113 年公務人員高等考試三級考試試題

類科：資訊處理
科目：資料結構

一、㈠若有 200 人，其中一個人開始打電話給兩個人。隨後，每個接到電話的人都會打電話給另外兩個尚沒有接到電話的人。請問總共會撥打多少通電話？有多少人不會打電話？（無推導過程不給分）（10 分）

㈡若一個二元樹其前序追蹤順序（Preorder Traversal）及後序追蹤順序（Postorder Traversal）分別如下，請問此樹是否唯一？並請列出此二元樹的中序追蹤順序（Inorder Traversal）。（無推導過程不給分）（15 分）

前序追蹤順序：T, S, R, F, D, I, H, E, Z, G, M, L, J, N, Q
後序追蹤順序：F, I, H, D, R, Z, G, E, S, J, N, L, Q, M, T

1. 考題難易：★★★☆☆
2. 破題關鍵：本題為二元樹應用題，需要熟悉二元樹基本概念與反推操作始可作答。

【擬答】：

㈠

1. 撥電話過程會產生一棵二元樹，有 200 個節點，撥打電話數即為此二元樹的邊數，根據樹中邊數為節點數-1，因此總共撥了 199 通電話。

2. 不會打電話的人就是二元樹中沒有子節點的節點，即 n0，若設打 1 通電話的人有 n1 人，打 2 通電話的人有 n2 人，
因此 n1+2*n2=199，n1=199-2*n2，n1=1(當 n2=99),3(當 n2=98)…199(當 n2=0)
此時因為 n0+n1+n2=200，因此 n0=100(當 n1=1,n2=99),99(當 n1=3,n2=98)…1(當 n1=199,n2=0)
即不會打電話的人數與打 1 通電話、打 2 通電話人數有關)，最多 100 人，最少 1 人。

㈡

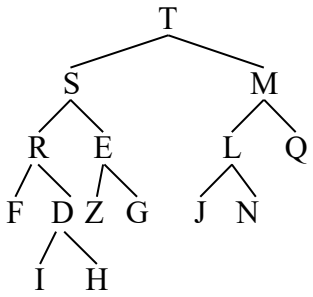
1. 推斷樹的結構過程如下：

⑴根據前序追蹤順序的第一個元素為根節點：T；根據後序追蹤順序的最後一個元素為根節點：T。兩者相同，故確定跟節點為 T。

⑵找出左子樹和右子樹的節點劃分：
從前序追蹤順序中找到根節點 T 後面的第一個元素 S，這是 T 的左子樹的根節點，接著到後序追蹤順序中以 S 來分隔，F, I, H, D, R, Z, G, E, S 為 T 的左子樹節點；J, N, L, Q, M 為 T 的右子樹節點。在後序追蹤順序中找到根節點 T 前面的最後一個元素 M，這是 T 的右子樹的根節點，接著到前序追蹤順序中以 M 來分隔，M, L, J, N, Q 為 T 的右子樹節點；S, R, F, D, I, H, E, Z, G 為 T 的左子樹節點。上列兩者推論並無矛盾，確認 S 是 T 的左子樹的根節點，M 是 T 的右子樹的根節點。

⑶重複步驟來劃分左右子樹：
從前序追蹤順序(S, R, F, D, I, H, E, Z, G)中的 S 後面找到 R，並在後序追蹤順序(F, I, H, D, R, Z, G, E, S)中的 S 前面找到 E，因此 S 的左子樹根節點為 R，S 的右子樹根節點為 E，且再用 *2.* 的步驟驗證如此分隔是否有矛盾，可得 S 的左子樹節點有 F, I, H, D, R；S 的右子樹節點有 Z, G, E。同樣對 T 的右子樹分割，可得 M 的左子樹樹根為 L，M 的右子樹樹根為 Q；M 的左子樹節點為 L, J, N；M 的右子樹節點僅有 Q。

⑷持續遞迴劃分，直到葉子節點：重複上述步驟，直到每個子樹都劃分完畢。可得下列二元樹



由於上面樹中無僅有 1 個子節點的節點，故此樹為唯一。

2. 中序追蹤順序：

根據推斷出的樹結構，其中序追蹤順序為：

F,R,I,D,H,S,Z,E,G,T,J,L,N,M,Q

二、㈠快速排序法（Quick Sort）最壞的情況下所需的時間複雜度（Time Complexity）為 O(n2)，請說明是在何種情況下造成？（10 分）

㈡請列出其最壞的時間複雜度為 O(n2)的推導過程。（15 分）

1. 考題難易：★★☆☆☆
2. 破題關鍵：本題為快速排序基本概念題，了解快速排序操作過程即可作答。

【擬答】：

㈠快速排序法在最壞情況下的時間複雜度為 O(n^2)，當且僅當每次選擇的基準元素都是當前子陣列中的最大或最小元素時，導致每次分割僅能減少一個元素。具體來說，這種情況通常發生在以下兩種情形之一：

1. **陣列已經排序好或反向排序：**如果快速排序的基準元素每次都是當前子陣列中的最大或最小值，那麼每次分割後，只能減少一個元素，導致分割的子陣列大小縮小非常緩慢，從而使得遞迴深度達到 O(n)，時間複雜度為 O(n^2)。
2. **基準元素選擇不當：**如果基準元素的選擇導致每次分割的左右子陣列大小差異非常大，例如每次選擇的基準元素都是當前子陣列中的極端值（最大或最小），同樣會導致遞迴深度增加，時間複雜度為 O(n^2)。

在這些情況下，快速排序的效率會退化到 0(n2)，這是因為每次分割的效果並未明顯減少問題的規模，而是僅稍稍減少，導致整體的排序過程變得非常緩慢。

㈡快速排序法在最壞情況下的時間複雜度 O(n^2)，可以透過以下推導過程來理解：

1. 選擇基準元素：快速排序每次選擇一個基準元素（pivot），通常可以是第一個元素、最後一個元素或者隨機選擇。
2. 分割操作：根據所選擇的基準元素，將陣列劃分為兩部分：小於基準元素的元素和大於基準元素的元素。
3. 最壞情況下的分割：如果每次分割操作後，基準元素所在的位置都不是中間位置，而是分割後的子陣列的最小或最大位置，即基準元素選擇不當。
4. 每次遞迴的規模：在最壞情況下，每次遞迴的規模沒有有效地減少。例如，如果每次選擇的基準元素都是當前子陣列中的最大或最小值，分割後的左右子陣列大小差距非常大。
5. 遞迴深度：由於每次分割沒有明顯地減少問題的規模，遞迴深度會增加到 O(n) 層次，每層次的工作量為 O(n)。
6. 時間複雜度分析：在最壞情況下，遞迴深度為 O(n)，且每層次的分割操作需遍歷整個子陣列，因此時間複雜度為 O(n)×O(n)=O(n^2)。

三、請使用虛擬碼（Pseudo Code）或任何程式語言，完成下列問題：

㈠撰寫二元搜尋（Binary Search）的遞迴及非遞迴程式。（20 分）

㈡推導二元搜尋的時間複雜度（Time Complexity）。（5 分）

1.考題難易：★★☆☆☆

2.破題關鍵：本題為二元搜尋程式實'做題，了解二元搜尋基本操作即可作答。

【擬答】：

㈠下面二元搜尋（Binary Search）的程式用 python 撰寫

1.遞迴版本的二元搜尋

```
def binary_search_recursive(arr, left, right, x):
    if right >= left:
        mid = left + (right - left) // 2

        # 如果目標值正好在中間
        if arr[mid] == x:
            return mid
        # 如果目標值比中間值小，則在左半邊搜尋
        elif arr[mid] > x:
            return binary_search_recursive(arr, left, mid - 1, x)
        # 如果目標值比中間值大，則在右半邊搜尋
        else:
            return binary_search_recursive(arr, mid + 1, right, x)
    else:
        # 找不到目標值的情況
        return -1
```

2.非遞迴版本的二元搜尋

```
def binary_search_iterative(arr, x):
    left, right = 0, len(arr) - 1

    while left <= right:
        mid = left + (right - left) // 2

        # 如果目標值正好在中間
        if arr[mid] == x:
            return mid
        # 如果目標值比中間值小，則在左半邊搜尋
        elif arr[mid] > x:
            right = mid - 1
        # 如果目標值比中間值大，則在右半邊搜尋
        else:
            left = mid + 1
    # 找不到目標值的情況
    return -1
```

㈡推導二元搜尋的時間複雜度（Time Complexity）如下所示：

假設有一個長度為 n 的已排序陣列。

1.遞迴版本分析：

每次遞迴時，將問題規模削減為原來的一半（因為每次將範圍劃分為左右兩部分）。

如果我們稱每次分割的操作時間為 O(1)，則遞迴的深度最多為 $\log_2(n)$ 層。

因此，遞迴版本的時間複雜度為 O(logn)。

2.非遞迴版本分析：非遞迴版本同樣是每次將範圍劃分為左右兩部分，但使用迴圈進行控制，迴圈會進行 $\log_2(n)$ 層迭代。每次迭代中，計算中間元素並更新左右邊界，操作時間也是 O(1)。因此，非遞迴版本的時間複雜度同樣為 O(logn)。

四、堆疊（Stack）與佇列（Queue）是常見的資料結構，請回答下列問題：

㈠利用雙向佇列（Deque）循序輸入 1, 2, 3, 4, 5, 6, 7，請問能否得到 5174236 的輸出排列？並說明其過程或理由。（10 分）

㈡若有 1, 2, 3, 4 四個數字要依序 Push 進堆疊，再於任意時間點 Pop 出堆疊，請列出可能的輸出組合。（15 分）

1.考題難易：★★★☆☆

2.解題關鍵：本題為堆疊與佇列應用題，需要熟悉 deque 排序與堆疊排序操作始可作答。

【擬答】：

㈠利用雙向佇列（Deque）進行循序輸入 1, 2, 3, 4, 5, 6, 7，無法直接得到輸出排列 5174236。因為如果可以從 1, 2, 3, 4, 5, 6, 7 的數字串列中找到循序子串列就可以得到此輸出。但因為雙向佇列的特性決定了元素的出入順序。雙向佇列（Deque）具有兩端操作特性，支持從兩端進行插入和刪除操作，即可以從前端（front）插入和刪除元素，也可以從後端（rear）插入和刪除元素。因此上列問題轉換成用兩個子串列表示即可得到此輸出，但是上列輸出無法以此表示。因為若要得到 51 輸出，要先循序輸入 1, 2, 3, 4, 5 再輸出，接下來為了要輸出 7，要將 6, 7 輸入，但接下來的 4 與 2 都被夾在佇列中間而無法輸出。因此，無法通過正常操作順序得到題目要求的排列 5174236。

㈡當將數字 1, 2, 3, 4 依序 Push 進堆疊後，再進行 Pop 操作時，可能的輸出組合取決於 Pop 時該元素是否再堆疊頂端，因此有些組合是不可能出現的。這裡列出所有可能的輸出組合：

1.先 pop 1 的組合有如下 5 種，1, 4, 3, 2 不可能出現，因為在 pop 4 時，元素 2 不可能在堆疊頂端

1, 2, 3, 4
1, 2, 4, 3
1, 3, 2, 4
1, 3, 4, 2
1, 4, 2, 3

2.先 pop 2 的組合有如下 5 種，2, 4, 1, 3 不可能出現，因為在 pop 4 時，元素 1 不可能在堆疊頂端

2, 1, 3, 4
2, 1, 4, 3
2, 3, 1, 4
2, 3, 4, 1、
2, 4, 3, 1

3.先 pop 3 的組合有如下 3 種，3, 1, 2, 4、3, 1, 4, 2、3, 4, 1, 2 這三個組合不可能出現

3, 2, 1, 4
3, 2, 4, 1
3, 4, 2, 1

4.先 pop 4 的組合僅有 1 種 4, 3, 2, 1 組合，其餘的 4, 1, 2, 3、4, 1, 3, 2、4, 2, 1, 3、4, 2, 3, 1、4, 3, 1, 2 等 5 種組合不會出現，原因都是先 pop 4 時，堆疊內為 3, 2, 1，無法出現其他組合。

以上共計產出 14 種輸出組合。