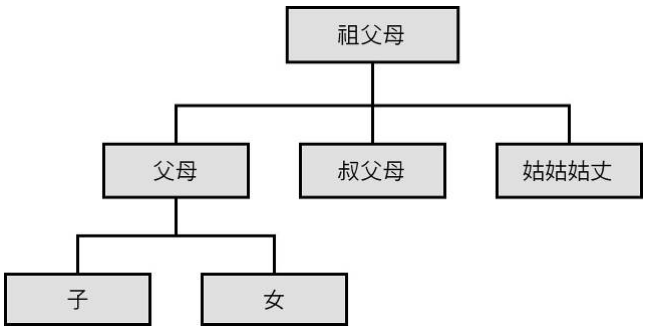
資料結構的樹與二元樹（Trees and Binary Trees）

資訊科技系
林偉川

樹的基本觀念

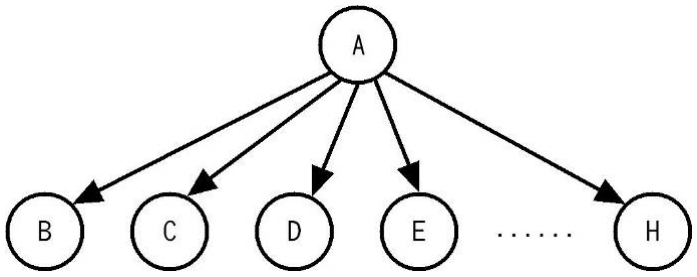
- 「樹」（Trees）是一種模擬現實生活中樹幹和樹枝的資料結構，屬於一種階層架構的非線性資料結構，例如：家族族譜，如下圖所示：



2

樹的基本觀念

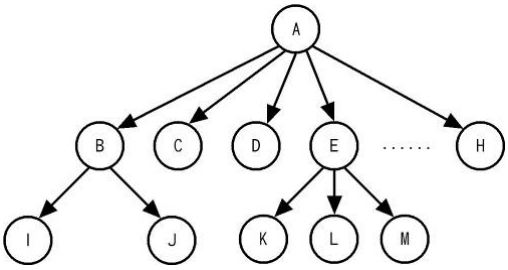
- 樹的樹根稱為「根節點」（Root），在根節點之下是樹的樹枝，擁有0到n個「子節點」（Children），即樹的「分支」（Branch），節點A是樹的根節點，B、C、D....和H是節點A的子節點，即樹枝，如下圖所示：



3

樹的基本觀念

- 在樹枝下還可以擁有下一層樹枝，I和J是B的子節點，K、L和M是E的子節點，節點B是I和J的「父節點」（Parent），節點E是K、L和M的父節點，節點I和J擁有共同父節點，稱為「兄弟節點」（Siblings），K、L和M 是兄弟節點，B、C...和H節點也是兄弟節點，如下圖所示：



4

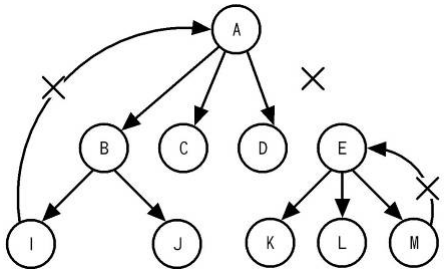
樹的基本觀念

定義 1：樹的節點個數是一或多個有限集合，且：

(1) 存在一個節點稱為根節點。

(2) 在根節點下的節點分成n >= 0 個沒有交集的多個子集合 t1、t2…, tn，每一個子集合也是一棵樹，而這些樹稱為根節點的「子樹」（Subtree）。

- 樹在各節點之間不可以有迴圈，或不連結的左、右子樹，如下圖所示：



5

樹的基本觀念

- n元樹：樹的一個節點最多擁有n個子節點。
- 二元樹（Binary Trees）：樹的節點最多只有兩個子節點。
- 根節點（Root）：沒有父節點的節點是根節點。例如：節點A。
- 葉節點（Leaf）：節點沒有子節點的節點稱為葉節點。例如：節點I、J、C、D、K、L、M、F、G和H。
- 祖先節點（Ancenstors）：指某節點到根節點之間所經過的所有節點，都是此節點的祖先節點。

6

樹的基本觀念

- 非終端節點（Noterminal Nodes）：除了葉節點之外的其它節點稱為非終端節點。例如：節點A、B和E是非終端節點。
- 分支度（Dregree）：指每個節點擁有的子節點數。例如：節點B的分支度是2，節點E的分支度是3。
- 階層（Level）：如果樹根是1，其子節點是2，依序可以計算出樹的階層數。例如：上述圖例的節點A階層是1，B、C到H是階層2，I、J到M是階層3。
- 樹高（Height）：樹高又稱為樹深（Depth），指樹的最大階層數。例如：上述圖例的樹高是3。

7

二元樹的基礎

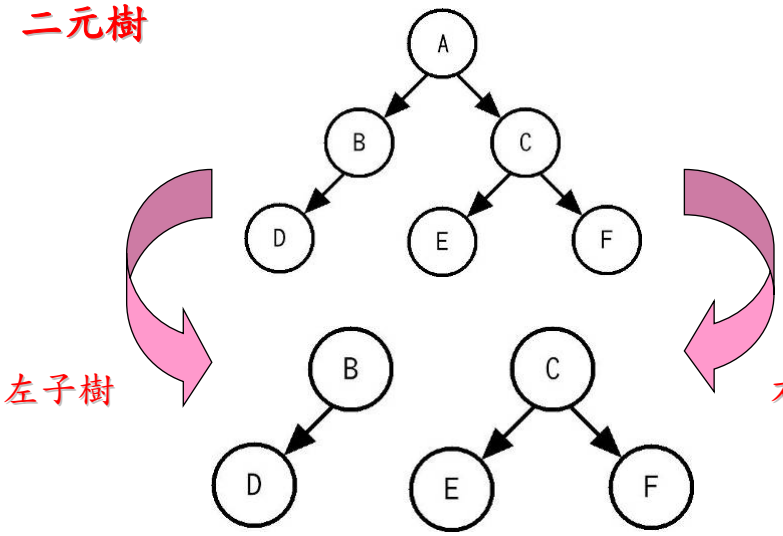
- 樹依不同分支度可以區分成很多種，在資料結構中最廣泛使用的樹狀結構是「二元樹」，二元樹是指樹中的每一個「節點」（Nodes）最多只能擁有2個子節點，即分支度小於或等於2。
- 二元樹的定義如下所示：

定義2：二元樹的節點個數是一個有限集合，或是沒有節點的空集合。二元樹的節點可以分成兩個沒有交集的子樹，稱為「左子樹」（Left Subtree）和「右子樹」（Right Subtree）。

8

二元樹的基礎

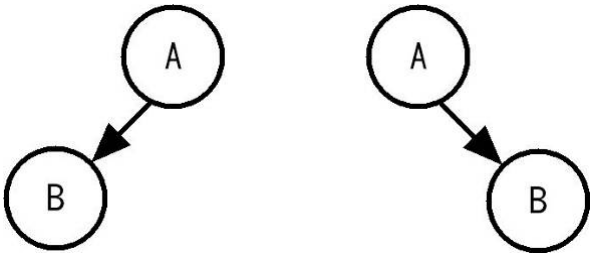
二元樹



9

歪斜樹

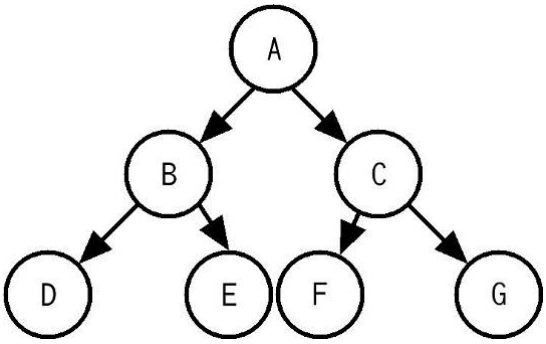
- 左邊這棵樹沒有右子樹，右邊這棵樹沒有左子樹，雖然擁有相同節點，但是這是兩棵不同的二元樹，因為所有節點都是向左子樹或右子樹歪斜，稱為「歪斜樹」（Skewed Tree），如下圖所示：



10

完滿二元樹

- 若二元樹的樹高是h且二元樹的節點數是2^h-1，滿足此條件的樹稱為「完滿二元樹」（Full Binary Tree），如下圖所示：



11

完滿二元樹

- 因為二元樹的每一個節點有2個子節點，二元樹樹高是3，也就是有3個階層（Level），各階層的節點數，如下所示：

  第1階：$1 = 2^{(1-1)} = 2^0 = 1$

  第2階：第1階節點數的2倍，$1*2 = 2^{(2-1)} = 2$

  第3階：第2階節點數的2倍，$2*2 = 2^{(3-1)} = 4$

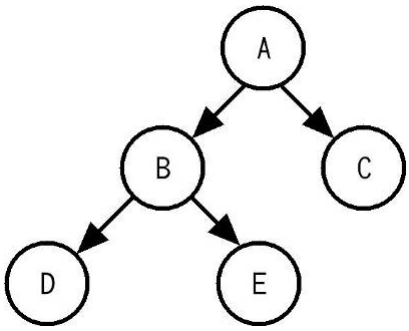
- 以此類推，可以得到每一階層的最大節點數是：$2^{(l-1)}$，l是階層數，整棵二元樹的節點數一共是：$2^0+2^1+2^2 = 7$個，即2^{3-1}，可以得到：

  $2^0+2^1+2^2+\ldots.+2^{(h-1)} = 2^{h-1}$，h是樹高

12

完整二元樹

- 若二元樹的節點不是葉節點，一定擁有兩個子節點，不過節點總數不足2^{h-1}，其中h是樹高，滿足此條件的二元樹稱為「完整二元樹」（Complete Binary Tree），如下圖所示：



13

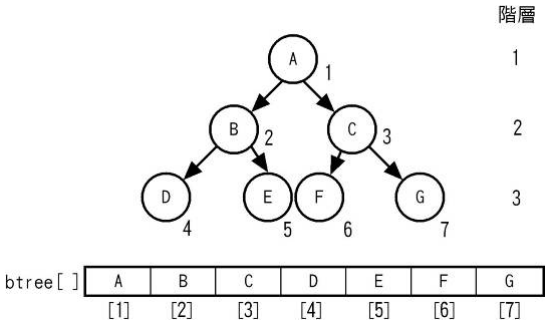
二元樹的表示法

- 二元樹在實作上有多種方法可以建立二元樹，常用的方法有三種，如下所示：
  - 二元樹陣列表示法。
  - 二元樹結構陣列表示法。
  - 二元樹鏈結表示法。

14

二元樹陣列表示法

- 完滿二元樹是一棵樹高h擁有2^{h-1}個節點的二元樹，這是二元樹在樹高h所能擁有的最大節點數，換句話說，只需配置2^{h-1}個元素，我們就可以儲存樹高h的二元樹，如下圖所示：



15

二元樹陣列表示法

- 二元樹的節點編號擁有循序性，根節點1的子節點是節點2和節點3，節點2是4和5，依此類推可以得到節點編號的規則，如下所示：
  - 左子樹是父節點編號乘以2。
  - 右子樹是父節點編號乘以2加1。

16

二元樹陣列表示法

```
02: #define MAX_LENGTH   16 /* 最大陣列尺寸 */
03: int btree[MAX_LENGTH];  /* 二元樹陣列宣告 */
04: /* 抽象資料型態的操作函數宣告 */
05: extern void createBTree(int len, int *array);
06: extern void printBTree();
```

17

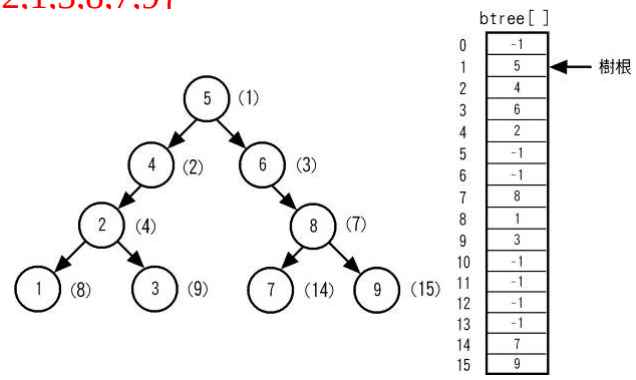
二元樹陣列表示法-建立二元樹

- 函數createBTree()讀取一維陣列的元素建立二元樹，其建立的規則，如下所示：
  - 將第1個陣列元素插入成為二元樹的根節點。
  - 將陣列元素值與二元樹的節點值比較，如果元素值大於節點值，將元素值插入成為節點的右子節點，如果右子節點不是空的，重覆比較節點值，直到找到插入位置後，將元素值插入二元樹。
  - 如果元素值小於節點值，將元素值插入成為節點的左子節點，如果左子節點不是空的，繼續重覆比較，以便將元素值插入二元樹。

18

二元樹陣列表示法-建立二元樹

- 二元樹陣列表示法圖例的索引值0並沒有使用，整個二元樹在16個陣列元素中使用的元素一共有9個，括號內是陣列的索引值，如下圖所示：
- {5,4,6,2.1.3.8.7.9}



19

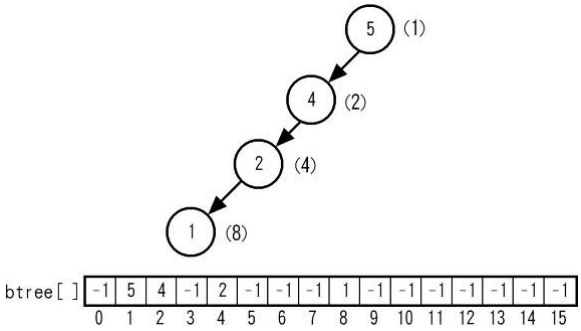
二元樹陣列表示法-顯示二元樹

- 函數printBTree()：顯示二元樹
  - 走訪btree[]陣列，將元素值不是-1的元素都顯示出來。

20

二元樹陣列表示法

- 一棵歪斜樹的二元樹陣列表示法使用不到三分之一的陣列元素4/16，因為二元樹的節點是以循序方式儲存在陣列中，如果需要插入或刪除節點，都需要在陣列中搬移大量元素，如下圖所示：



21

二元樹結構陣列表示法

- 在二元樹的每一個節點可以使用C語言的結構來儲存，整棵二元樹使用一個結構陣列，每一個結構是一個節點，使用結構陣列儲存整棵二元樹，data是節點資料，使用left和right成員變數的索引值指向子節點的索引值，如為-1表示沒有子節點，如下圖所示：

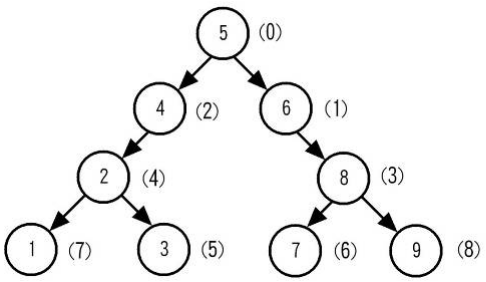
22

二元樹結構陣列表示法

```
02: #define MAX_LENGTH   16    /* 最大陣列尺寸 */
03: struct Node {              /* 二元樹的結構宣告 */
04:    int data;               /* 節點資料 */
05:    int left;               /* 指向左子樹的位置 */
06:    int right;              /* 指向右子樹的位置 */
07: };
08: typedef struct Node TreeNode; /* 樹的節點新型態 */
09: TreeNode btree[MAX_LENGTH];
10: /* 抽象資料型態的操作函數宣告 */
11: extern void createBTree(int len, int *array);
12: extern void printBTree();
```


二元樹結構陣列表示法

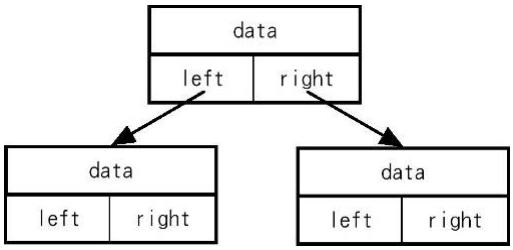
- 例如：一棵二元樹和其結構陣列表示法，如下圖所示：



	left	data	right
0	2	**5**	1
1	-1	**6**	3
2	4	**4**	-1
3	6	**8**	8
4	7	**2**	5
5	-1	**3**	-1
6	-1	**7**	-1
7	-1	**1**	-1
8	-1	**9**	-1

二元樹鏈結表示法

- 二元樹鏈結表示法是使用動態記憶體配置來建立二元樹，類似結構陣列表示法的節點結構，只是成員變數改成兩個指向左和右子樹的指標，如下圖所示：



25

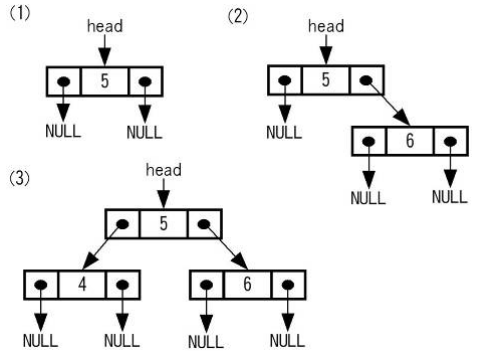
二元樹鏈結表示法

```
02: struct Node {           /* 二元樹的節點宣告 */
03:    int data;            /* 儲存節點資料 */
04:    struct Node *left;   /* 指向左子樹的指標 */
05:    struct Node *right;  /* 指向右子樹的指標 */
06: };
07: typedef struct Node TNode;  /* 二元樹節點的新型態 */
08: typedef TNode *BTree;       /* 二元樹鏈結的新型態 */
09: BTree head = NULL;          /* 二元樹根節點的指標 */
10: /* 抽象資料型態的操作函數宣告 */
11: extern void createBTree(int len, int *array);
12: extern void insertBTreeNode(int d);
13: extern int isBTreeEmpty();
14: extern void printBTree();
15: extern void inOrder(BTree ptr);
16: extern void printInOrder();
17: extern void preOrder(BTree ptr);
18: extern void printPreOrder();
19: extern void postOrder(BTree ptr);
20: extern void printPostOrder();
```

26

二元樹鏈結表示法

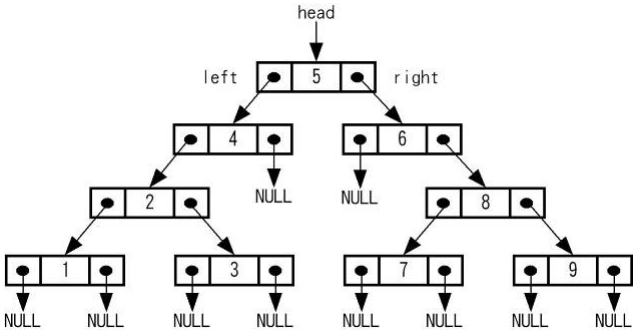
- 函數createBTree()使用for迴圈走訪參數的陣列元素，依序呼叫insertBTreeNode()函數將一個一個陣列元素的節點插入二元樹。首先是二元樹的根節點5，left和right指標指向NULL，如下圖所示：



27

二元樹鏈結表示法-建立二元樹2

- 第二次呼叫insertBTreeNode()函數插入元素6，鏈結至右子樹。第三次呼叫插入元素4成為左子樹。等到執行完createBTree()函數的for迴圈後，建立的二元樹，如下圖所示：



28

二元樹鏈結表示法-顯示二元樹

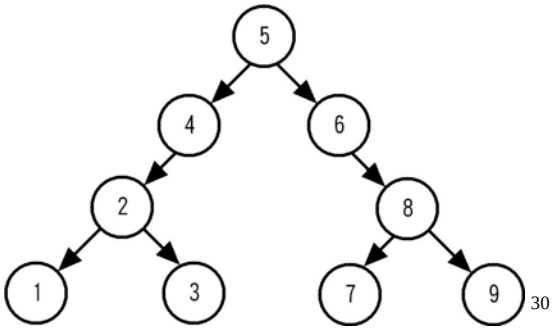
函數printBTree()：顯示二元樹

- 函數printBTree()使用while迴圈分別走訪二元樹的左右分支，不過這個函數並沒有辦法顯示二元樹的全部節點資料。

29

二元樹的走訪

- 陣列和單向鏈結串列都只能從頭至尾或從尾至頭執行單向「走訪」（Traverse），不過二元樹的每一個節點都擁有指向左和右2個子節點的指標，所以走訪可以有兩條路徑。例如：一棵二元樹，如下圖所示：



30

二元樹的走訪-種類

- 二元樹的走訪過程是持續決定向左或向右走，直到沒路可走。很明顯的！二元樹的走訪是一種遞迴走訪，依照遞迴函數中呼叫的排列順序不同，可以分成三種走訪方式，如下所示：
  - 中序走訪方式（Inorder Traversal）。
  - 前序走訪方式（Preorder Traversal）。
  - 後序走訪方式（Postorder Traversal）。

31

中序走訪方式

- 中序走訪是沿著二元樹的左方往下走，直到無法繼續前進後，顯示節點，退回到父節點顯示父節點，然後繼續往右走，如果右方都無法前進，顯示節點，再退回到上一層。其顯示的節點順序，如下所示：

  1,2,3,4,5,6,7,8,9
- 在上述中序走訪節點順序中，可以看出根節點5是位在正中間，之前都是左子樹的節點，之後都是右子樹的節點。

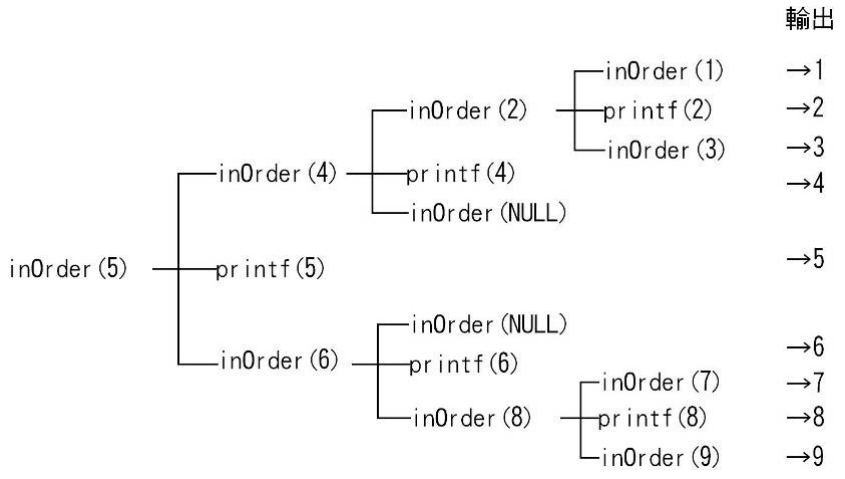
32

中序走訪方式

- 中序走訪的遞迴函數inOrder()使用二元樹指標ptr進行走訪，中序走訪的步驟，如下所示：
  - Step 1：檢查是否可以繼續前進，即指標ptr不等於NULL。
  - Step 2：如果可以前進，其處理方式如下所示：
    - (1) 遞迴呼叫inOrder(ptr->left)向左走。
    - (2) 處理目前的節點，顯示節點資料。
    - (3) 遞迴呼叫inOrder(ptr->right)向右走。

33

中序走訪方式-遞迴呼叫



34

前序走訪方式

- 前序走訪方式是走訪到的二元樹節點，就立刻顯示節點資料，走訪的順序是先向樹的左方走直到無法前進後，才轉往右方走。其顯示的節點順序，如下所示：

  5,4,2,1,3,6,8,7,9

- 在上述前序走訪節點順序中，可以看出根節點5一定是第1個，左右子樹的根節點一定在其它節點之前。

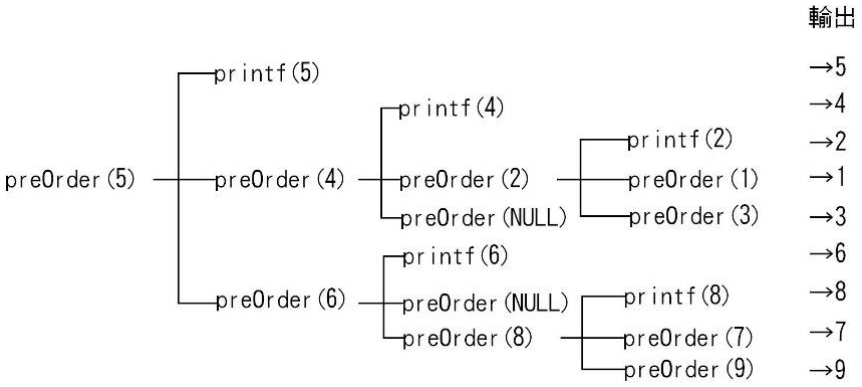
35

前序走訪方式

- 前序走訪的遞迴函數preOrder()使用二元樹指標ptr進行走訪，前序走訪的步驟，如下所示：
  - Step 1：先檢查是否已經到達葉節點，也就是指標ptr等於NULL。
  - Step 2：如果不是葉節點表示可以繼續走，其處理方式如下所示：
    - (1) 處理目前的節點，顯示節點資料。
    - (2) 遞迴呼叫preOrder(ptr->left)向左走。
    - (3) 遞迴呼叫preOrder(ptr->right)向右走。

36

前序走訪方式-遞迴呼叫



後序走訪方式

- 後序走訪方式剛好和前序走訪相反，它是等到節點的2個子節點都走訪過後才執行處理，顯示節點資料。依照後序走訪的二元樹，其顯示的節點順序，如下所示：

  1,3,2,4,7,9,8,6,5

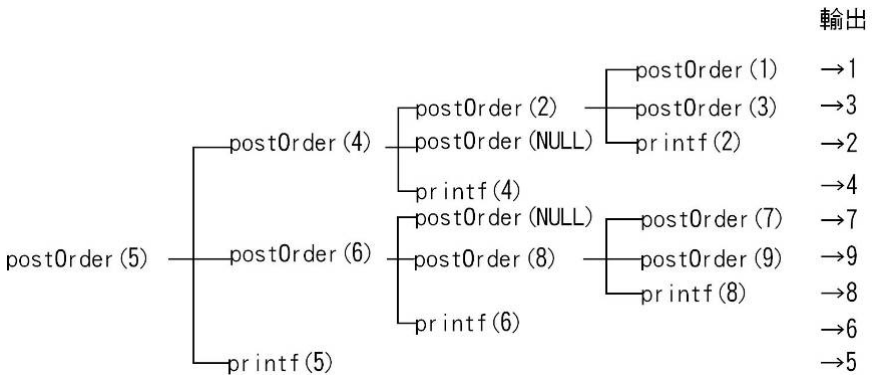
- 在上述後序走訪節點順序中，可以看出根節點5是最後1個，而且左右子樹的根節點一定在其它節點之後。

後序走訪方式

- 後序走訪的遞迴函數postOrder()使用二元樹指標ptr進行走訪，後序走訪的步驟，如下所示：
  - Step 1：先檢查是否已經到達葉節點，就是指標ptr等於NULL。
  - Step 2：如果不是葉節點表示可以繼續走，其處理方式如下所示：
    - (1) 遞迴呼叫postOrder(ptr->left)向左走。
    - (2) 遞迴呼叫postOrder(ptr->right)向右走。
    - (3) 處理目前的節點，顯示節點資料。

39

後序走訪方式-遞迴呼叫



40

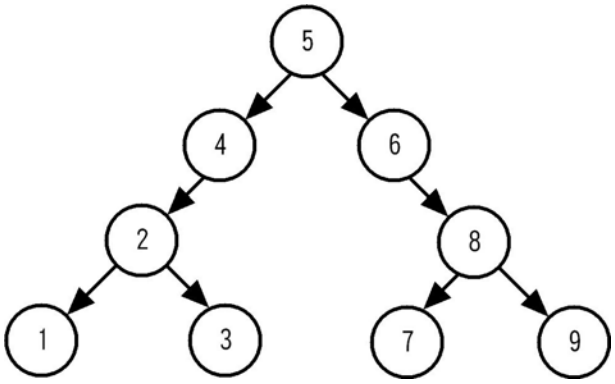
二元搜尋樹

- 「二元搜尋樹」（Binary Search Trees）是一種二元樹，其節點資料的排列擁有一些特性，如下所示：
  - 二元樹的每一個節點值都不相同，在整棵二元樹中的每一個節點都擁有不同值。
  - 每一個節點的資料大於左子節點（如果有的話）的資料，但是小於右子節點（如果有的話）的資料。
  - 節點的左、右子樹也是一棵二元搜尋樹。

41

二元搜尋樹

- 例如：在前所建立的二元樹就是一棵二元搜尋樹，如下圖所示：



42

二元搜尋樹的節點搜尋

- 二元搜尋樹的節點搜尋十分簡單，因為右子節點的值一定大於左子節點，所以只需從根節點開始比較，就知道搜尋值是位在右子樹或左子樹，繼續往子節點進行比較，就可以找出是否擁有指定的節點值。
- 例如：在前所描述的二元搜尋樹找尋節點資料8，第一步與樹根5比較，因為比較大，所以節點在右子樹，接著和右子樹的節點6比較，還是比較大，所以繼續向右子樹走，然後是節點8，只需三次比較就可以找到搜尋值。

43

二元搜尋樹的節點搜尋

- 在C程式是使用while迴圈配合ptr指標（指向根節點head）進行各子節點資料的比較，以便執行二元搜尋樹的搜尋，如下所示：

```
while ( ptr != NULL ) {
  if ( ptr->data == d )
    return ptr;
  else
    if ( ptr->data > d ) ptr = ptr->left;
    else          ptr = ptr->right;
}
```

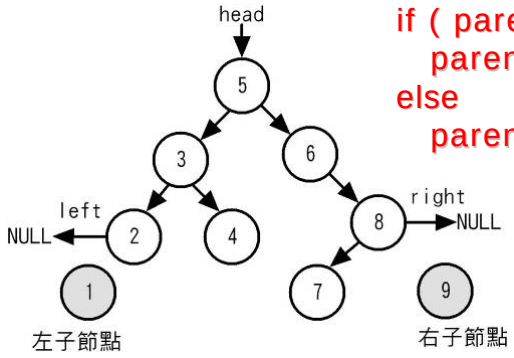
- 上述while迴圈的if條件判斷是否找到，如果節點值比搜尋值大，ptr = ptr->left向左子樹找，否則，ptr = ptr->right向右子樹找。

44

二元搜尋樹的節點刪除-情況1

情況1：刪除葉節點

- 葉節點是指沒有左和右子節點的節點，例如：刪除二元搜尋樹的葉節點1和9，如下圖所示：

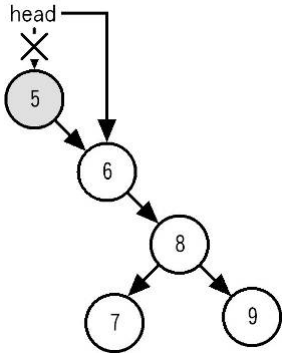



```
if ( parent->left == ptr )
   parent->left = NULL;
else
   parent->right = NULL;
```


二元搜尋樹的節點刪除-情況2

情況2：刪除節點沒有左子樹

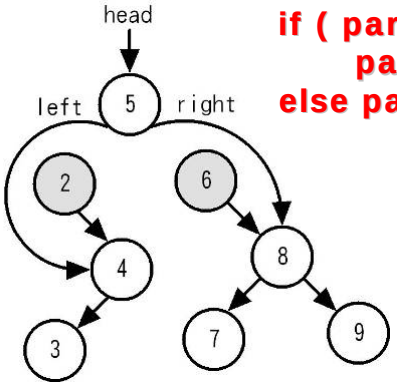
- **根節點**：刪除根節點5，只需將根節點指標指向其右子樹節點，如下圖所示：




```
head = head->right;
```

二元搜尋樹的節點刪除-情況2

- 中間節點：如果刪除的是中間節點2和6，這兩個節點都沒有左子樹，此時是將刪除節點的父節點指向其右子節點即可，如下圖所示：




```
if ( parent->left == ptr )
     parent->left = ptr->right;
else parent->right = ptr->right;
```

47

二元搜尋樹的節點刪除-情況3

情況3：刪除節點沒有右子樹

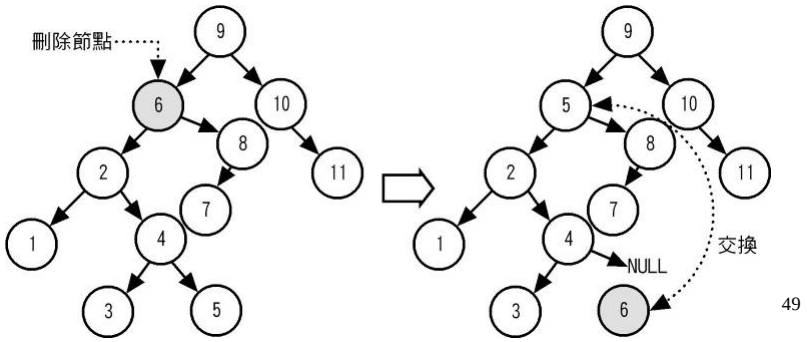
- 如果節點沒有右子樹，在此情況刪除節點，依節點的位置一樣可以分成二種：根節點和中間節點，節點刪除和情況2相似，只是左指標和右指標的交換。

48

二元搜尋樹的節點刪除-情況4

情況4：刪除節點擁有左子樹和右子樹

- 刪除節點如果擁有左子樹和右子樹，其處理方式並不會因刪除節點的位置而不同。例如：一棵二元搜尋樹，如下圖所示：



49

二元搜尋樹的節點刪除-情況4

- 在二元搜尋樹刪除節點6，它是父節點9的左子樹，如果可以找到節點位在節點2和節點8之間，將它取代成刪除節點的位置，如此並不需要搬移太多節點，就可以完成節點刪除。
  - 例如：刪除節點6事實上是刪除原來的葉節點5，因為刪除操作是交換這兩個節點來完成。
- 從二元搜尋樹的特性可以看出符合條件的交換節點有兩個，如下所示：
  - 節點5：從節點6的左子節點2一直從右子樹走到的葉節點
  - 節點7：從節點6的右子節點8一直往左子樹走到的葉節點。

50

二元搜尋樹的節點刪除-情況4

- 筆者使用第一種方式找出符合條件的節點，即節點5，程式碼如下所示：

```
parent = ptr;
child = ptr->left;
while ( child->right!=NULL ) {
  parent = child;
  child = child->right;
}
```

- 上述parent是父節點，ptr是刪除節點，child是子節點，在先走到左子節點後，使用while迴圈往右子節點走，直到葉節點。

51

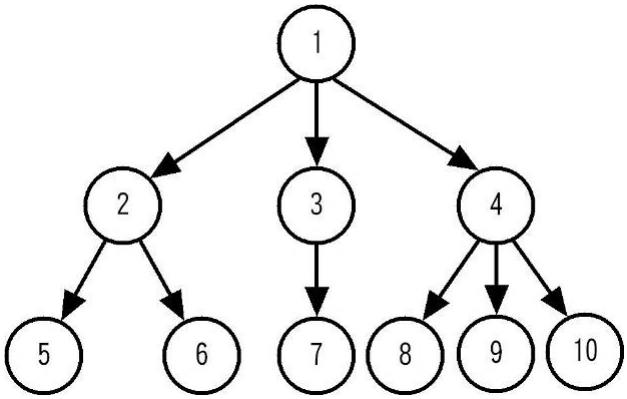
樹的二元樹表示法

- 二元樹在樹狀結構佔有十分重要的地位，這是因為所有樹都可以經過轉換，將它轉換成二元樹。例如：n元樹狀結構的每個節點擁有n個分支，處理不同數分支的節點都需要設計不同表示方法的程式碼，例如：二元樹需要2個指標，三個分支需要3個指標，以此類推。
- 不只如此，n元樹的NULL指標問題比二元樹更加嚴重，因為葉節點將擁有分支數個數的NULL指標，所以可以將樹先轉換成二元樹，直接使用二元樹表示法來建立樹狀結構。

52

樹的二元樹表示法-過程1

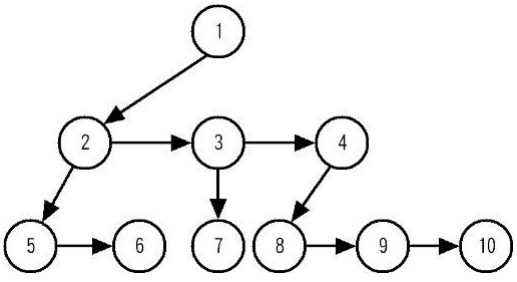
- 例如：一棵樹，如下圖所示：



53

樹的二元樹表示法-過程2

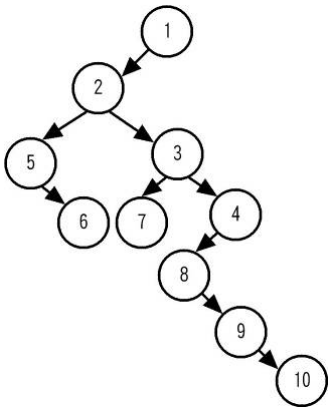
- 將樹轉換成二元樹，也就是將n個分支變成2個分支，只需把每個擁有同一個父節點的兄弟節點，將這些兄弟節點鏈結起來，保留最左邊的父子鏈結，將其它父子鏈結都打斷，就可以產生一棵二元樹，如下圖所示：



54

樹的二元樹表示法-過程3

- 接著將鏈結方向調整一下，就可以得到一棵二元樹，如下圖所示：



55

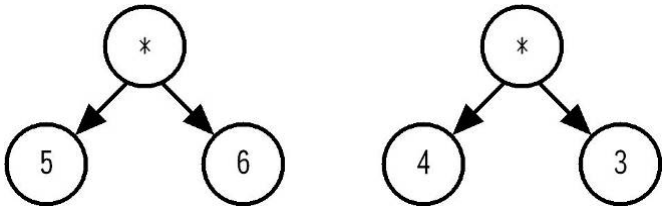
二元樹的運算式處理

- 從樹的觀念而言，樹可以處理各種階層關係的問題，例如：賽程表和家族族譜等，如果將資料建立成二元搜尋樹，就成為一種很好的資料搜尋方法。
- 運算式處理可以使用堆疊執行轉換和求值，現在我們可以改為二元樹來處理運算式，建立運算式二元樹。

56

二元樹的應用 - 運算式處理

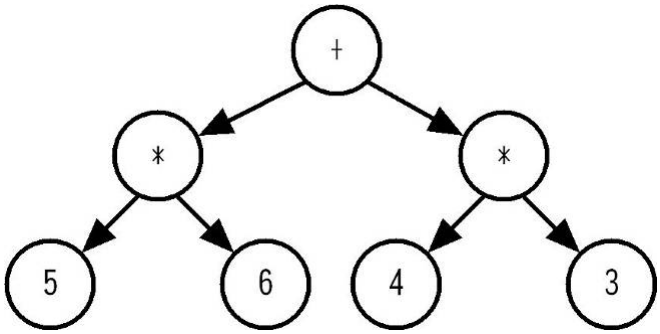
- 例如：將中序運算式轉換成二元樹，如下所示：
  5*6+4*3
- 上述中序運算式的運算元是二元樹的葉節點，運算子是非終端節點，因為考量運算子的優先順序，乘號大於加號，所以前後兩個乘號運算子先處理，可以建立成二棵二元樹，如下圖所示：



57

二元樹的應用 - 運算式處理

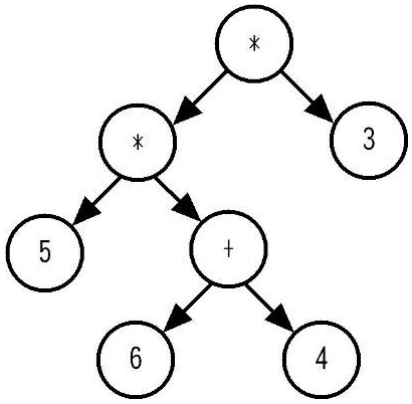
- 接著處理低優先順序的加號，就完成運算式二元樹，如下圖所示：



58

二元樹的應用 - 運算式處理

- 一棵沒有依據算子優先順序建立的運算式二元樹，如下圖所示：



59

二元樹的應用 - 運算式處理

- 運算式二元樹只需從葉節點開始計算各子節點的值，然後依序往上就可以計算出整棵運算式二元樹的值，如下所示：
  - 有優先順序的二元樹：42
    - 5*6 = 30
    - 4*3 = 12
    - 30+12 = 42
  - 不考慮優先順序的二元樹：150
    - 6+4 = 10
    - 5*10 = 50
    - 50 *3 = 150

60

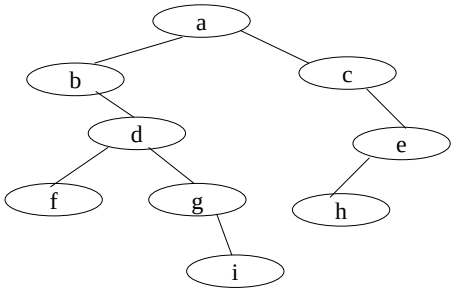
二元樹的應用 - 運算式處理

- 中序走訪運算式二元樹的結果如下：
  - **有優先順序的二元樹**：5*6+4*3
  - **不考慮優先順序的二元樹**：5*6+4*3
- 前序走訪運算式的結果如下：
  - **有優先順序的二元樹**：+*56*43
  - **不考慮優先順序的二元樹**：**5+643
- 後序走訪運算式二元樹的結果如下：
  - **有優先順序的二元樹**：56*43+
  - **不考慮優先順序的二元樹**：564+*3*

61

請寫出答案

1. 現有一個二元樹如下圖，並將之以陣列表示？其中節點f、i、h的儲存位置分別為何？並將之以結構陣列表示？並請寫出中、前、後序走訪結果？陣列為{a,b,c,d,e,f,g,h,i}
2. 樹根節點為何？節點b的兄弟節點為何？此樹高為何？樹葉節點為何？節點g的祖先節點為何？左右子樹為何？



62

請寫出答案

- 現有一個二元樹，請寫出下列問題：
  - 二元樹階層為i，最多有多少個節點？
  - 二元樹高度為k，最多有多少個節點？
  - 二元樹高度為5的完整二元樹最少有多少個節點？最多有多少個節點？
  - 請畫出高度為6的右歪斜樹？
  - 二元樹高度為k，最多有多少個樹葉節點？最多有多少個非樹葉節點？
  - 有n個節點的二元樹其高度最高為何？最低為何？

63

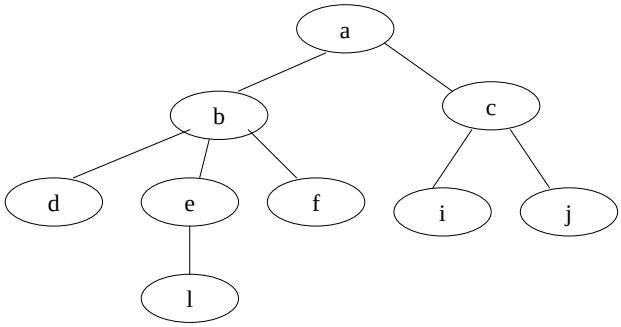
請寫出答案

- 現有一個二元樹之前序與中序走訪如下：前序：abcdefg，中序：dcbeafg，請畫出此二元樹？並請寫出後序結果？

64

請寫出答案

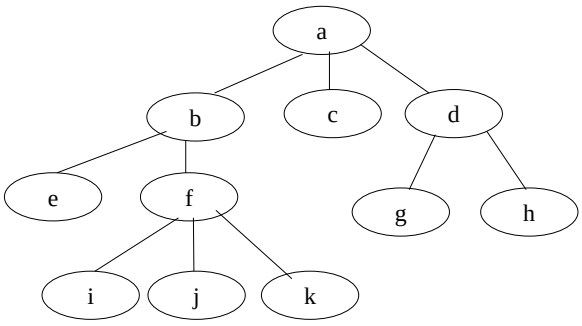
將下列一般樹轉為二元樹後並將之以陣列表示？
並將之以結構陣列表示？陣列為{a,b,c,d,e,f,i,j,l}
並請寫出中、前、後序走訪結果？



65

請寫出答案

將下列一般樹轉為二元樹後並將之以陣列表示？
並將之以結構陣列表示？陣列為{a,b,c,d,e,f,g,h,i,j,k}
並請寫出中、前、後序走訪結果？



66