分工

30322張家銘:全部

人數	總分	平均配分
1	100	100

快速排序法
QuickSort

1. 選擇一個「基準點（pivot)」。
2. 把所有比 pivot 小的放左邊，比 pivot 大的放右邊。
3. 對左右兩邊的子列表「重複相同動作（遞迴)」。
4. 最後把左邊 + pivot + 右邊合併成完整排序結果。

```python
def quick_sort(data):
    """
    實作快速排序法 (Quick Sort)。
    :param data: 待排序的數字列表
    :return: 排序後的數字列表
    """
    # 【任務】請在此處實作快速排序法的演算法。
    # 提示：這是一個遞迴函式，你需要選擇一個基準點 (pivot)。
    # 為了不修改到原始資料，建議複製一份再進行排序。
    sorted_data = data[:] # 複製列表
    # ... 你的程式碼 ...
    def _quick_sort(lst):
        if len(lst) <= 1:
            return lst
        pivot = lst[len(lst) // 2]
        left = [x for x in lst if x < pivot]
        middle = [x for x in lst if x == pivot]
        right = [x for x in lst if x > pivot]
        return _quick_sort(left) + middle + _quick_sort(right)

    sorted_data = _quick_sort(sorted_data)
    print("（快速排序法）已排序完成。")
    return sorted_data
```

選擇排序法
Selection Sort

1. 從整個列表中找出最小的數字。
2. 把它和「第一個位置」交換。
3. 接著從剩下的未排序部分（第 2 位到最後）再找最小的。
4. 重複直到整個列表排序完。

```python
def selection_sort(data):
    """
    實作選擇排序法 (Selection Sort)。
    :param data: 待排序的數字列表
    :return: 排序後的數字列表
    """
    # 【任務】請在此處實作選擇排序法的演算法。
    # 提示：每一輪都找到未排序部分中的最小值，然後放到正確位置。
    # 為了不修改到原始資料，建議複製一份再進行排序。
    sorted_data = data[:] # 複製列表
    # ... 你的程式碼 ...
    sorted_data = data[:]
    n = len(sorted_data)
    for i in range(n - 1):
        min_index = i
        for j in range(i + 1, n):
            if sorted_data[j] < sorted_data[min_index]:
                min_index = j
        sorted_data[i], sorted_data[min_index] = sorted_data[min_index], sorted_data[i]
    print("（選擇排序法）已排序完成。")
    return sorted_data
```

線性搜尋法
Linear Search

逐一從頭到尾檢查每個元素
看它是否等於要找的目標值

```python
def linear_search(data, value):
    """
    實作線性搜尋法 (Linear Search)。
    :param data: 數字列表
    :param value: 要搜尋的數字
    :return: 找到的數字索引(int)或 None
    """
    # 【任務】請在此處實作線性搜尋法的演算法。
    # 提示：從頭到尾一個一個比對。
    print(f"（此處應為線性搜尋法）正在搜尋數字 {value}...")
    # ... 你的程式碼 ...
    for i in range(len(data)):
        if data[i] == value:
            return i
    return None
```

二元搜尋法
Binary Search

前提：資料必須已經排序

1. 取中間元素 mid
2. 若 data[mid] == target → 找到
   若 data[mid] < target → 在右半部繼續找
   若 data[mid] > target → 在左半部繼續找
3. 重複直到找到或發現沒有此值

```python
def binary_search(data, value):
    """
    實作二元搜尋法 (Binary Search)。
    :param data: **已排序好**的數字列表
    :param value: 要搜尋的數字
    :return: 找到的數字索引(int)或 None
    """
    # 【任務】請在此處實作二元搜尋法的演算法。
    # 提示：使用這個函式前，資料必須先排序好。
    # 提示：需要 low, high, mid 三個指標。
    print(f"（此處應為二元搜尋法）正在搜尋數字 {value}...")
    # ... 你的程式碼 ...

    low, high = 0, len(data) - 1
    while low <= high:
        mid = (low + high) // 2
        if data[mid] == value:
            return mid
        elif data[mid] < value:
            low = mid + 1
        else:
            high = mid - 1
    return None
```

進階挑戰任務

```python
def find_top_n(data, n):
    """
    找出前 N 個最大的數字。
    :param data: 數字列表
    :param n: 要找出的數量
    """
    #  【進階挑戰任務】
    # 1. 使用你實作的任一排序法，進行降冪排序（由大到小）。
    #    (你可能需要稍微修改你的排序函式，讓它可以支援降冪排序)
    # 2. 取出前 n 個數字並印出。
    print(f"（此處應為找出前 N 大數字）正在找出前 {n} 個最大的數字...")
    # ... 你的程式碼 ...
    sorted_desc = quick_sort(data)[::-1]  # 使用快速排序後反轉為降冪
    top_n = sorted_desc[:n]
    print(f"前 {n} 個最大數字為：{top_n}")
    return top_n
```

```
===== 選擇執行的動作 =====
1. 快速排序 (Quick Sort)
2. 選擇排序 (Selection Sort)
3. 線性搜尋 (Linear Search)
4. 二元搜尋 (Binary Search)
--- 進階挑戰 ---
5. 找出前 N 個最大的數字
0. 離開程式
請輸入您的選擇：5
請輸入要找出前幾名：10
（此處應為找出前 N 大數字）正在找出前 10 個最大的數字...
（快速排序法）已排序完成。
前 10 個最大數字為：[99987, 99967, 99967, 99963, 99962, 99957, 99928, 99909, 99908, 99902]
請按Enter繼續...
```