

G4-14 课后练习 —— 选择排序

姓名： ____ 日期： ____

>

回家先念三遍口诀：

"选择排序选最矮，放到前排再继续！"

"插入排序像摸牌，新牌插入合适位！"

一、基础练习（必做）

1.1 填空题

- (1) 选择排序每轮要找出 ____ 的元素，放到 ____。
- (2) 插入排序每轮把当前元素插入到 ____ 中。
- (3) 选择排序在每一轮中，先假设 ____ 是最小的，然后在 ____ 中找更小的。
- (4) 插入排序中，当新元素比前面的元素小，前面的元素要 ____ 一位。
- (5) 选择排序是 ____ （稳定/不稳定）排序，插入排序是 ____ （稳定/不稳定）排序。

1.2 选择题

- (1) 选择排序的核心操作是什么？
 - A. 相邻比较交换
 - B. 找最小的交换到前面
 - C. 把元素插入已排好序列
 - D. 随机打乱再排
- (2) 插入排序的核心操作是什么？
 - A. 相邻比较交换
 - B. 找最小的交换到前面
 - C. 把元素插入已排好序列

- D. 随机打乱再排

(3) 如果有6个数，选择排序要找几轮最小值？

- A. 6轮
- B. 5轮
- C. 4轮
- D. 3轮

(4) 插入排序从第几个元素开始"摸牌"？

- A. 第0个
- B. 第1个
- C. 第2个
- D. 最后一个

(5) 下面哪种情况插入排序比选择排序快得多？

- A. 数据完全随机
- B. 数据已经基本有序
- C. 数据是完全逆序
- D. 数据全部相等

1.3 模拟排序题

题目1：用选择排序给 「[8, 3, 6, 1, 7]」 排序

轮次	找到的最小值	跟谁交换	交换后的数组
第1轮	__	跟第1位的__交换	[_,_,_,_,_]
第2轮	__	跟第2位的__交换	[_,_,_,_,_]
第3轮	__	跟第3位的__交换	[_,_,_,_,_]
第4轮	__	跟第4位的__交换	[_,_,_,_,_]

最终结果：[_,_,_,_,_]

题目2：用插入排序给 「[8, 3, 6, 1, 7]」 排序

步数	摸到的牌	比较过程	插入位置	当前数组
i=1	3	3<8后移	第0位	[_,_,_,_,_]
i=2	6	6<8后移, 6>3停	第1位	[_,_,_,_,_]
i=3	1			[_,_,_,_,_]
i=4	7			[_,_,_,_,_]

最终结果：[_,_,_,_,_]

二、编程练习（必做）

2.1 写选择排序函数

```
def my_selection_sort(arr):  
    # 在这里写你的选择排序  
    # 提示：外循环n-1轮，内循环从i+1到n找最小  
    pass
```

```
# 测试  
test = [9, 4, 7, 2, 5, 1]  
print("排序前：", test)  
result = my_selection_sort(test)  
print("排序后：", result)
```

2.2 写插入排序函数

```
def my_insertion_sort(arr):  
    # 在这里写你的插入排序  
    # 提示：从第1个开始摸牌，往前比较插入  
    pass
```

```
# 测试  
test = [9, 4, 7, 2, 5, 1]  
print("排序前：", test)  
result = my_insertion_sort(test)  
print("排序后：", result)
```

2.3 对比实验

用下面的代码测试三种排序的性能差异：

```
import time  
import random
```

```
# 生成100个随机数  
data = [random.randint(1, 1000) for _ in range(100)]
```

```
# 复制三份  
data1 = data.copy()  
data2 = data.copy()  
data3 = data.copy()
```

```
# 测试冒泡排序  
start = time.time()  
bubble_sort(data1) # 假设你写了冒泡排序
```

```
bubble_time = time.time() - start
```

```
# 测试选择排序
start = time.time()
selection_sort(data2)
selection_time = time.time() - start
```

```
# 测试插入排序
start = time.time()
insertion_sort(data3)
insertion_time = time.time() - start
```

```
print(f"冒泡排序: {bubble_time:.4f}秒")
print(f"选择排序: {selection_time:.4f}秒")
print(f"插入排序: {insertion_time:.4f}秒")
```

记录结果：

- 冒泡排序： ____ 秒
- 选择排序： ____ 秒
- 插入排序： ____ 秒
- 谁最快？ ____
- 为什么？ ____

三、进阶练习（选做）

3.1 修改排序顺序

把选择排序和插入排序改成从大到小排序（降序）：

```
def selection_sort_desc(arr):
    """从大到小排序（降序）"""
    n = len(arr)
    for i in range(n-1):
        max_idx = ____ # 改成找最大的!
        # ... 需要改哪里?
    return arr
```

```
def insertion_sort_desc(arr):
    """从大到小排序（降序）"""
    n = len(arr)
    for i in range(1, n):
        key = arr[i]
        j = i - 1
        while j >= 0 and arr[____] < key: # 需要改什么?
```

```
arr[j + 1] = arr[j]
j -= 1
arr[j + 1] = key
return arr
```

3.2 统计排序信息

写一个增强版排序函数，除了排序还返回统计信息：

```
def selection_sort_with_stats(arr):
    """
    返回：(排序后的数组, 比较次数, 交换次数)
    """
    # 在这里实现
    pass

# 测试
arr = [5, 3, 1, 4, 2]
sorted_arr, comp, swaps = selection_sort_with_stats(arr)
print(f"结果: {sorted_arr}")
print(f"比较了{comp}次, 交换了{swaps}次")
```

3.3 生活中的排序

场景	用哪种排序方法?	为什么?
整理扑克牌		
老师按成绩排名		
图书馆整理书架		
自己编一个例子: _____		

四、挑战题（超级选做）

4.1 插入排序的"二分优化"

普通的插入排序是线性比较（一个一个比），但我们可以用二分查找来快速找到插入位置！

```
import bisect # Python内置的二分查找

def insertion_sort_binary(arr):
    """
    用二分查找优化的插入排序
    """
```

```
# 研究一下 bisect 模块的用法
# 提示: bisect_left 可以找到插入位置
pass
```

4.2 一题多解：三种排序写在一起

写一个函数，接收参数选择用哪种排序：

```
def sort_array(arr, method="bubble"):
    """
    method: "bubble" / "selection" / "insertion"
    根据method选择不同的排序方法
    """
    if method == "bubble":
        # 冒泡排序
        pass
    elif method == "selection":
        # 选择排序
        pass
    elif method == "insertion":
        # 插入排序
        pass
    return arr
```

```
# 测试
data = [5, 3, 1, 4, 2]
print(sort_array(data, "bubble")) # 冒泡
print(sort_array(data, "selection")) # 选择
print(sort_array(data, "insertion")) # 插入
```

孩子完成了以上练习（请家长在完成的打√）：

>

☐ 基础练习（1.1~1.3）

☐ 编程练习（2.1~2.3）

☐ 进阶练习（3.1~3.3）

☐ 挑战题（4.1~4.2）

>

>

孩子的话：今天学了两种排序，我觉得选择排序像_____，

插入排序像_____。我更喜欢_____排序，

因为_____。

自我评价： ☐ 全都会了 ☐ 大部分会了 ☐ 还要再练练