

第4次作業-作業-HW4

學號：112111204

姓名：邱元平

作業撰寫時間：180 (mins · 包含程式撰寫時間)

最後撰寫文件日期：2023/01/07

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:

```
class BSTA: def init(self, size): self.tree = [None] * size #初始化 self.size = size
```

```
def insert(self, value):
    if self.tree[0] is None: #根節點 = 空(none)
        self.tree[0] = value
        return
    index = 0
    while index < self.size:
        if value < self.tree[index]: #左
            next_index = 2 * index + 1
        else: #右
            next_index = 2 * index + 2

        if next_index >= self.size:
            print("樹值已滿，無法加入")
            return
        if self.tree[next_index] is None:
            self.tree[next_index] = value
            return
        index = next_index

def __str__(self):
    return str(self.tree)
```

```
print("\n") #讓輸出結果在VSC上有分行以便辨識 bst = BSTA(4) bst.insert(10) bst.insert(5) bst.insert(15)
bst.insert(3) bst.insert(7)
```

```
print(bst)
```

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 BSTL: def init(self): self.root = None
```

```
def insert(self, value):
    if self.root is None: #根節點 = 空(none)
        self.root = TreeNode(value)
        return
    self._insert_recursive(self.root, value)

def _insert_recursive(self, node, value):
    if value < node.value: #左
```

```

        if node.left is None:
            node.left = TreeNode(value)
        else:
            self._insert_recursive(node.left, value)
    else: #右
        if node.right is None:
            node.right = TreeNode(value)
        else:
            self._insert_recursive(node.right, value)

def inorder_traversal(self, node, result=None):
    if result is None:
        result = []
    if node:
        self.inorder_traversal(node.left, result)
        result.append(node.value)
        self.inorder_traversal(node.right, result)
    return result

```

```

print() bst = BSTL() bst.insert(10) bst.insert(5) bst.insert(15) bst.insert(3) bst.insert(7)
print(bst.inorder_traversal(bst.root))

```

3. 請回答下面問題：

Ans:

數組方式(1.py): 插入操作需要從根節點開始，透過計算索引定位位置，時間複雜度為 $O(h)$ ，其中 $h=\log n$ 為樹的高度。

鏈結表方式(2.py): 插入操作同樣需要從根節點遞歸查找，時間複雜度為 $O(h)$ ，但鏈結表支援動態存儲，節省空間。

4. 請回答下面問題：

Ans:

樹狀結構常用於快速尋找和排序，例如檔案系統、資料庫索引、以及表示層級資料(如組織結構)。

操作範例：新增(Insert): 數組：根據值計算插入位置(如程式1.py)。鏈結表：遞歸比較並插入到正確的子節點(如程式2.py)。

修改(Modify)：找到目標節點後直接修改其值。鏈結表更靈活，數組需要根據索引定位，且可能需要擴展存儲。

刪除(Delete)：若節點為葉節點，直接刪除；若節點有一個子樹，替換為該子樹；若節點有兩個子樹，找到中序遍歷的後繼節點替代目標節點的值，然後刪除後繼節點。

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