第4次作業-作業-HW4

> 學號：112111215
> 姓名：莊博勳
> 作業撰寫時間：180 (mins，包含程式撰寫時間)
> 最後撰寫文件日期：2024/12/29

本份文件包含以下主題：(至少需下面兩項，若是有多者可以自行新增)

- ☑ 說明內容
- ☑ 個人認為完成作業須具備觀念

1. 請回答下面問題。

Ans:

```
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

class BinarySearchTree:
    def __init__(self):
        self.root = None

    def insert(self, value):
        if self.root is None:
            self.root = TreeNode(value)
        else:
            self._insert_helper(self.root, value)

    def _insert_helper(self, node, value):
        # 比較 value 和當前節點的值，決定插入到哪個子樹
        if value < node.value:
            if node.left is None:
                node.left = TreeNode(value)
            else:
                self._insert_helper(node.left, value)
        elif value > node.value:
            if node.right is None:
                node.right = TreeNode(value)
            else:
                self._insert_helper(node.right, value)

    def inorder_traversal(self):
        return self._inorder_traversal_helper(self.root)

    def _inorder_traversal_helper(self, node):
        if node is None:
```

```
            return []
        return self._inorder_traversal_helper(node.left) + [node.value] +
self._inorder_traversal_helper(node.right)

# 使用陣列來建立二元搜尋樹
def build_bst_from_array(array):
    bst = BinarySearchTree()
    for value in array:
        bst.insert(value)
    return bst

# 測試代碼
array = [20, 8, 22, 4, 12, 10, 14]
bst = build_bst_from_array(array)

# 輸出中序遍歷結果，應該是升冪順序
print("In-order Traversal:", bst.inorder_traversal())
```

2. 請回答下面問題。

Ans:

```
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

class BinarySearchTree:
    def __init__(self):
        self.root = None

    def insert(self, value):
        # 如果樹是空的，建立根節點
        if self.root is None:
            self.root = TreeNode(value)
        else:
            self._insert_helper(self.root, value)

    def _insert_helper(self, node, value):
        """
        遞迴方式進行插入：
        - 如果 value 小於 node.value，則插入到左子樹
        - 如果 value 大於 node.value，則插入到右子樹
        """
        if value < node.value:
            if node.left is None:
                node.left = TreeNode(value)
            else:
                self._insert_helper(node.left, value)
        elif value > node.value:
```

```
            if node.right is None:
                node.right = TreeNode(value)
            else:
                self._insert_helper(node.right, value)

    def inorder_traversal(self):
        return self._inorder_traversal_helper(self.root)

    def _inorder_traversal_helper(self, node):
        """
        中序遍歷，返回一個包含樹中元素的升冪列表
        """
        if node is None:
            return []
        return self._inorder_traversal_helper(node.left) + [node.value] +
self._inorder_traversal_helper(node.right)

# 使用陣列來建立二元搜尋樹
def build_bst_from_array(array):
    bst = BinarySearchTree()
    for value in array:
        bst.insert(value)
    return bst

# 測試代碼
array = [20, 8, 22, 4, 12, 10, 14]
bst = build_bst_from_array(array)

# 輸出中序遍歷結果，應該是升冪順序
print("In-order Traversal:", bst.inorder_traversal())
```

3. 請回答下面問題：

Ans:

1. 最壞情況（Worst-case）：
假設你插入的數字是按順序排列的（例如：1, 2, 3, 4, 5），這樣樹就會變成一條 鏈表。
插入每個新節點時，必須沿著整條鏈表（即整棵樹）尋找插入位置。
在這種情況下，每次插入需要走過 n 個節點。
所以，插入 n 個節點的總時間複雜度是 O(n²)（這是一個比較差的情況）。

2. 最好的情況（Best-case）：
假設插入的數字是隨機的，樹能保持平衡，這樣每次插入時，都能選擇左邊或右邊的子樹，樹的高度大約是 log n。
每次插入只需要走 log n 個節點。
所以，插入 n 個節點的總時間複雜度是 O(n log n)（這是最佳情況）。

3. 平均情況（Average-case）：
在一般情況下，數字的插入順序是隨機的，樹的高度會在 O(log n) 和 O(n) 之間波動。
平均來說，樹的高度大約是 log n，所以每次插入的時間複雜度是 O(log n)。
插入 n 個節點的總時間複雜度大約是 O(n log n)。

總結：
最壞情況：O(n²)（當樹變成鏈表）
最佳情況：O(n log n)（樹保持平衡）
平均情況：O(n log n)（大多數情況）
如果樹能夠保持平衡，則大部分情況下的建樹時間複雜度是 O(n log n)。

4. 請回答下面問題：

Ans:

何時使用樹狀結構？
當你需要處理 有層級的資料 時，像是：

檔案夾（資料夾裡有檔案、子資料夾）
公司結構（老闆 -> 經理 -> 員工）
目錄分類（網站或商品的分類）
如何操作樹中的節點？
新增子節點：

就是把新的資料加到樹的某個位置。
例子：在資料夾裡新增一個檔案或資料夾。
修改節點內容：

找到要改的資料，然後修改它。
例子：修改檔案名稱或資料夾名稱。
刪除節點：

把某個資料刪掉。
例子：刪除檔案或資料夾。
沒有子資料夾直接刪掉。
有子資料夾的話，改變結構或刪除整個資料夾。
總結：
新增：加一個新的資料。
修改：改一個資料。
刪除：刪掉一個資料。
樹狀結構主要用來儲存和操作這些有層次關係的資料。

個人認為完成作業須具備觀念

開始寫說明，需要說明本次練習需學會那些觀念 (需寫成文章，需最少50字，並且文內不得有你、我、他三種文字)且必須提供完整與練習相關過程的notion筆記連結