資料結構 PA1

b09901080 電機三 吳宣逸

- My Solutions
  - ◆ **資料結構**

I. TreeNode

由以下三個 pointer 以及一個 key 組成：

- ✓ `TreeNode *parent = nullptr`
- ✓ `TreeNode *leftChild = nullptr`
- ✓ `TreeNode *rightChild = nullptr`
- ✓ `int key`

II. BinarySearchTree

由 root 這個 pointer 指到其他 TreeNode，並且 parent 和 child 互相 point，以此建立整棵 tree。

◆ **演算法**

I. 用 postorder 建 BinarySearchTree

1. 首先將 postorder 的所有數字做成 TreeNode，根據原先順序由後往前 enqueue 進 postTree 這個收集 TreeNode 的 queue。
2. 由右方可看到 post order traversal 會到最後才 visit 到 x，因此只要依照 postTree 的順序自 root 將所有 treeNode insert 就可以建立對應的 BinarySearchTree 了。

```
PostOrderTraversal (x)
    PostOrderTraversal (x.leftChild)
    PostOrderTraversal (x.rightChild)
    visit x
```

II. PreOrderTraversal

演算法如下，visit 的行為是將 x 的 key push 進用來儲存 preorder 結果的 vector，由 x = root 開始進行 Traversal 可以得到正確的 preorder：

```
PreOrderTraversal (x)
    if x != nullptr
        visit x
        PreOrderTraversal (x.leftChild)
        PreOrderTraversal (x.rightChild)
```

III. treeHeight

求 treeHeight 的方法是借助 Traversal，在每次回傳 x 對應的 tree height，並且

1. 當 x = nullptr，回傳 0。
2. 若 x ≠ nullptr，則 tree height = (left subtree 和 right subtree 回傳值最大者) + 1。

由於在前面已經要求做 preorder Traversal，所以求 treeHeight 會跟著 preorder Traversal 進行。

IV. 找每層(同樣 depth)key 最大的 node

雖然作業說明提到 n 是 height of tree，但與 height 無關，必須用 depth 做才行，演算法如以下 pseudo code。想法是先設長度 n 的陣列 Max，Max[i]儲存 depth i 的最大值，在每次都記錄 depth，

而 children 的 depth 則是 depth+1，如此便克服各 treeNode 的 height 不好找的問題了。

```
updateMaxOfEachDepth(current, depth, Max)
    Max[depth] = max(Max[depth], current.key)

    if current.leftChild != nullptr
        updateMaxOfEachDepth(current.leftChild, depth + 1, Max)
    if current.rightChild != nullptr
        updateMaxOfEachDepth(current.rightChild, depth + 1, Max)
```

- Reference

1. treeNode class 型式：
   http://alrightchiu.github.io/SecondRound/binary-tree-jian-li-yi-ke-binary-tree.html
2. C++ queue 用法：
   https://shengyu7697.github.io/std-queue/
3. Traversals 與 tree 相關知識：
   陳和麟教授資料結構課堂筆記、孫紹華教授演算法 ppt

- Collaborators

No Collaborator.