorted_nums = []     while nums:         sorted_nums.append(heapq.heappop(nums))     # 将排序后的结果复制回原数组     nums[:] = sorted_nums     return nums
```

要点解释

1. 为什么要返回原数组，而不是返回sorted_nums呢?
    1. 保持一致性：这种设计模式使得所有排序函数的行为保持一致，便于使用和维护
    2. 内存效率：返回原数组意味着我们不需要额外的内存空间来存储排序结果
    3. 原地排序的约定：我们的实现遵循了原地排序的约定，这样更符合Python的编程习惯
    4. 避免混淆：如果返回sorted_nums，可能会让调用者误以为原数组没有被修改
2. 为什么是“nums[:] = sorted_nums”而不是“nums = sorted_nums”
    1. nums[:]表示选择nums列表的所有元素
    2. nums = sorted_nums：这会创建一个新的引用，原来的nums变量会指向新的列表
    3. 实际效果如下：

```
# 假设我们有两个列表     nums = [3, 1, 4, 2]     sorted_nums = [1, 2, 3, 4]     # 使用 nums[:] = sorted_nums     nums[:] = sorted_nums     # 现在 nums 的内容变成了 [1, 2, 3, 4]     # 但 nums 的引用（内存地址）没有改变
```

错误记录

```python
# 堆排序
def heap_sort(nums):

    """
    :param nums: 待排序列。
    :return: 返回排好的序列。
    """
    # 构建最小堆
    heap = heapq.heapify(nums)  # 原地转换为最小堆，时间复杂度O(N)
    heapq.heapify(nums)  # 原地转换为最小堆，时间复杂度O(N)

    # 逐个弹出最小元素
    sorted_nums = []
    while heap:
        sorted_nums.append(heapq.heappop(heap))
    while nums:
        sorted_nums.append(heapq.heappop(nums))

    return heap
    # 将排序后的结果复制回原数组
    nums[:] = sorted_nums
    return nums
```

Reject Ctrl+N Accept Ctrl+Shift+Y

错误原因：
使用heapq.heapify(nums)后，heap变量实际上并没有被赋值，因为heapify是原地操作，返回值为None
这是因为heapq执行了原地操作（In-place Operation）

归并：递归实现

提交结果

题目	用户	提交时间
09-排序1	飞翔的小师弟	2025/05/24 17:35:40
编译器	内存	用时
Python (python3)	17668 / 65536 KB	1168 / 5000 ms
状态	分数	评测时间
答案正确	25 / 25	2025/05/24 17:35:41

评测详情

测试点	提示	内存(KB)	用时(ms)	结果	得分
0		3060	16	答案正确	1 / 1
1		3188	15	答案正确	10 / 10
2		3232	18	答案正确	2 / 2
3		4608	55	答案正确	2 / 2
4		17668	1159	答案正确	2 / 2
5		16688	636	答案正确	2 / 2
6		16156	642	答案正确	2 / 2
7		16456	756	答案正确	2 / 2
8		15684	1168	答案正确	2 / 2

代码记录

```
# 归并排序：递归实现     def merge_sort_recursive(nums):     """     :param
nums: 待排序列。     :return: 返回排好的序列。     """     if len(nums) <= 1:
      return nums         mid = len(nums) // 2     left =
merge_sort_recursive(nums[:mid])     right =
merge_sort_recursive(nums[mid:])         return merge(left, right) def
merge(left, right):     """     :param left: 左半部分，已排序。     :param
```

```
right: 右半部分，已排序。      :return: 返回合并后的序列。      """      result =
[]     while left and right:          if left[0] <= right[0]:
result.append(left.pop(0))         else:
result.append(right.pop(0))          # 处理剩余元素      result.extend(left)
  result.extend(right)     return result def main():     # 处理输入      n =
int(input())     nums = list(map(int, input().split()))     # 归并排序：递归实
现     nums = merge_sort_recursive(nums)     # 输出     print('
'.join(map(str, nums))) if __name__ == '__main__':     main()
```

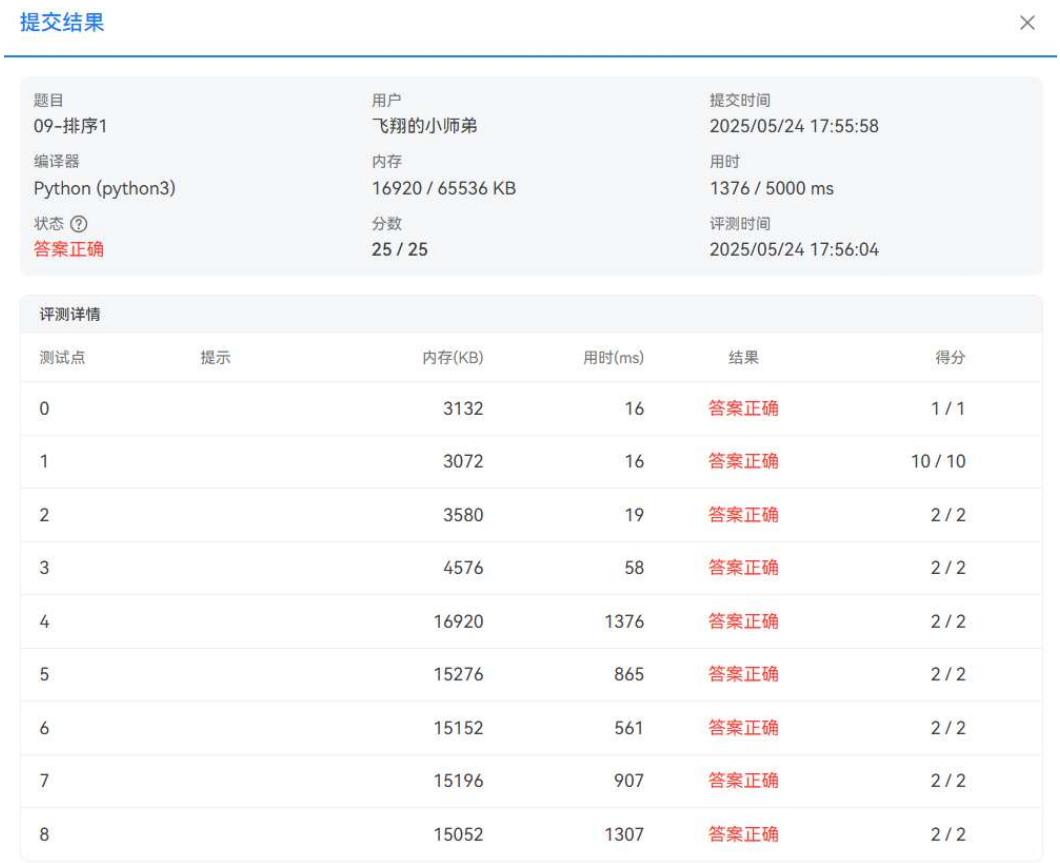
要点记录

nums[:mid] 中，: 前面的数字表示起始位置（这里是空，表示从开始）
具体含义：

- 当 mid = len(nums) // 2 时
- nums[:mid] 表示获取 nums 列表从开始到 mid 位置（不包含 mid）的所有元素
- 相当于获取列表的前半部分

注意：nums = merge_sort_recursive(nums)，归并排序的接口和其他算法不一样

归并：迭代实现



提交结果

题目	用户	提交时间
09-排序1	飞翔的小师弟	2025/05/24 17:55:58
编译器	内存	用时
Python (python3)	16920 / 65536 KB	1376 / 5000 ms
状态	分数	评测时间
答案正确	25 / 25	2025/05/24 17:56:04

评测详情

测试点	提示	内存(KB)	用时(ms)	结果	得分
0		3132	16	答案正确	1 / 1
1		3072	16	答案正确	10 / 10
2		3580	19	答案正确	2 / 2
3		4576	58	答案正确	2 / 2
4		16920	1376	答案正确	2 / 2
5		15276	865	答案正确	2 / 2
6		15152	561	答案正确	2 / 2
7		15196	907	答案正确	2 / 2
8		15052	1307	答案正确	2 / 2

代码记录

```
# 归并排序：非递归实现 def merge_sort_iterative(nums):     """     :param
nums: 待排序列。      :return: 返回排好的序列。      """     n = len(nums)     #
初始步长为1     step = 1     while step < n:         for i in range(0, n,
step * 2):             left = nums[i:i + step] # 左半部分             right =
nums[i + step:i + step * 2] # 右半部分             nums[i:i + step * 2] =
merge(left, right) # 合并         step *= 2         return nums
```