作業三禮拜四
5pm due

更多的樹

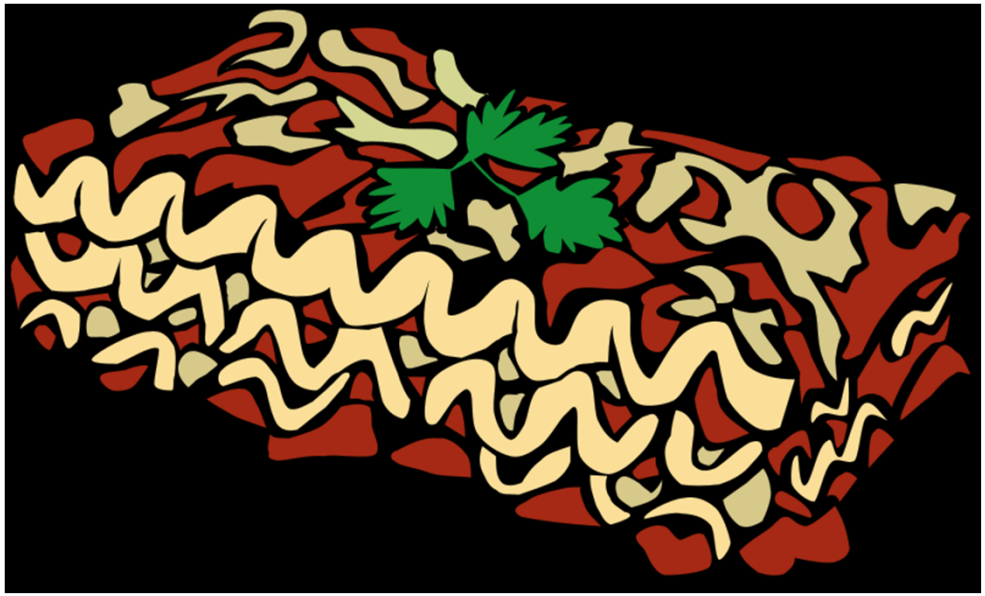
Michael Tsai

2010/10/29

請各位多利用下周二三四的
office hour

今日菜單

- 樹裡面找東西
- 選擇樹 (勝者樹與敗者樹)
- 樹的相關動作(複製樹...)
- 有線頭的樹
- 森林
- 解邏輯運算式



Binary search tree

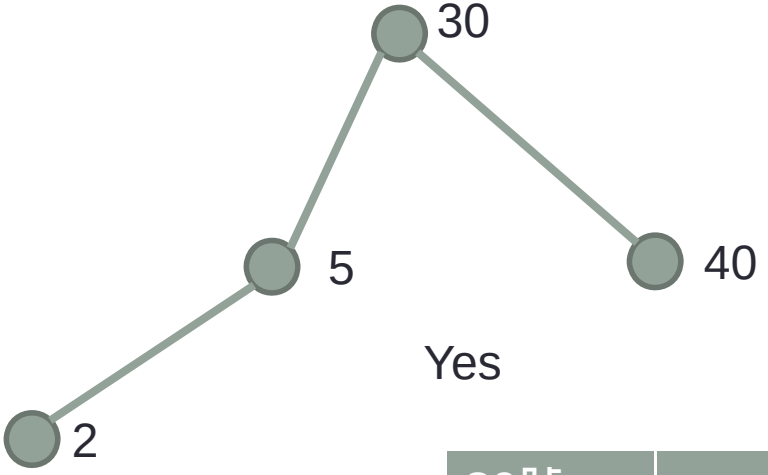
- 問題: 找系上某個同學的作業成績
- 條件:
- 知道學號 (key)
- 找到學號就可以對應到儲存資料的地方(search)
- 常常有人進來(add node)
- 常常有人出去(delete node)

Binary search tree

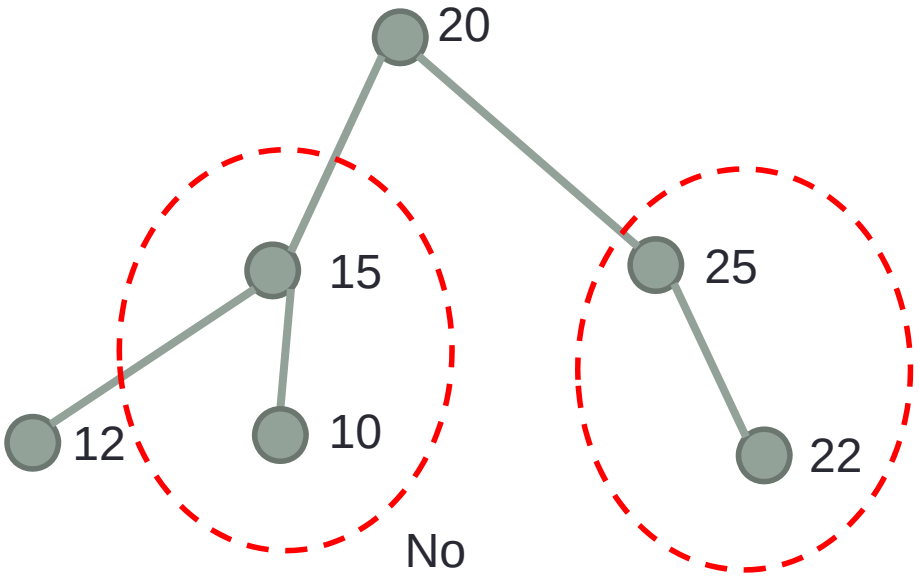
- Definition: A binary search tree is a binary tree. It may be empty. If it is not empty then it satisfies the following properties:
- 1. The root has a key.
- 2. The keys (if any) in the **left** subtree are **smaller** than the key in the root
- 3. The keys (if any) in the **right** subtree are **larger** than the key in the root
- 4. The left and right subtrees are **also binary search trees**
- (隱藏) All keys are distinct.

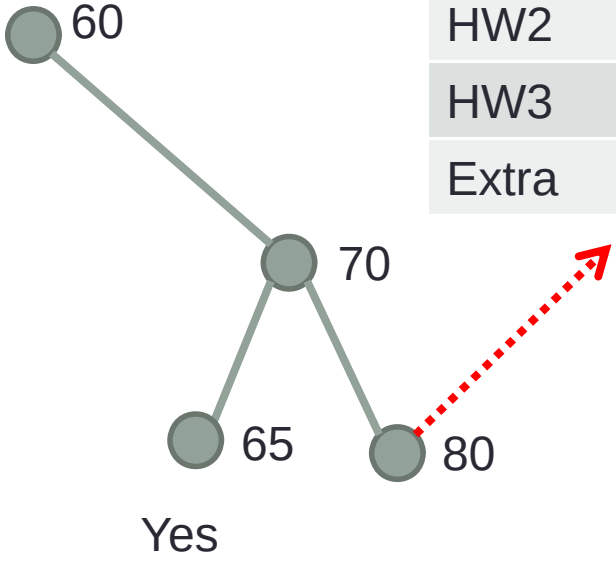
Binary search tree

- 這些是不是binary search tree?
- 找到以後要做什麼?



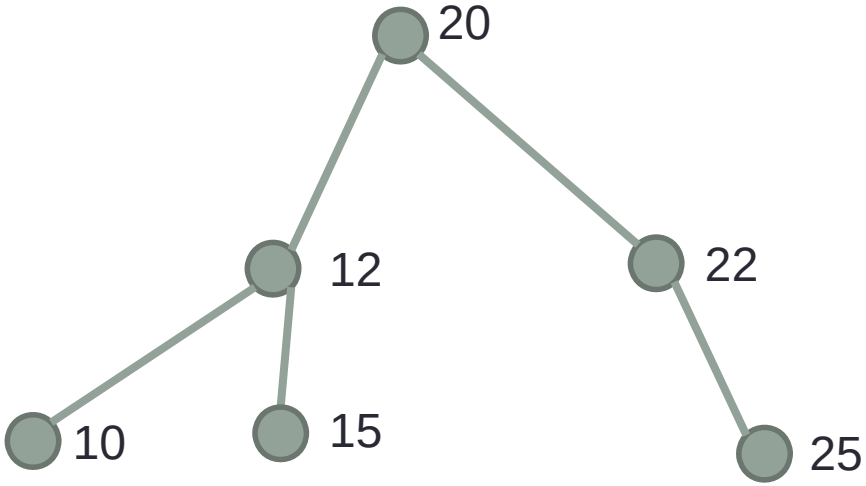
80號	
HW1	65
HW2	65
HW3	空
Extra	-20000





如何尋找?

- 假設要找10

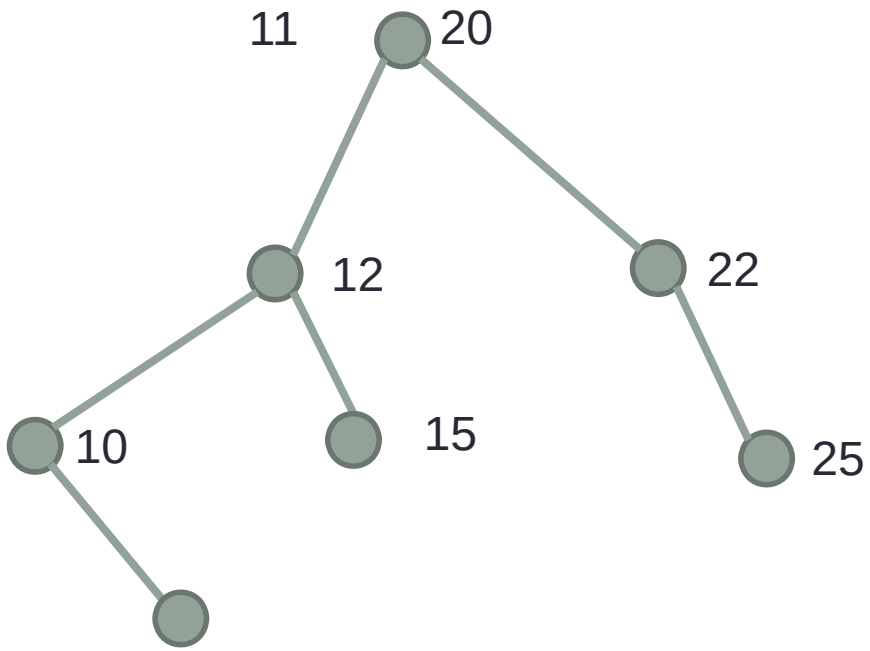



```
element *search(tPointer root, int key) {
    if (!root) return NULL;
    if (k==root->data.key) return &(root->data);
    if (k<root->data.key) return search(root->leftChild, k);
    return search(root->rightChild, k);
}
```

- 簡單. 那time complexity = O(??)

如何插入新的node?

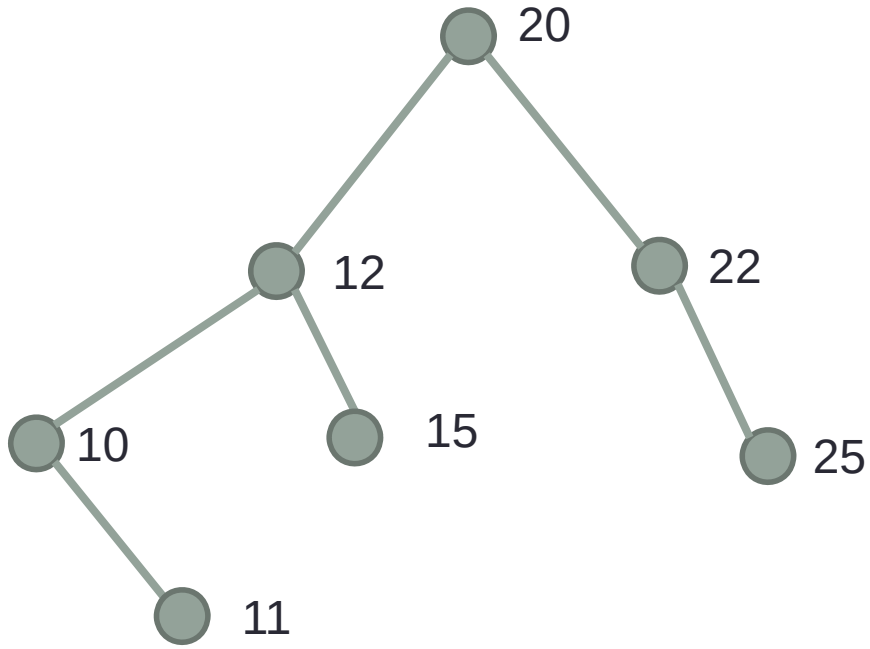
- 先找有沒有一樣的 (隱藏版rule: binary search tree裡面不可以有一樣的key)
- 找不到的話, 插在最後”找不到”的位置
- 插入: 11



如何刪掉一個node?

- 首先要先找到那個node
- 接著, 有各種不同情形:

- 如果沒有手(degree=0)
- 直接拿掉

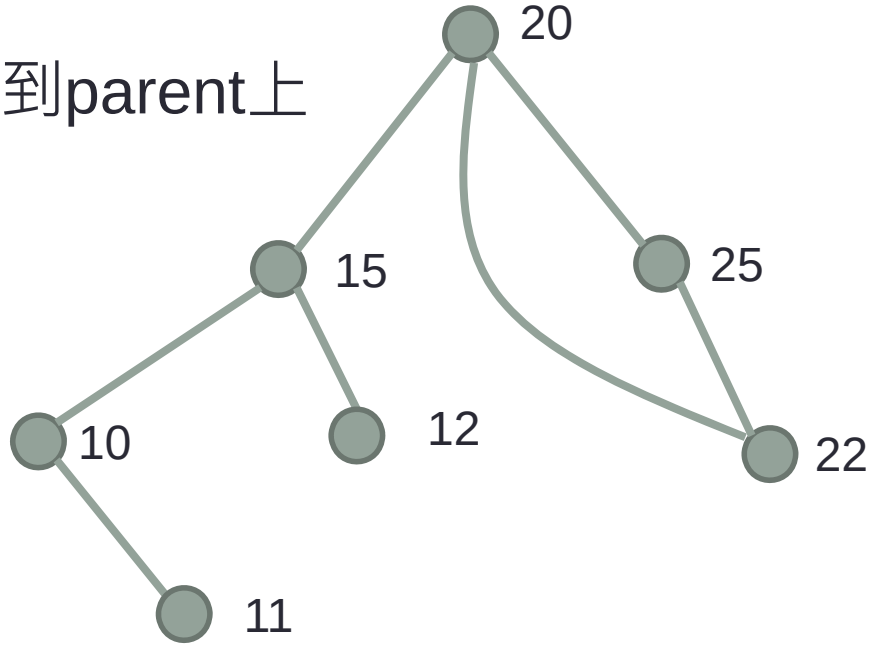


如何刪掉一個node?

- 如果只有一隻手 (degree=1)
- 則把那個唯一的child拿上來接到parent上

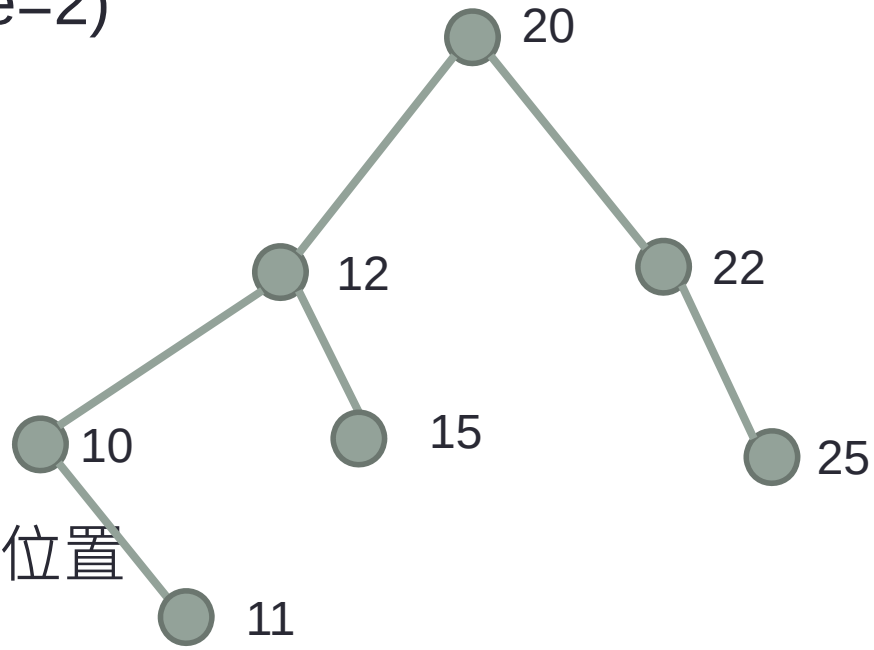
- 例如: 拿掉25

- 問題: 要怎麼接?
- 怎麼記得parent是誰?



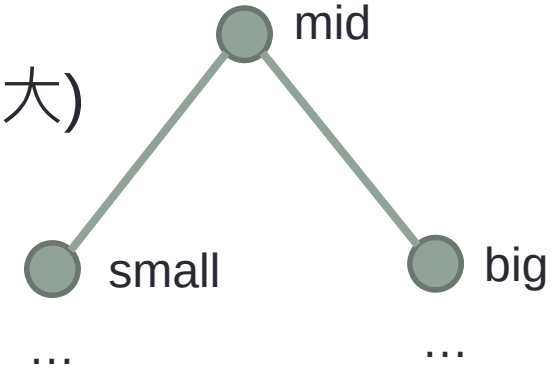
如何刪掉一個node?

- 如果兩手都有東西呢? (degree=2)
- 例如刪掉12
- 找左邊child底下最大的
- (或者是右邊child底下最小的)
- 刪掉它並把它移到原本刪掉的位置
- 問題: 那如果那個最大的底下還有child呢?



一些binary search tree的動作

- threeWayJoin(small, mid, big)
- (small裡面都比mid小, big裡面都比mid大)
- mid->left=small;
- mid->right=big;
- 結束!



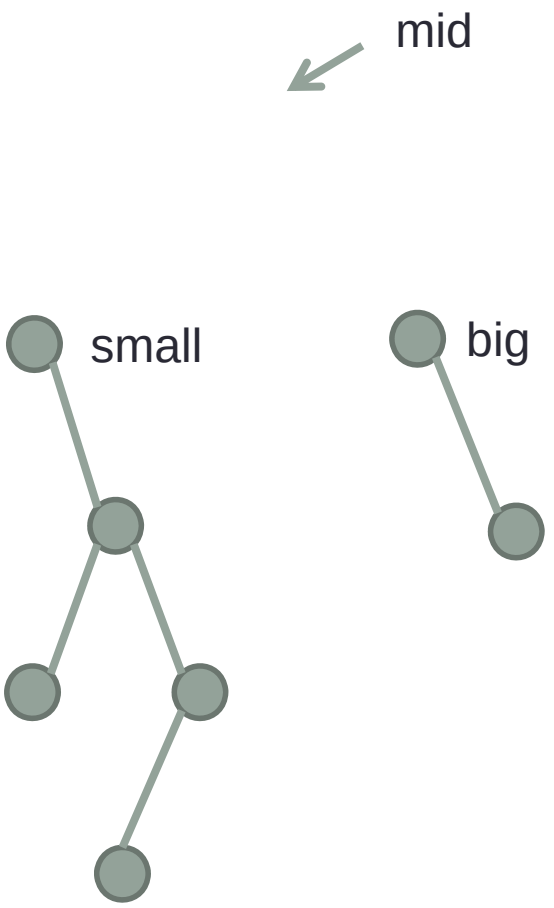
- height=?

一些binary search tree的動作

- twoWayJoin(small, big)
- (small裡面的都比big裡面的小)

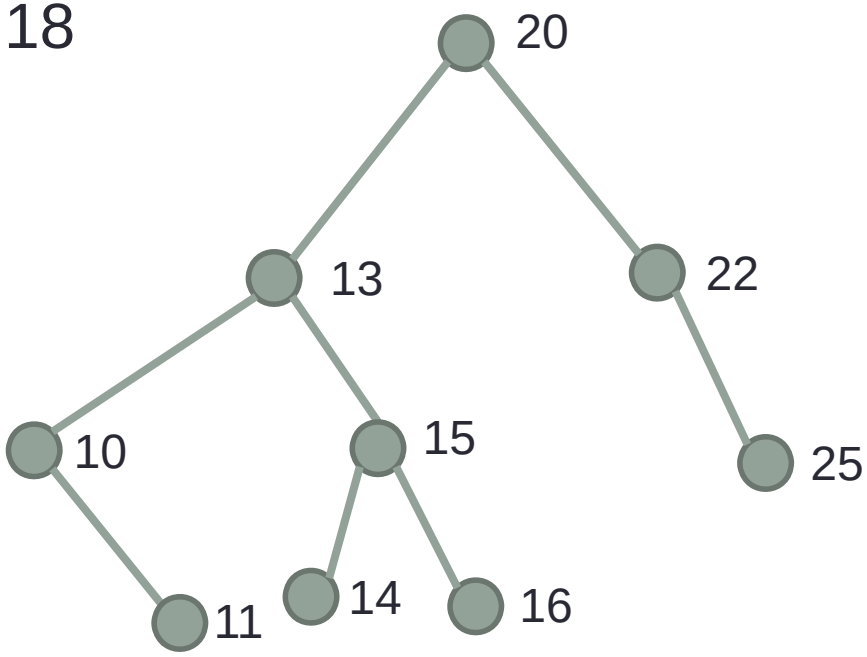
- 找到small裡面最小的(O(??))
- delete它(用前面講的刪node方法)
- 當成mid
- threeWayJoin(small, mid, big)

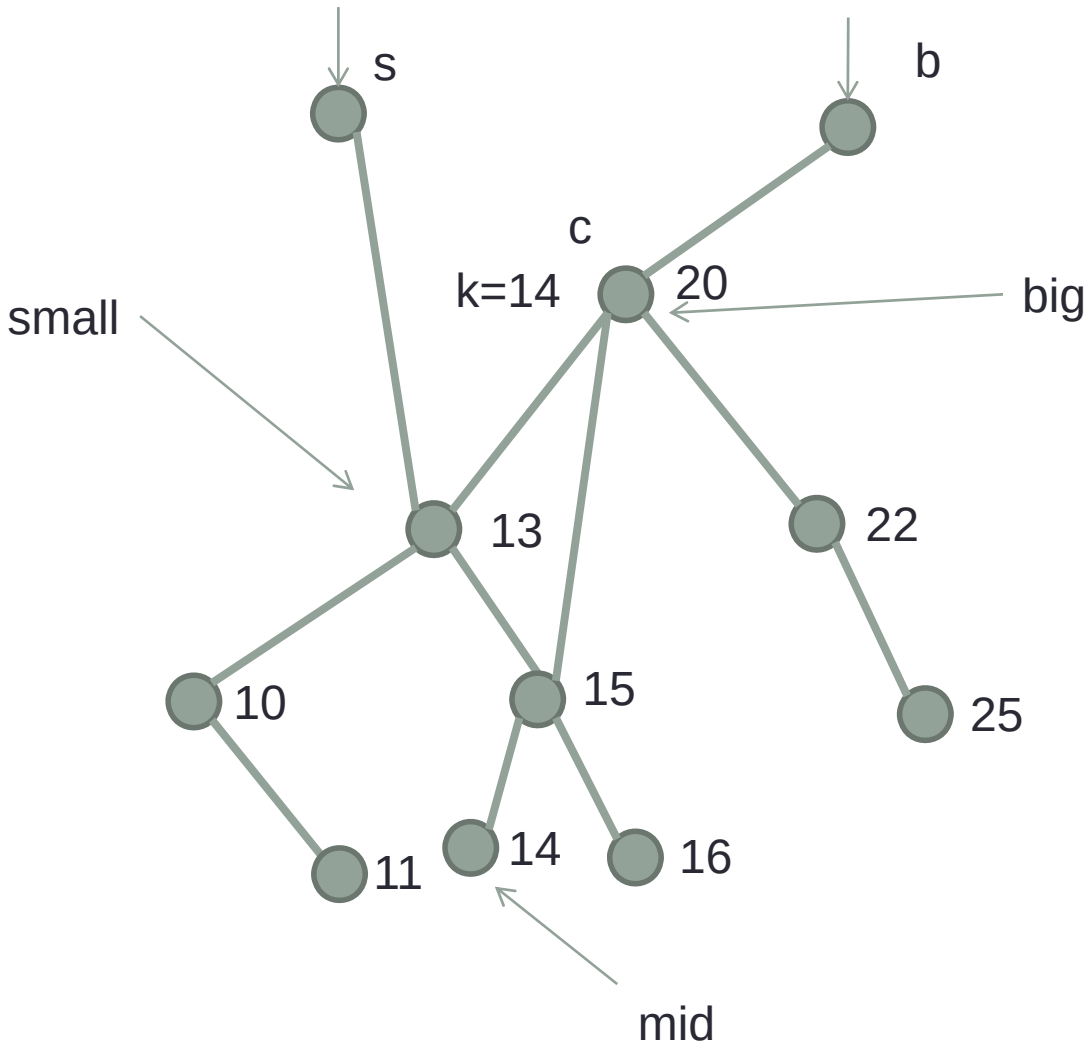
- height=?
- time complexity=O(??)



一些binary search tree的動作

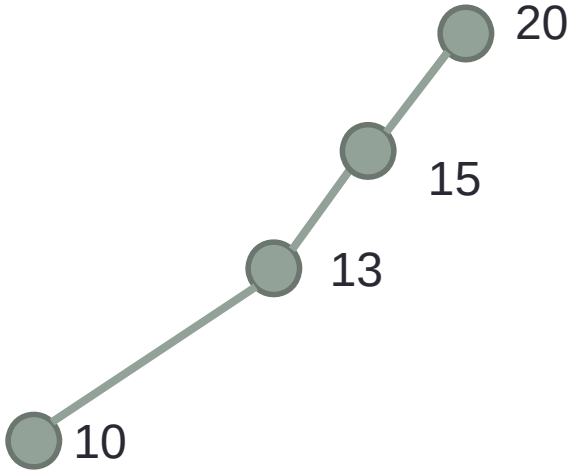
- split(tree,k,small,mid,big)
- 要把一棵binary search tree以k為分界切成small, mid, big
- small都比k小, big都比k大, mid跟k一樣
- 例子: k=14
- 課本program 5.18




s
b
c
k=14
20
small
big
13
22
10
15
25
11
14
16
mid

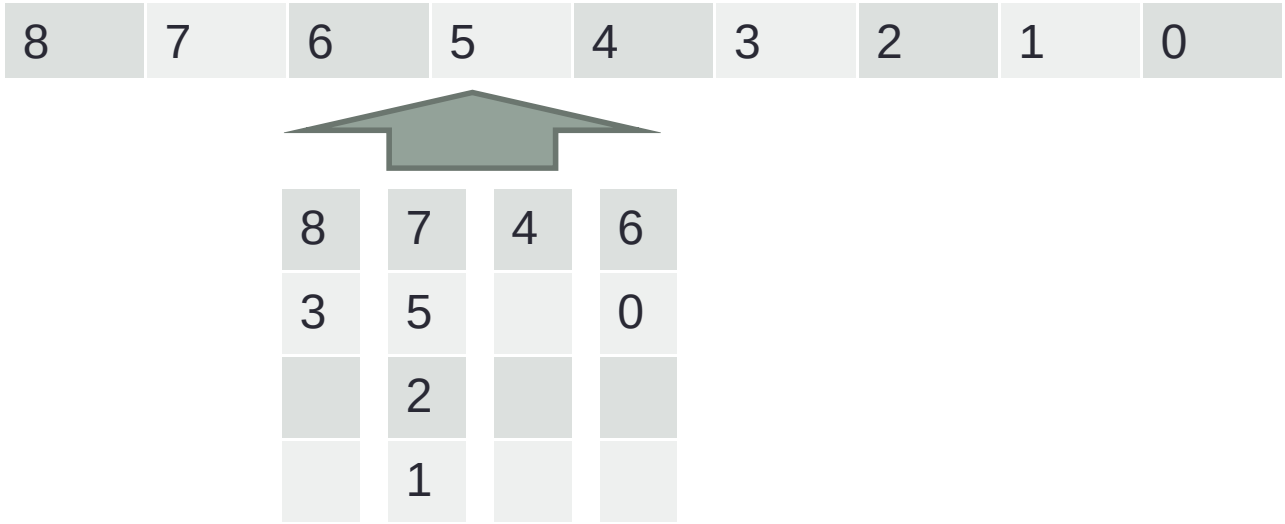
binary search tree的高度

- 高度跟node數目有何關係?
- h=O(log n)??
- No. In the worst case, h=O(n) ☹
- 之後chapter 10 & 11會講到h=O(log n)的”balanced search trees”



選擇樹

- 問題: 有k組排好的數列, 要合併成一組(共n個數字)
- 先來一個爛方法
- 每回合都比較各數列的頭頭(共k(k-1)次), 比較出最大的一個
- 然後放入最後的數列裡



- 這樣的話, time complexity=$O(nk^2)$

選擇樹

- 使用樹來記錄一些資訊:

- 精神: 上一回合比較的結果並不是全然無用
- (只有上一回合勝出的數字和別人的比較結果需要重做)

- 紀錄一些資訊使得下一回合不用全部重來

- 目標 time complexity = $O(n \log_2 k)$

勝者樹

- Tree裡面記得每次贏的數字
- 很像錦標賽

- 用黑板講解

- 有沒有什麼缺點?
- 每次都要和sibling比
- Linked representation的時候不好找
- 從上一回合贏的那個數列一直到root每個數字都要改掉(曼一點點)

敗者樹

- Tree裡面記得每次贏的數字
- 用黑板講解
- 這樣就不需要跟sibling比了
- 每回合跟parent比, 輸的留下, 贏的晉級
- time complexity=O(??)
- <動腦時間>如果$n \neq 2^i$, 怎麼處理?

講一些輕鬆一點的...

- 如果要拷貝binary tree怎麼做?
- 使用recursive, 會很簡單 ☺

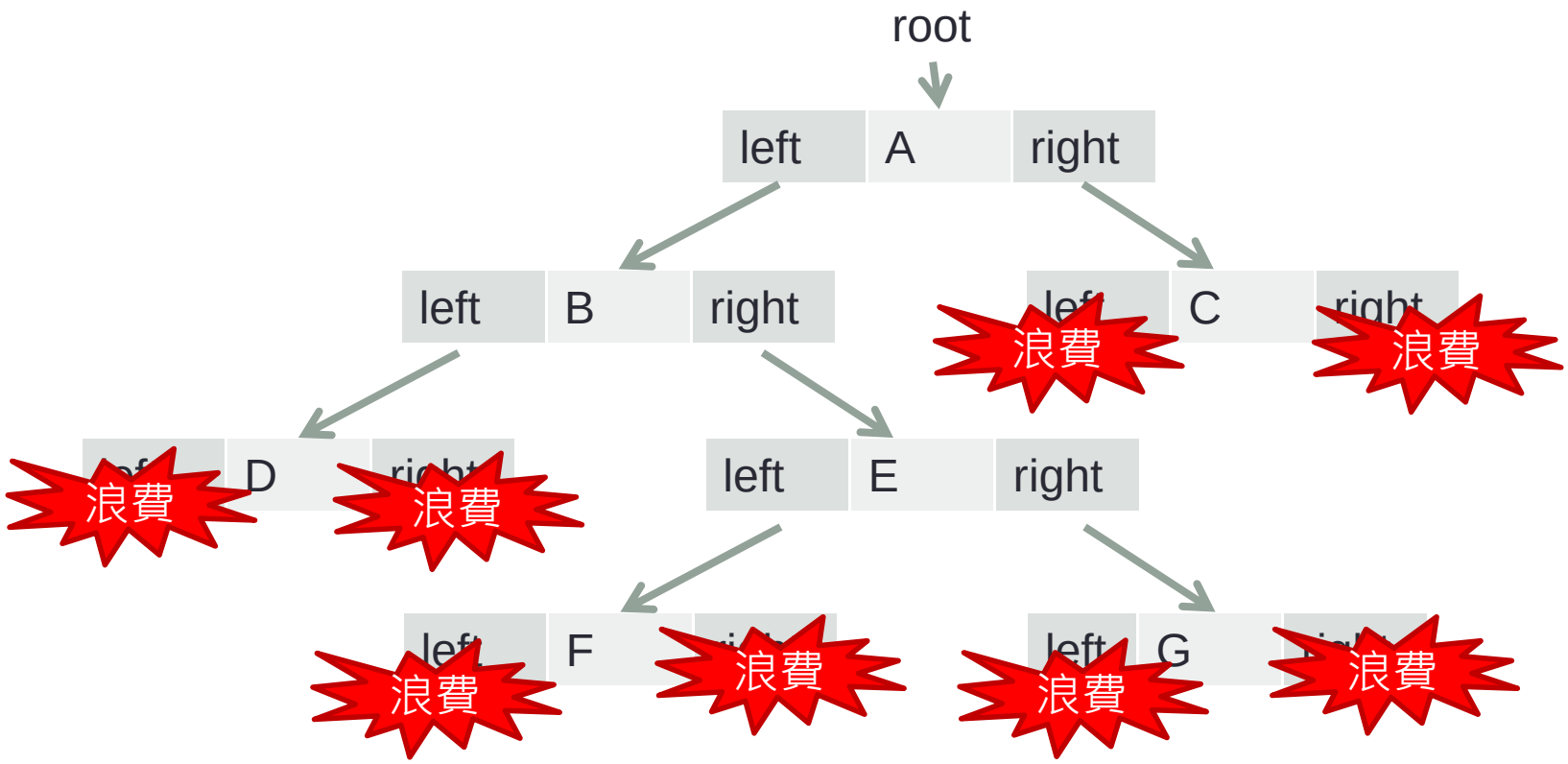
```
tPointer copy(tPointer original) {
        tPointer temp;
        if (original) {
                MALLOC(temp, sizeof(*temp));
                temp->leftChild=copy(original->leftChild);
                temp->rightChild=copy(original->rightChild);
                temp->data=original->data;
                return temp;
        }
        return NULL;
}
```

講一些輕鬆一點的...

- 如果要測試兩個binary tree有沒有一模一樣怎麼做?
- 一樣用recursive

```
int equal(tPointer first, tPointer second) {
        if (!first && !second) return TRUE;
        return (first->data == second->data) &&
                equal(first->leftChild, second->leftChild) &&
                equal(first->rightChild, second->rightChild);
}
```

線頭樹??


root
left
A
right
left
B
right
C
D
left
E
right
F
G
浪費
浪費
浪費
浪費
浪費
浪費
浪費

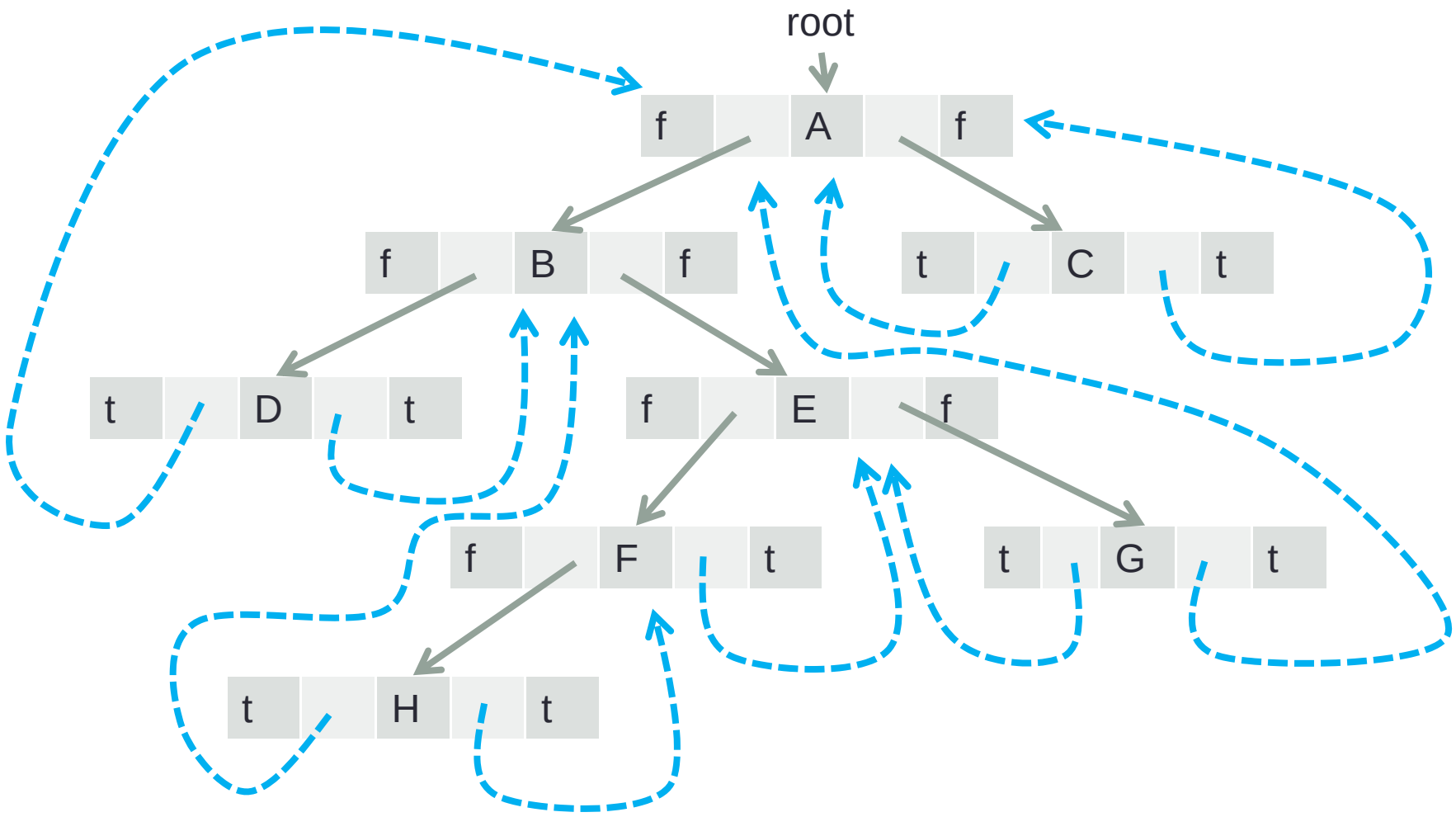
浪費掉的link field…

- 可以拿來存一些別的資訊
- Threaded Binary Tree

- 1. 如果leftChild是null, 那就改成指到inorder traversal的前一個node (又稱為inorder predecessor) (此為thread)
- 2. 如果rightChild是null,那就改成指到inorder traversal的後一個node (又稱為inorder successor) (此為thread)
- 3. node的structure裡面放兩個額外的boolean欄位, 說明是link還是thread

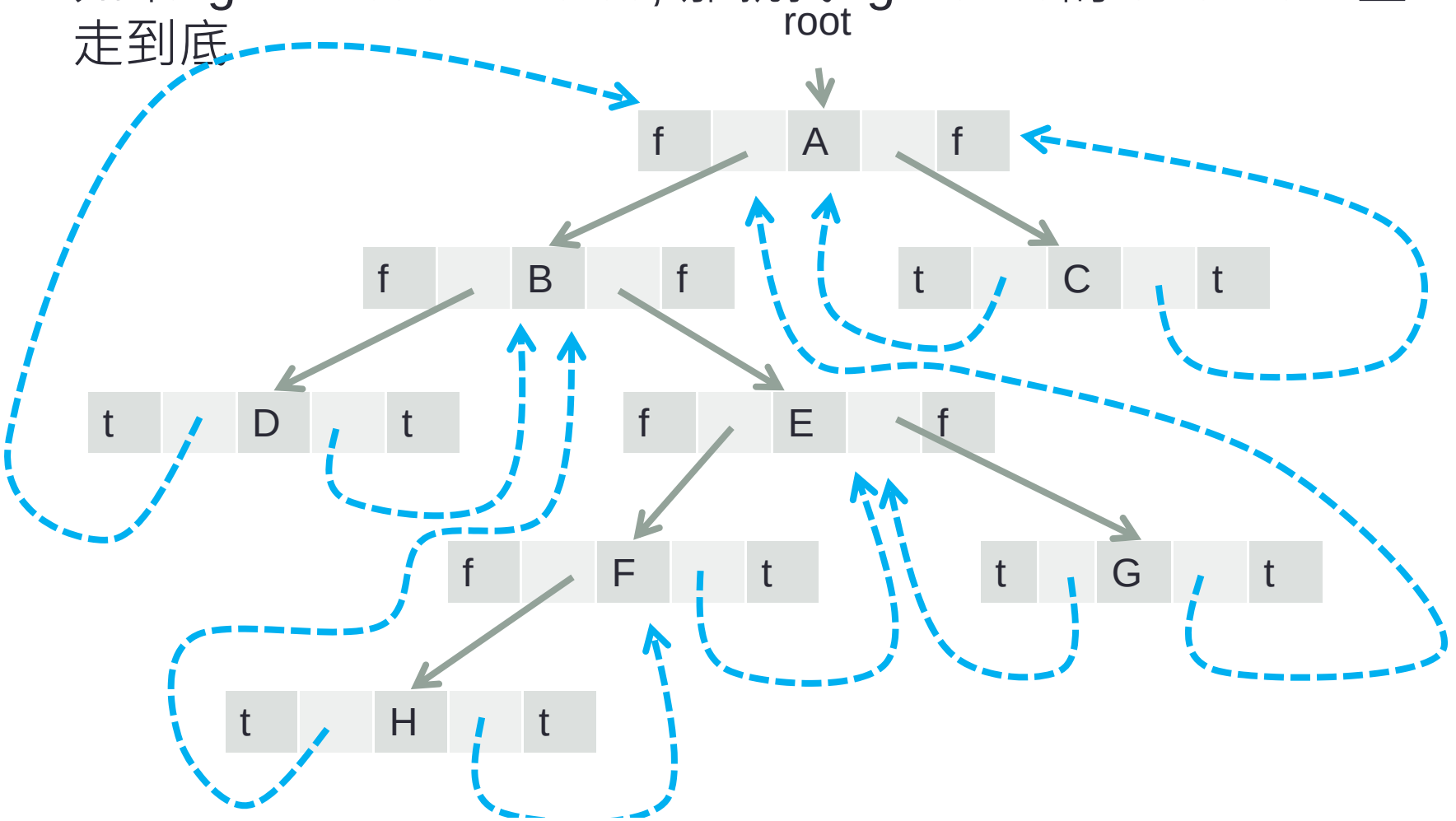
- 效果: 之後做inorder traversal不需要stack了!

Threaded binary tree長這樣



Threaded Binary Tree

- 怎麼找到inorder successor?
- 如果rightThread == true, 那麼就是thread指到的地方
- 如果rightThread == false, 那就找rightChild的leftChild一直走到底



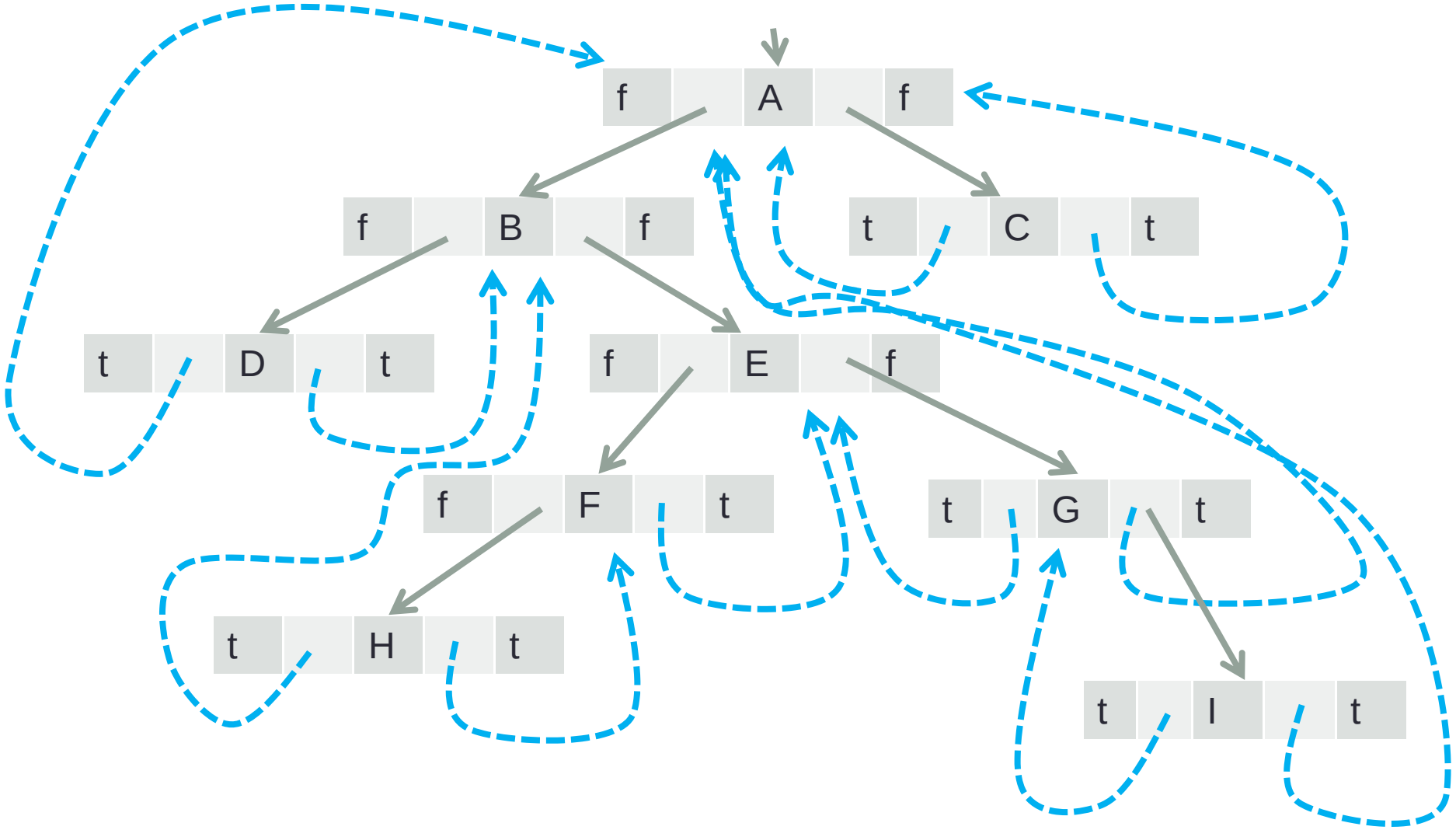
Threaded Binary Tree

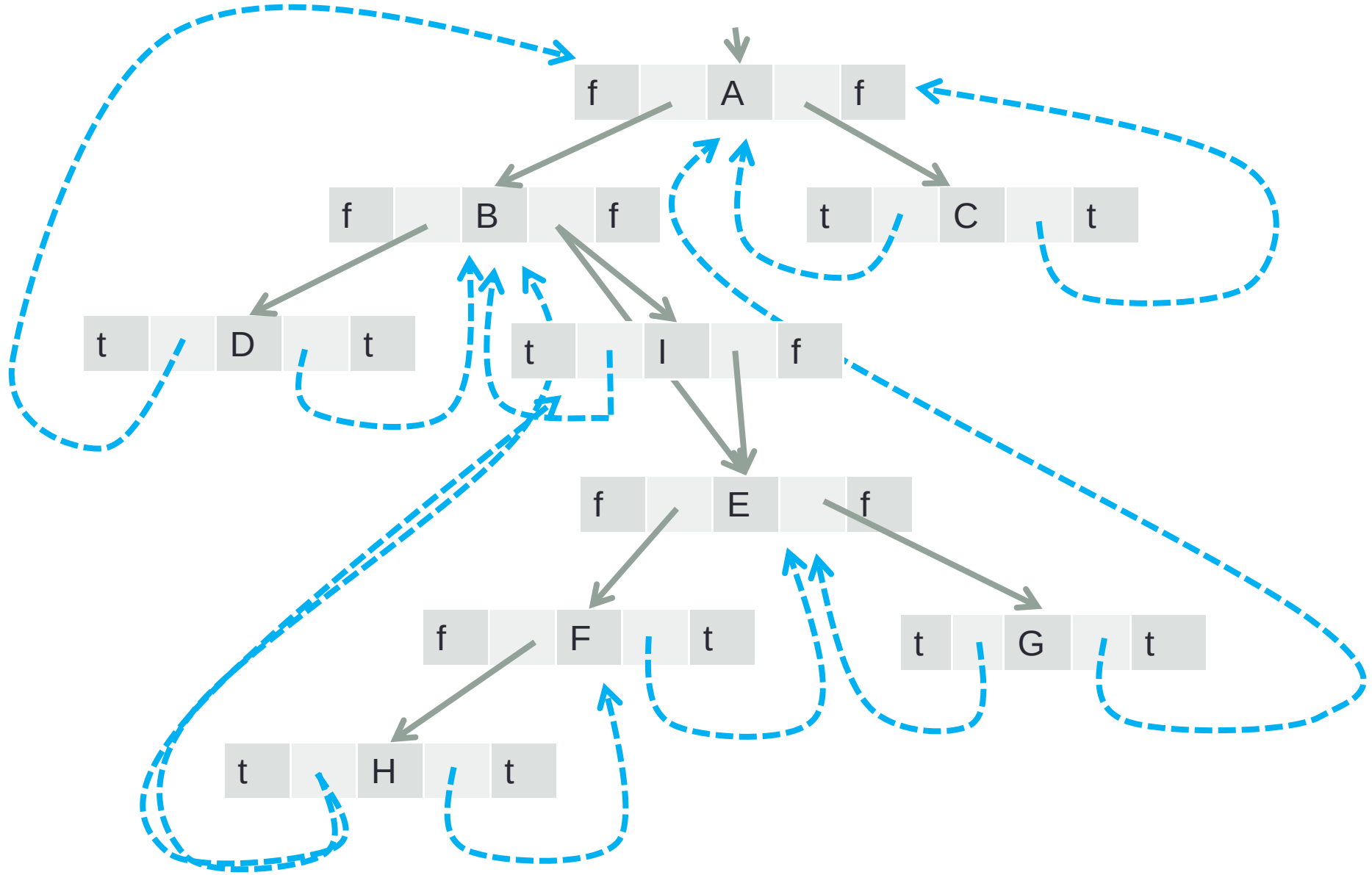
- 接著, 要做inorder traversal就很簡單了

```
for(;;) {
        temp=insucc(temp);
        if (temp==tree) break; //走回root了
        printf("%3c", temp->data);
}
```

<動腦時間> 如果要用threaded binary tree做postorder or preorder traversal呢?

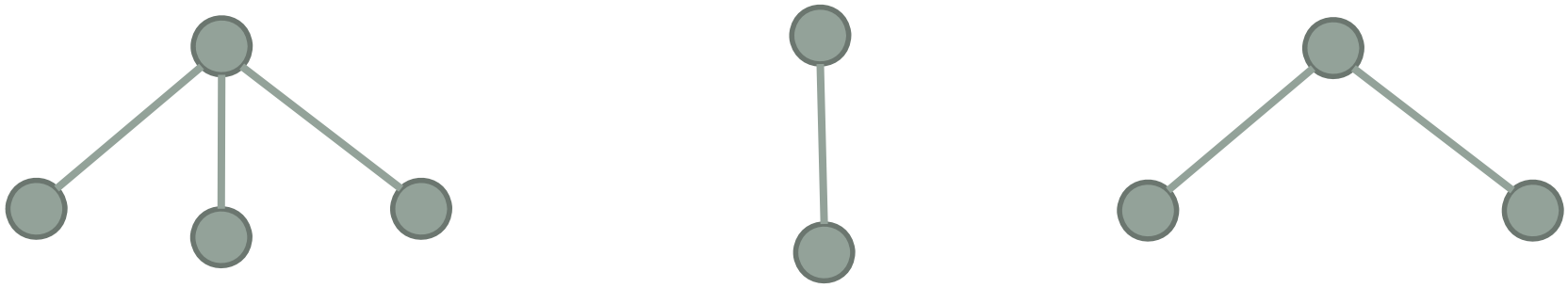
在Threaded Binary裡面加一個node




f A f
f B f
t C t
t D t
t I f
f E f
f F t
t G t
t H t

森林

- Definition: A forest is a set of $n \geq 0$ disjoint trees.
- 要怎麼把森林轉成一棵binary tree呢?
- 提示: 左小孩右兄弟姊妹法 + root右手沒有連東西



森林的traversal

- forest preorder
- (1) If F is empty then return
- (2) Visit the root of the first tree of F.
- (3) Traverse the subtrees of the first tree in forest preorder (把第一個tree的subtrees當成forest)
- (4) Traverse the remaining trees of F in forest preorder
- → 對應到轉換後的binary tree的preorder traversal
- forest inorder →對應到轉換後的binary tree的inorder traversal
- forest postorder →沒有對應
- forest level-order

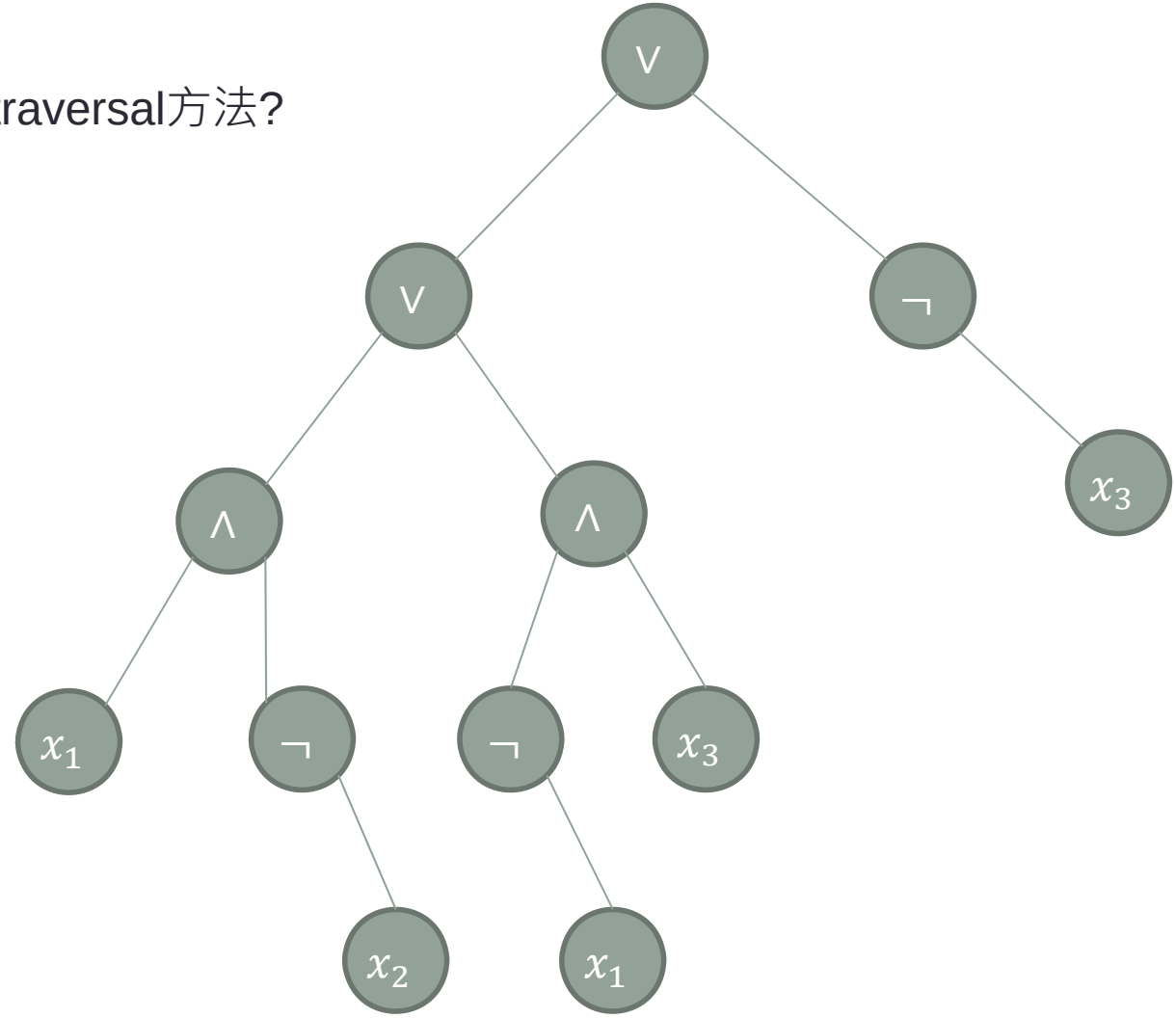
計算邏輯運算式

- 變數可為true or false
- 三種operator: $\neg\,(and)$,$\wedge\,(or)$,$\vee\,(and)$
- 可以使用括號
- 例如 $(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_3) \vee \neg x_3$
- 如何計算當$(x_1, x_2, x_3) = (t, t, f)$時的結果?
- 進階題: 如何找出所有組合使得結果為true?

計算邏輯運算式

$(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_3) \vee \neg x_3$

用什麼traversal方法?



周末愉快

- 作業一已經改好
- 改好的作業下周會發給同學們.
- 成績分布會公布
- 請比較低分的同學多加留意!
- 多利用office hour (下周二三四)
- 請同學準備好下周要發問的問題
- 我也會準備一些我覺得重要的課題來複習