tags: 資料結構

HW11

本題需要講述線索二叉樹的實作，線索二叉樹與普通二叉樹基本上沒有區別，唯一差別在於普通二叉樹會有 $n+1$ 個節點空間是浪費的 (葉子節點)，線索二叉樹可以運用這些空節點來儲存任意序列遍歷的前驅及後繼。

Insert

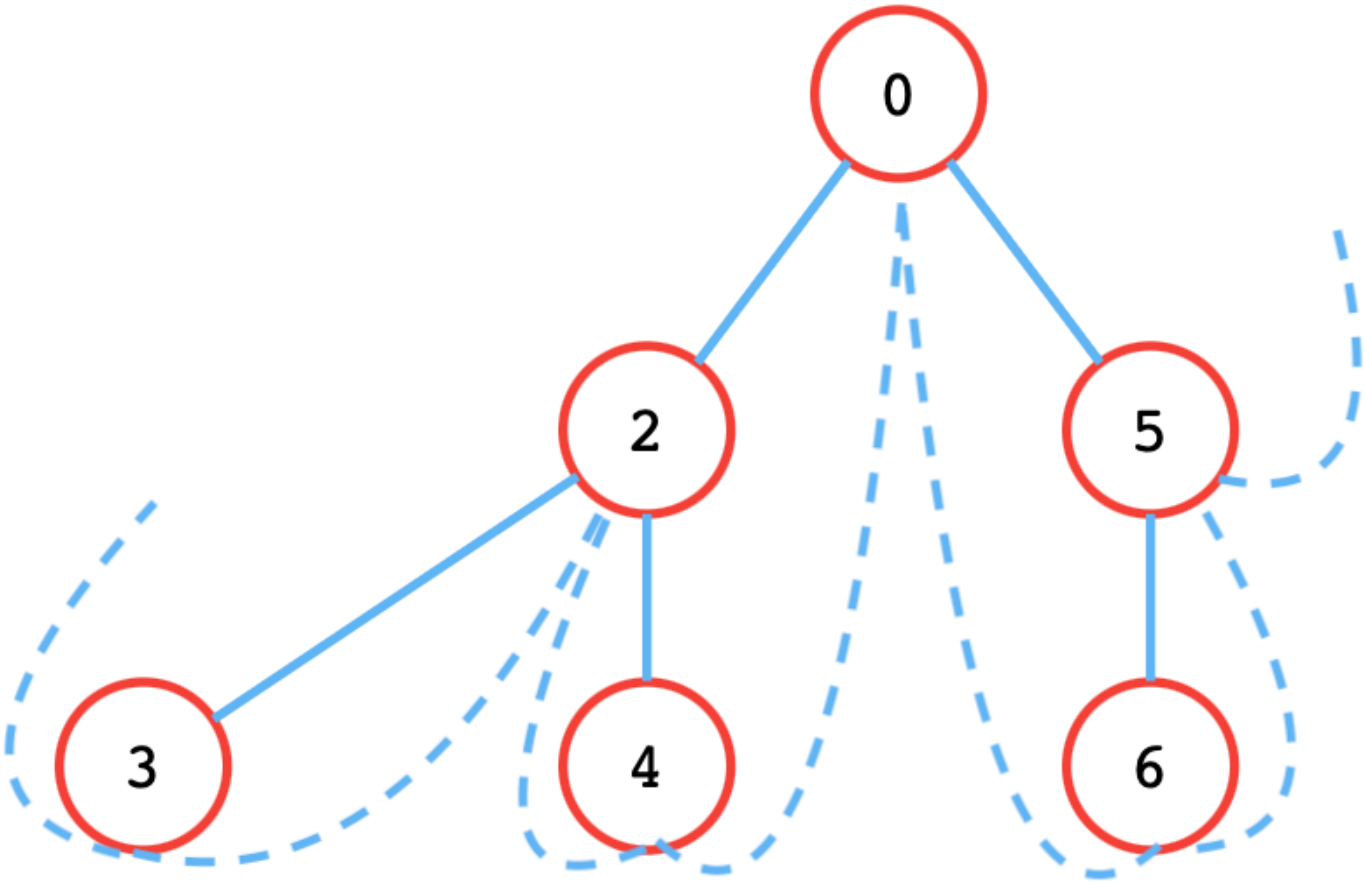
插入部分因為二叉樹並沒有一定規律，所以我們想在特定節點的左右插入新節點的話，在最壞情況下需要花費 $O(N)$ 尋找到節點才能做插入，以下為 pseudo code。

```
int insert(Node* root, int x, int y, int flag) // 0 ->left 1->right
{
    if (root->val == x) {
        if (flag)
            root->right = init(y);
        else
            root->left = init(y);
        return 1;
    } else {
        if (root->left && insert(root->left, x, y, flag))
            return 1;
        if (root->right && insert(root->right, x, y, flag))
            return 1;
    }
    return 0;
}
```

Thread

對於一棵二叉樹的線索化就是將其空指標賦值前驅、後繼，首先在建樹時需要一個標記用來記錄這個節點的左右子樹是否為空，如果為空則打上標記，且在全域空間設立 `pre` 變數用來在線索化時儲存前驅。

中序線索化二叉樹，若節點的左標記為 0， 則左子樹指向前驅；否則，該節點的前驅是以該節點為根的左子樹上按中序遍歷的最後一個節點，若右標記為 0，則右標記指向後繼；否則，該節點的後繼是以該節點為根的右子樹上按中序遍歷的第一個節點，下圖為線索化後的二叉樹。



以下為 pseudo code。

```
void thread(Node* root)
{
    if (root) {
        thread(root->left);
        if (!root->left)
            root->ltag = 1, root->left = pre;
        if (pre && !pre->right)
            pre->rtag = 1, pre->right = root;
        pre = root, thread(root->right);
    }
}
```

Traverse

因為線索化後各個節點的指向有改變，因此不能使用原來的遍歷方式，這時候需要使用新的方法來遍歷樹，這邊可以不用像一般二叉樹使用遞迴的方式，可以直接改用線性遞歸的方法來遍歷，這樣也能提高效率，以下為 pseudo code。

```
void inorder(Node* root)
{
    Node* cur = root;
    while (cur) {
        while (!cur->ltag)
            cur = cur->left;
        printf("%d ", cur->val);
        while (cur && cur->rtag)
            cur = cur->right, printf("%d ", cur->val);
        cur = cur->right;
    }
    puts("");
}
```