tags: 資料結構

HW9

題目要求使用非遞迴方法建立二元樹以及前後序遍歷。

Build

在這邊我們使用 stack 代替遞迴維護狀態，使用 stack 維護當前需要操作的節點，題目給測前序測資，因此我們從頭開始到尾處理，並且期望所有點從左開始往右插入，所以先從左子樹插入，如果左子樹找不到位置可以插入的話就換成從右子樹，總共會有下列三種狀態。

- 狀況一
  如果當前節點左右子樹都不為空，則取出棧頂並且退棧，如果還是不為空的話則繼續退棧直到上述兩個條件其中之一成立。
- 狀況二
  如果當前左子樹為空，當前節點插入左子樹，並且將當前節點入棧。
- 狀況三
  如果當前左子樹不為空，則將當前節點插入右子樹，並將當前節點入棧。

需要注意的地方是如果值為 -1 的話則不能入棧，因為在實作面上我們將 -1 當成 NULL，pseudo code 如下。

```
void build() {
    Node* cur;
    stack<Node*> stk;

    root = init(arr[1]), stk.push(root);
    for (int i=2; i<=n; i++) {
        Node* tmp = init(init[i]), cur=stk.top();
        while (cur->left && cur->right)
            cur = stk.top(), stk.pop();
        if (!cur->left)
            cur->left = init(arr[i]), cur = cur->left;
        else
            cur->right = init(arr[i]), cur = cur->right;
        if (arr[i]!=-1) stk.push(cur);
    }
}
```

Pre-order

前序的走訪順序為中左右，我們首先將根節點入棧。

- 取出棧頂作為當前節點，並且印出其值。
- 判斷其右子樹是否為 -1，否的話則入棧。
- 判斷其左子樹是否為 -1，否的話則入棧

接著按上面步驟重複操作直到棧為空，需要注意的點是先將右子樹入棧的原因是棧的特性是 First in Last out，這樣在出棧時才會維持前序的走訪順序，以下為 pseudo code。。

```
void pre() {
    Node* cur;
    stack<Node*> stk; stk.push(root);
    while (!stk.empty()) {
        cur = stk.top(), stk.pop();
        cout << cur->val << ' ';
        if (cur->right) stk.push(cur->right);
        if (cur->left) stk.push(cur->left);
    }
}
```

Post-order

後序走訪順序為左右中，我們首先將根節點入棧，跟前序不太一樣的地方是我們需要維護兩個棧，一個是當前節點，一個是輸出順序。

- 取出當前節點的棧頂作為當前節點，並將其入棧輸出順序。
- 判斷其左子樹是否為 -1，否的話則入棧。
- 判斷其右子樹是否為 -1，否的話則入棧。

接著按上面步驟重複操作直到當前節點棧為空，接著將輸出順序棧依次取出棧頂並輸出，這邊先入左子樹再入右子數的原因是第一次順序會是正的，再次入棧後就會反過來了，以下為 pseudo code。

```
void post() {
    Node* cur;
    stack<int> out;
    stack<Node*> stk; stk.push(root);
    while (!stk.empty()) {
        cur = stk.top(), stk.pop();
        out.push(cur->val);

        if (cur->left) stk.push(cur->left);
        if (cur->right) stk.push(cur->right);
    }
    while (!out.empty()) {
         cout << out.top() << ' ';
         out.pop();
    }
}
```