陸、樹

一、樹 (Tree) 的簡介

1. 樹(Tree)是由一個或一個以上的節點所組成的有限集合
   ①樹根(root)：存在一個特定節點
   ②子樹(subtree)：其餘的節點可以分割成 n 大於等於 0 個彼此間沒有交集的(disjoint)集合$\boldsymbol{T_1}$、$\boldsymbol{T_2}$、⋯、$\boldsymbol{T_n}$，其中每一個集合也都是一棵樹
   ③節點(node)：代表資料項及連接至其他節點的分支

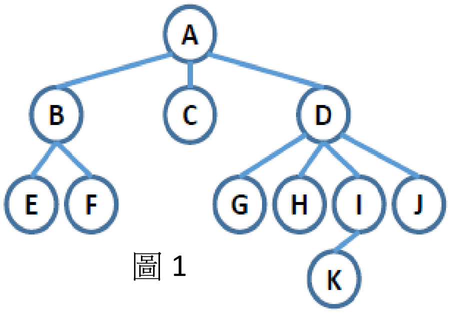


圖 1

2. 樹(Tree)的介紹：(如下表 1)

編號	名稱	解釋
1	**節點的分支度**	一個節點的子樹數目
2	**樹的分支度**	一棵樹中所有節點分支度的最大值
3	**樹葉節點**	分支度為 0 的節點(又稱終端(terminal)節點)
4	**非終端節點**	不是樹葉節點的節點
5	**節點的兒子與父親**	對於任一個節點 X，它的子樹的樹根，稱為 X 節點的兒子(Children)，反之 X 稱為其兒子的父親(Parent)
6	**兄弟節點**	擁有相同的父親的節點們
7	**節點的祖先**	一個節點的祖先(Ancestors)指的是這個節點通往樹根時所經過的所有節點
8	**節點的階層**	一棵樹裡所有節點階層的最大值為此樹的深度(Depth)或高度(Height)

(表 1)

※隨堂測驗

⊙請以表 1 各項目回答圖 2

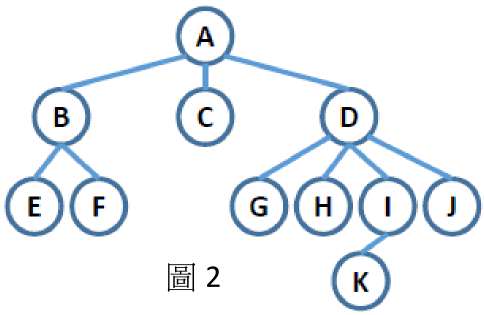


圖 2

補充：為什麼 mysql 的索引使用 B+樹而不是 B 樹呢？？

1. B+樹更適合外部儲存(一般指磁碟儲存)，由於內節點(非葉子節點)不儲存 data，所以一個節點可以儲存更多的內節點，每個節點能索引的範圍更大更精確。也就是說使用 B+樹單次磁碟 IO 的資訊量相比較 B 樹更大，IO 效率更高。
2. mysql 是關係型資料庫，經常會按照區間來訪問某個索引列，B+樹的葉子節點間按順序建立了鏈指標，加強了區間訪問性，所以 B+樹對索引列上的區間範圍查詢很友好。而 B 樹每個節點的 key 和 data 在一起，無法進行區間查詢。

二、樹 (Tree) 的串列表示法

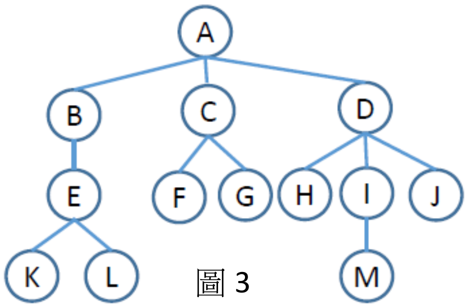


圖 3

1. 串列表示法：每一個記憶體節點包含了資料欄位與指向兒子節點的指標

   (A(B(E(K,L),F),C(G),D(H(M),I,J)))

2. 左子-右兄弟表示法(Left Child-Right Sibling Representation)

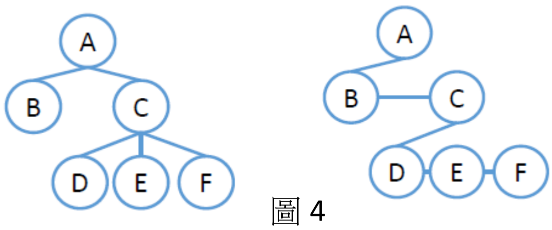


圖 4

3. 表示成一棵分支度為二的樹

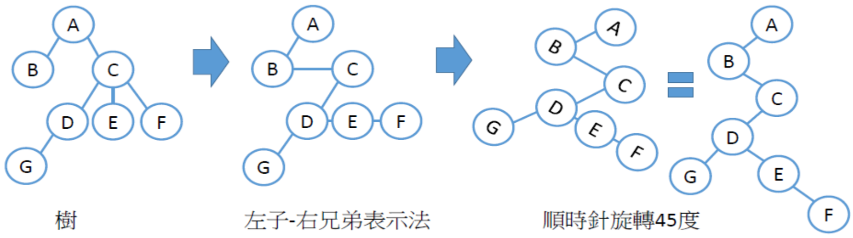


圖 5

※範例講解

⊙一個有 n 個節點的樹，若用串列表示法表示，請問：

❶總共有幾個鏈結欄位? ❷其中有多少個欄位是空的未用?

⊙若有一串列表示法如下，請依表示法畫出樹

(A(B(E,(F,G)),C(H,I(K,L),J),D(M,N(O(P)))))

柒、二元樹

一、二元樹(Binary Tree)簡介

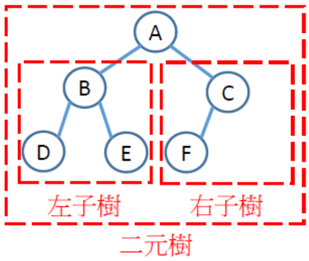


圖 6

1. 二元數(Binary Tree)：分支度為二的樹(樹中任一節點分支度不會超過 2)，可以不含任何節點，且有左子樹與右子樹的區別
2. 二元樹(Binary Tree)的特性：在二元樹的第 L 階(Level)上，最多可有的節點數目是2^{L-1}個
3. 二元樹(Binary Tree)的特性(續)：

①一棵樹高為 h 的二元樹，h≧1，最多有的節點數目S_h

$$S_h = 2^0+2^1+\ldots2^{h+1}= (2^h-2^0) \div (2-1) = 2^h-1$$

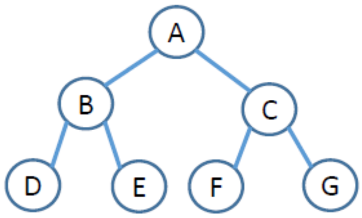


圖 7

②一棵二元樹，已知有 n 個節點，且分支度為 2 的節點數目是n_2；分支度為 1 的節點數目是n_1；分支度為 0 的節點數目是n_0；總邊數為 E

❶ $n = n_0 + n_1 + n_2$

❷ $n = E + 1$

❸ $E = n_1 + 2n_2$

❹ **由❸代入❷得** $n = n_1 + 2n_2 + 1$

❺ 由❹代入❶得 $n_0 + n_1 + n_2 = n_1 + 2n_2 + 1 \rightarrow n_0 = n_2 + 1$

❻ 由❶知 $n_1 = n - (n_0 + n_2)$

❼ 代入❺得 $n_1 = n - (n_2 + 1 + n_2) = n - 2n_2 - 1$

❽ 結論：樹葉節點的數量 n_0 等於分支度為 2 的節點數量n_2+1

③一棵二元樹，已知有 n 個節點，且分支度為 2 的節點數目是n_2；分支度為 0 的節點數目是 n_0

❶ $n \geqq 2n_2 + 1$

❷ $n \geqq 2n_0 - 1$

❸ 若等於發生，表示其為完全二元樹(Full binary tree)，又稱為完滿二元樹，因為沒有分支度為 1 的節點

4. 完全二元樹(Full Binary Tree)：

①一個高度為 h 且具有 n 個節點的二元樹稱為完全二元樹(Full binary tree)，**$n = 2^h-1$**

②從第 1 階的樹根開始，接著是第 2 階的節點，依此類推來編號樹裡的所有節點。位於同一階的節點則從左往右來編號

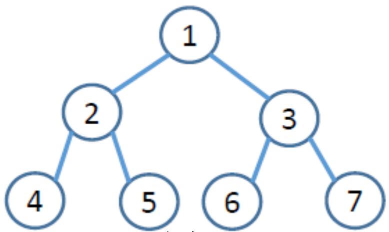


圖 8

5. 完整二元樹(Complete Binary Tree)：一個有 n 個節點且深度為 k 的二元樹是完整二元樹，若且為若，他的節點和一個深度為 k 的完全二元樹中，從編號 1 到 n 的節點一致
6. 完整二元樹(Complete Binary Tree)的特性：

①一棵完整二元樹，有 n 個節點且 n ≧ 0 ，樹高 h 且 h ≧ 0，n 和 h 為整數

②則

$2^{h}-1 \geq n > 2^{(h-1)}-1 \rightarrow 2^{h} \geq n+1 > 2^{(h-1)} \rightarrow h$ --❶

$h = \mathrm{ceil}\ (\log_2(n+1))$ --❷

③結論：

❶一棵完整二元樹，樹高 h；則總節點數量 n 滿足

❷一棵完整二元樹，有 n 個節點；則樹高 $h = \mathrm{ceil}\ (\log_2(n+1))$

7. 完整二元樹(Complete Binary Tree)的特性(續)：

①一個節點其在第 k 階，具有編號 i

②則其編號必滿足：

$2^{k-1}-1 < i \leq 2^{k}-1$

$2^{k-1} < i+1 \leq 2^{k}$

$k-1 < \log_2(i+1) \leq k$

③編號 i 的節點，在第 k 階：$k = \mathrm{ceil}\ (\log_2(i+1))$

④在第 k 階的節點 i

❶最小編號為2^{k-1}

❷最大編號為$2^{k}-1$

⑤父親與兒子編號

❶左子編號 L 為$2i$

❷右子編號 R 為$2i+1$

❸父親編號 M 為 $i \div 2$

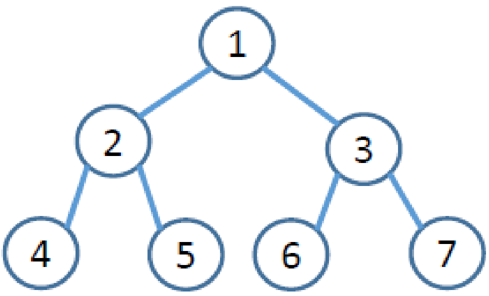


圖 9

8. 歪斜二元樹(Skewed Binary Tree)：當一個二元樹完全沒有右子樹或左子樹時，稱之左歪斜二元樹(Left-Skewed Binary Tree)或右歪斜二元樹(Right-Skewed Binary Tree)。

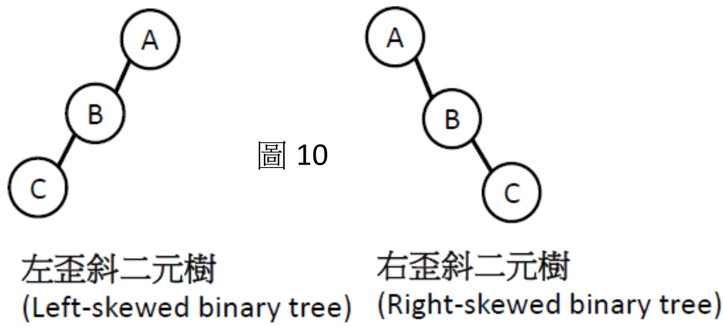


圖 10

9. 嚴格二元樹(Strictly Binary Tree)：若二元樹的每個非終端節點(樹葉節點)都有非空的左右子二元樹，則此二元樹稱為嚴格二元樹。(一棵沒有分支度為 1 的節點的二元樹)

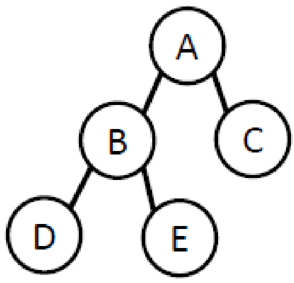


圖 11

- 若在完整二元樹內有個編號 401 的節點，請問
  ❶此節點在該樹的第幾階？
  ❷此節點的父節點的編號為何？
  ❸若此節點有左子，其左子的編號為何？
  ❹若此節點有右子，其右子的編號為何？

- 一棵完整二元樹，有 n 個節點且 n ≧ 0 ，樹高 h 且 h≧ 0 ，n 和 h 為整數。請問此樹的樹高 h 為何？

- 一棵二元樹，已知分支度為 2 的節點數目是n_2；請問此樹樹葉節點的節點數目是n_0是多少？

二、二元樹(Binary Tree)表示法

1. 陣列表示法
   ①適合完整二元樹
   ②索引值 k，放編號 k 的節點
   ③編號 i 的節點
     ❶在第 k 階，k = ceil $(\log_2(i+1))$
     ❷左子編號 L 為$2i$
     ❸右子編號 R 為$2i+1$
     ❹父親編號 M 為 i÷2

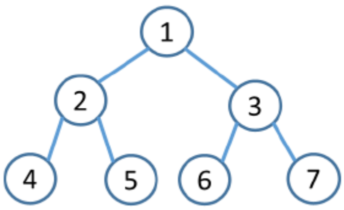


圖 12

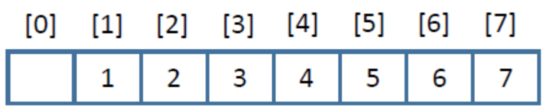


圖 13

2. 陣列表示法優缺點
   ①優點：❶可快速取的特定節點 k 的內容 ❷每個節點只需一個資料欄位
   ②缺點：若要更動節點於樹中所在的位置，例如刪除或插入運算，通常會牽涉到大量的資料搬移
3. 鏈結串列表示法
   ①每個節點，有三個欄位(左子(left-child)、資料(data)、右子(right-child))
   ②經改良後新增一個指向父節點的指標欄位

圖 13

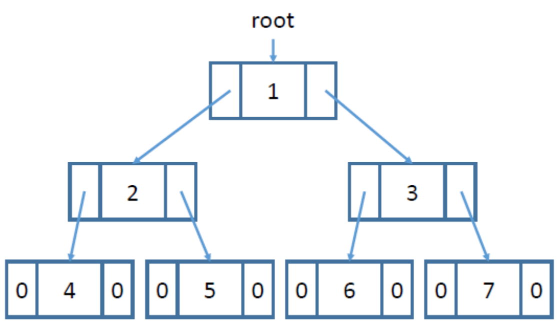


圖 14

4. 鏈節串列表示法優缺點

①優點：容易新增與刪除節點

②缺點：❶每個節點要至少三個欄位 ❷要取得特定鍵值之節點資料，需要從根節點開始尋找

三、二元樹(Binary Tree)走訪

1. 二元樹走訪有分為：中序走訪法（LVR)、前序走訪法（VLR)、後序走訪法（LRV)

2. 中序走訪法：

①往樹的左下方移動直到不能再移動為止

②拜訪這個節點

③接著往右移動一個節點再繼續

④如果沒辦法往右移動，那麼就往後移動一個節點(退到父節點)

```
void inorder(treePointerptr)
{ /* 中序走訪法*/
if( ptr){
inorder(ptr->leftChild);
printf("%d",ptr->data);
inorder(ptr->rightChild);
}
}
```

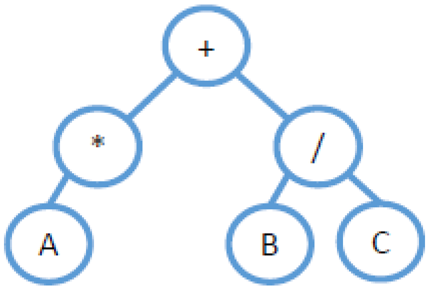


圖 15

※範例講解

⊙利用中序走訪，走訪圖 15 並列出順序

※學生練習

⊙利用中序走訪，走訪圖 16 並列出順序

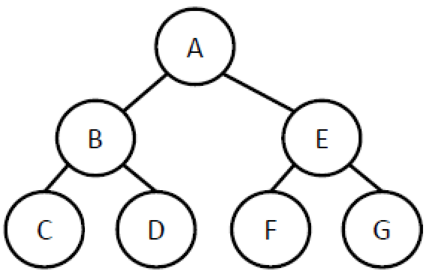


圖 16

3. 前序走訪法：追蹤順序為樹根->左子樹->右子樹

```
Procedure Preorder( T: TreePointer)
Begin
if ( T <> null ) then
{
```

```
print (T->Data) ;
Preorder( T->LChild) ;
Preorder( T->RChild) ;
}
End
```

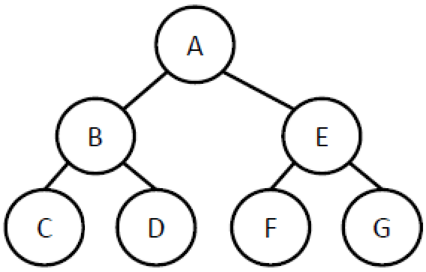


圖 17

※範例講解

⊙利用前序走訪，走訪圖 17 並列出順序

4. 後序走訪法：追蹤順序為左子樹->右子樹->樹根

```
Procedure Postorder( T: TreePointer)
Begin
if( T <> null ) then
{
Postorder( T->LChild) ;
Postorder( T->RChild) ;
Print( T->data ) ;
}
End
```

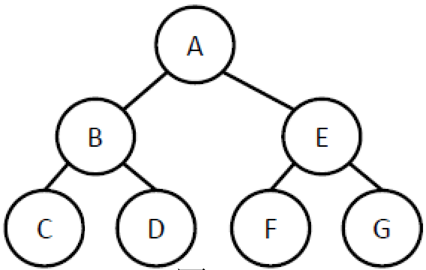


圖 18

※範例講解

⊙利用後序走訪，走訪圖 18 並列出順序

5. 決定唯一的二元樹：已知中序走訪的結果，再搭配後序走訪的結果或前序走訪的結果可決定出此二元樹。(表 2 以圖 18 走訪)

走訪名稱	特性	走訪結果
前序	二元樹的樹根必是走訪結果的第一個	A B C D E F G
中序	若知道樹根，則可以分出左子樹與右子樹的集合	C B D A F E G
後序	二元樹的樹根必是走訪結果的最後一個	C D B F G E A

(表 2)

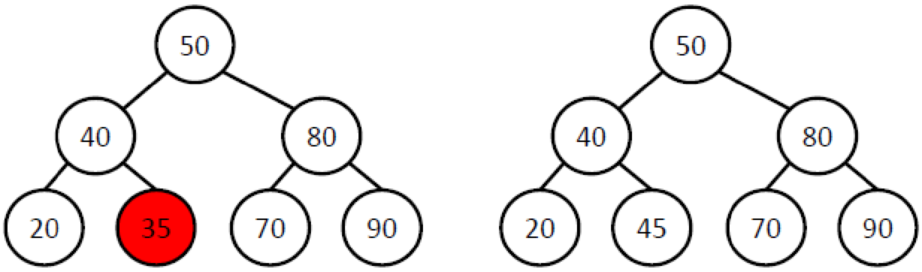
6. 二元搜尋樹(Binary Search Tree)：二元搜尋樹是一種二元樹，它可以為空，若不為空：

(一)二元搜尋樹須滿足以下條件(如圖 19)

①若左子樹非空，則左子樹的鍵值均須小於樹根的鍵值；

②若右子樹非空，則右子樹的鍵值均須大於樹根的鍵值

③左子樹與右子樹，也必須為二元搜尋樹



非二元搜尋樹　圖 19　二元搜尋樹

※範例講解

⊙利用中序走訪，走訪圖 19(右)並列出順序

(二)建立(加入)二元搜尋樹步驟

①取出一筆資料；加入二元搜尋樹內

②若樹不存在，輸入資料當此二元搜尋樹的樹根，返回

③否則輸入值與樹根比較；

❶若比樹根大，輸入資料加入左子樹這個二元搜尋樹；

❷否則，輸入資料加入右子樹這個二元搜尋樹；

※範例講解

⊙請依圖 20 將所有元素加入二元搜尋樹

圖 20

(三)刪除搜尋二元樹的節點步驟

①找出要刪除的節點

②若為樹葉節點，直接刪除，否則

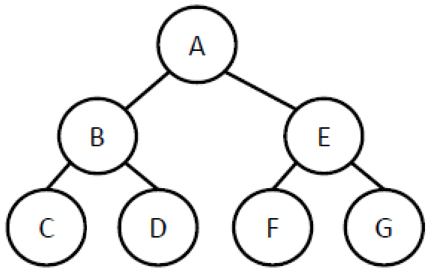
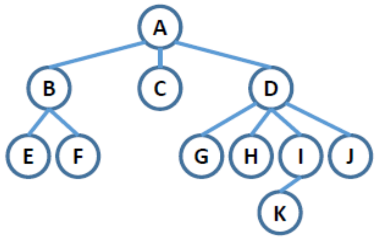
方法❶：以該節點為根的子二元搜尋樹的左子樹鍵值最大者，取代該節點

方法❷：以該節點為根的子二元搜尋樹的右子樹鍵值最小者，取代該節點

※範例講解

⊙請依蘇教授所出題目，將指定結點刪除後重新組合二元搜尋樹

模擬考練習

答案	題號	題目內容
	1	下列有關樹的敘述何者錯誤? A. 嚴格二元樹分支度恆為 2 B. 二元樹以陣列表示法優點為每個節點都需要三個資料欄位 C. 在完整二元樹內有個編號 1000 的節點，此節點的父親節點為 500 D. 樹葉節點與非終端節點分別是樹的末端與非末端
	2	如右圖，以二元樹走訪何者為是? A. 中序走訪: C D B A F G E B. 前序走訪: A B C D E F G C. 後序走訪: C D B A G E F D. 以上皆是 
	3	有關右圖的敘述何者錯誤?(複選題) A. A 的節點分支度為 3 B. 樹的節點為 3 C. 陣列表示法為 (A(B(E(K,L),F),C(G),D(H(M),I,J))) D. 此樹的階層為 4 
	4	請利用前序、中序、後序走訪右圖