

第4次作業-作業-HW4

學號：112109133

姓名：張廷瑋

作業撰寫時間：180 (mins，包含程式撰寫時間)

最後撰寫文件日期：2023/01/06

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

說明程式與內容

開始寫說明，該說明需說明想法，並於之後再對上述想法的每一部分將程式進一步進行展現，若需引用程式區則使用下面方法，若為.cs檔內程式除了於敘述中需註明檔案名稱外，還需使用語法```語言種類 程式碼```，其中語言種類若是要用python則使用py，java則使用java，C/C++則使用cpp，下段程式碼為語言種類選擇csharp使用後結果：

```
public void mt_getResult(){  
    ...  
}
```

若要於內文中標示部分網頁檔，則使用以下標籤```html 程式碼```，下段程式碼則為使用後結果：

```
<%@ Page Language="C#" AutoEventWireup="true" ...>  
  
<!DOCTYPE html>  
  
<html xmlns="http://www.w3.org/1999/xhtml">  
<head runat="server">  
<meta http-equiv="Content-Type" ...>  
    <title></title>  
</head>  
<body>  
    <form id="form1" runat="server">  
        <div>  
            </div>  
    </form>  
</body>  
</html>
```

更多markdown方法可參閱<https://ithelp.ithome.com.tw/articles/10203758>

請在撰寫"說明程式與內容"該塊內容，請把原該塊內上述敘述刪除，該塊上述內容只是用來指引該怎麼撰寫內容。

1. 請回答下面問題。

Ans: ''' tree = ['A', 'B', 'C', 'D', 'E', 'F', None]

print(f"Tree : {tree}") '''

2. 請回答下面問題。

Ans: ''' class CompactBinaryTree: def **init**(self, elements): """初始化二元樹，接收一個陣列作為基礎樹結構。""" self.tree = elements

```
def find_node_index(self, value):
    """找到節點索引，若不存在返回 None"""
    return self.tree.index(value) if value in self.tree else None

def add_child(self, parent_value, child_value, position='left'):
    """新增子節點到父節點的左或右位置"""
    parent_index = self.find_node_index(parent_value)
```

'''

3. 請回答下面問題：

Ans: ''' if parent_index is None: print(f"父節點 {parent_value} 不存在。") return False

```
# 計算子節點索引 (左子節點: 2i+2, 右子節點: 2i+1)
child_index = 2 * parent_index + (2 if position == 'left' else 1)

# 確保陣列長度足夠
if len(self.tree) <= child_index:
    self.tree += [None] * (child_index - len(self.tree) + 1)

# 確保該位置為空後新增子節點
if self.tree[child_index] is None:
    self.tree[child_index] = child_value
    print(f"成功在 {parent_value} 的 {position} 新增子節點 {child_value}。")
else:
    print(f"{parent_value} 的 {position} 已有子節點 {self.tree[child_index]}。")

def display(self):
    """顯示樹的結構 (過濾掉 None)"""
    print([node for node in self.tree if node is not None])
```

class TreeNode: def **init**(self, value, index): self.value = value self.index = index self.left = None self.right = None

class BinaryTree: def **init**(self, values): self.root = self.build_tree(values) ''' 4. 請回答下面問題：

Ans: ''' def build_tree(self, values): if not values: return None nodes = [TreeNode(value, i) if value is not None else None for i, value in enumerate(values)] for i in range(len(nodes)): if nodes[i] is not None: left_index = 2 * i + 1 right_index = 2 * i + 2 if left_index < len(nodes): nodes[i].left = nodes[left_index] if right_index < len(nodes): nodes[i].right = nodes[right_index] return nodes[0]

```
def find(self, value):
    return self._find_recursively(self.root, value)

def _find_recursively(self, current, value):
    if current is None:
        return None
    if current.value == value:
        return current
    left_result = self._find_recursively(current.left, value)
    if left_result:
        return left_result
    return self._find_recursively(current.right, value)

def add_child(self, parent_value, child_value, is_left=True):
    parent_node = self.find(parent_value)
    if parent_node:
        if is_left:
            if parent_node.left is None:
                parent_node.left = TreeNode(child_value, -1) # -1 表示索引未知
                print(f"新增成功：節點 {child_value} 作為 {parent_value} 的左子節點。")
            else:
                print("錯誤：左子節點已存在。")
        else:
            if parent_node.right is None:
                parent_node.right = TreeNode(child_value, -1) # -1 表示索引未知
                print(f"新增成功：節點 {child_value} 作為 {parent_value} 的右子節點。")
            else:
                print("錯誤：右子節點已存在。")
    else:
        print(f"錯誤：找不到節點 {parent_value}。")
```

```
def main(): print("選擇測試模式：") print("1. 純陣列寫法") print("2. 鏈結串列 (Linked List) 寫法") mode = input("請輸入模式編號 ( 1 或 2 )：")
```

```
if mode == "1":
    initial_tree = ['A', 'B', 'C', 'D', 'E', 'F']
    tree = CompactBinaryTree(initial_tree)
    print("初始二元樹已建立 ( 純陣列寫法 )：")

    while True:
        print("\n二元樹操作選單：")
```

```
print(f"二元樹：{[node for node in tree.tree if node is not None]}")
print("1. 搜尋節點")
print("2. 新增子節點")
print("3. 離開")
choice = input("請輸入選項 ( 1-3 ) : ")

if choice == "1":
    value = input("請輸入要搜尋的節點值 : ")
    index = tree.find_node_index(value)
    if index is not None:
        print(f"找到節點：值={value}，索引={index+1}")
    else:
        print("節點不存在。")
elif choice == "2":
    parent = input("請輸入父節點的值 : ")
    child = input("請輸入要新增的子節點值 : ")
    position = input("請輸入新增子節點的位置 ('left' 或 'right') : ").lower()
    if position not in ['left', 'right']:
        print("輸入的位置無效，請重新輸入 'left' 或 'right'。")
        continue
    tree.add_child(parent, child, position=position)
elif choice == "3":
    print("感謝使用，程式結束！")
    break
else:
    print("無效選項，請重新輸入。")

elif mode == "2":
    initial_tree = ['A', 'B', 'C', 'D', 'E', 'F']
    tree = BinaryTree(initial_tree)
    print("初始二元樹已建立 ( 鏈結串列寫法 ) : ")

    while True:
        print("\n二元樹操作選單 : ")
        print(f"二元樹：{initial_tree}")
        print("1. 搜尋節點")
        print("2. 新增子節點")
        print("3. 離開")
        choice = input("請輸入選項 ( 1-3 ) : ")

        if choice == "1":
            value = input("請輸入要搜尋的節點值 : ")
            node = tree.find(value)
            if node:
                print(f"找到節點：值={node.value}，索引={node.index+1}")
            else:
                print("節點不存在。")
        elif choice == "2":
            parent_value = input("請輸入父節點的值 : ")
            child_value = input("請輸入要新增的子節點值 : ")
            position = input("請輸入子節點位置 ( 左子節點輸入 'L'，右子節點輸入
```

```
'R' ) : ").upper()
    is_left = position == "L"
    tree.add_child(parent_value, child_value, is_left)
elif choice == "3":
    print("感謝使用，程式結束！")
    break
else:
    print("無效選項，請重新輸入。")

else:
    print("無效模式，程式結束。")
```

'''

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