Chapter 6
樹狀結構

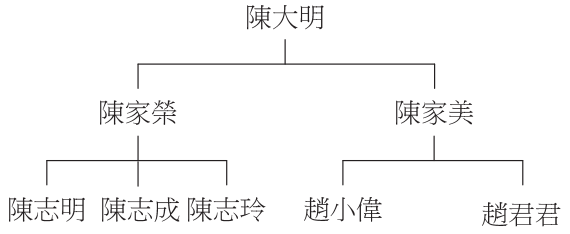
6-1 認識樹

樹（tree）是由一個或多個**節點**（node）所組成的有限集合，其特質如下：

- 有一個特殊節點，稱為**樹根**（root）。
- 其餘節點可以分為 n 個互斥集合 T_1、T_2、…、T_n $(n \geq 0)$，而且每個集合也都是一棵樹，稱為樹根的**子樹**（subtree）。

樹狀結構適合用來存放具有分支關係的資料，例如國人所熟悉的族譜、機關企業的組織表、運動項目的賽程表、事物的歸類分項等，以圖 6.1(a) 的族譜為例，陳大明的子女有陳家榮和陳家美，陳家榮的子女有陳志明、陳志成和陳志玲，而圖 6.1(b) 則是電腦軟體的歸類分項。

(a)



(b)

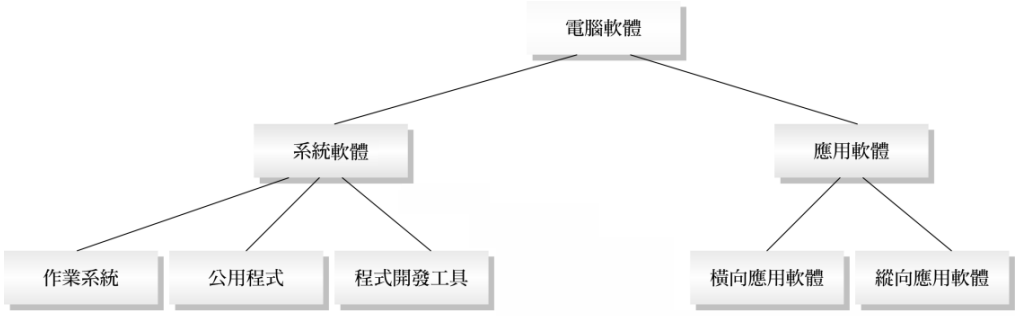


圖 6.1(a) 族譜 (b) 電腦軟體的歸類分項

6-1-1 樹的相關名詞

我們以圖 6.2 為例，說明樹的相關名詞：

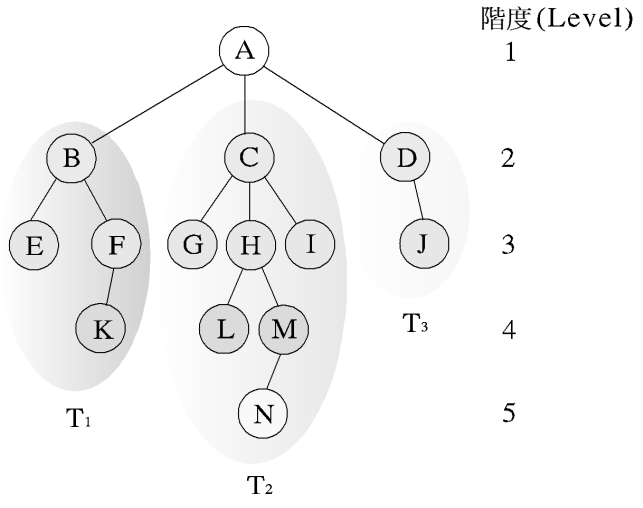


圖 6.2 樹

- 樹的每個圓圈代表一個**節點**（node)，裡面可以用來存放資料，而節點與節點之間的直線則稱為**邊**（edge)，例如圖 6.2 的樹有 14 個節點和 13 個邊，節點裡面的資料是英文字母 A、B、…、N。
- 節點有幾棵子樹稱為**節點的分支度**（degree of a node)，例如 A 的分支度為 3、B 的分支度為 2。
- 在樹的所有節點中，分支度最大者即為**樹的分支度**（degree of a tree)，例如 A 和 C 的分支度最大，均為 3，故樹的分支度為 3。
- 分支度為零的節點稱為**終端節點**（terminal node）或**樹葉**（leaf)，例如 E、K、G、L、N、I、J 均為終端節點，也就是沒有子樹的節點，而終端節點以外的節點則稱為**非終端節點**（nonterminal node)，例如 A、B、C、D、F、H、M 均為非終端節點。

- 某個節點的所有子樹的樹根均為其**子節點**（children），例如 G、H、I 為 C 的子節點；相反的，若某甲為某乙的子節點，那麼某乙為某甲的**父節點**（parent），例如 C 為 G、H、I 的父節點。
- 父節點相同的節點稱為**兄弟**（sibling），例如 G、H、I 為兄弟。
- 從樹根往下到某個節點之前所經過的所有節點均為該節點的**祖先**（ancestor），而該節點則為**子孫**（descendant），例如 A、B 均為 E 的祖先，而 E 為 A、B 的子孫。
- 從樹根開始，其**階度**（level）為 1，每往下一層的節點，其階度就遞增 1，例如 A 的階度為 1，B、C、D 的階度為 2。
- 一棵樹的最大階度稱為**高度**（height）或**深度**（depth），例如圖 6.2 的樹，其高度或深度為 5。
- **樹林**（forest）是由 n 棵互斥樹所組成的集合（$n \geq 0$），事實上，樹林和樹類似，只要把樹的樹根去掉，剩下的便是樹林，例如圖 6.2 的樹若去掉 A，就會變成一個包含 T_1、T_2、T_3 三棵樹的樹林，如圖 6.3，其樹根分別為 B、C、D。

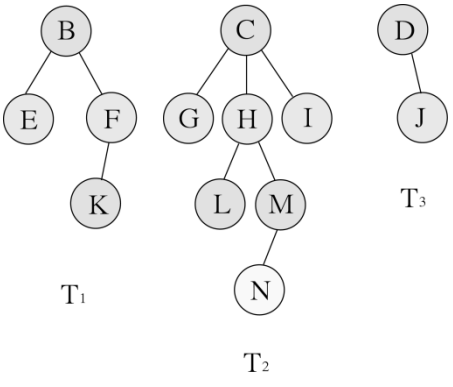


圖 6.3 樹林

範例 6.1：[V = E + 1] 假設樹的節點個數為 V、邊數為 E，試證明 V = E + 1，即節點個數等於邊數加 1。

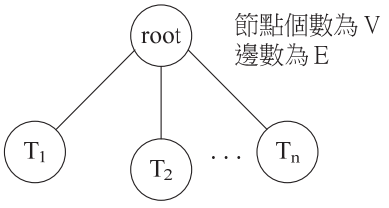
解答：我們使用數學歸納法來加以證明：

1. 當 V = 1 時，E = 0，故 V = E + 1 成立。

2. 假設當 $1 \le V \le k$ 時，V = E + 1 成立。

3. 證明當 V = k + 1 時，V = E + 1 亦成立。

假設樹的樹根有 n 棵子樹 T_1、T_2、…、T_n，其節點個數為 V_1、V_2、…、V_n，邊數為 E_1、E_2、…、E_n，$V_1 + V_2 + \cdots + V_n = k$，加上樹根後共有 k + 1 個節點，每棵子樹均藉由一個邊和樹根相連，所以將這 n 棵子樹連接到樹根共需 n 個邊，如下圖：



已知 $V_1 = E_1 + 1$ ..... ①

$V_2 = E_2 + 1$ ..... ②

…

$V_n = E_n + 1$ ..... ⓝ

而 $V = V_1 + V_2 + \cdots + V_n + 1$ ..... ⓐ

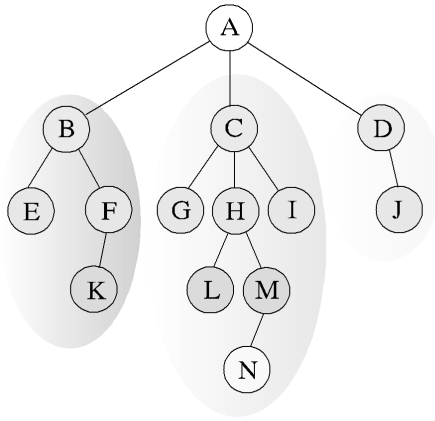
$E = E_1 + E_2 + \cdots + E_n + n$ ..... ⓑ

將① ~ ⓝ代入ⓐ，得到 $V = E_1 + E_2 + \cdots + E_n + n + 1$ ..... ⓒ

將ⓑ代入ⓒ，得到 V = E + 1，故得證。

6-1-2 樹的表示方式

我們可以使用串列表示樹，例如下圖的樹可以表示成 (A(B(E, F(K)), C(G, H(L, M(N)), I), D(J)))，其中 D 有一個子節點 J，M 有一個子節點 N，H 有兩個子節點 L、M，C 有三個子節點 G、H、I，…，依此類推，直到 A 有三個子節點 B、C、D。



至於每個節點在記憶體的儲存方式，我們可以使用如下結構，其中 data 欄位用來存放節點的資料，link1、link2、…、link*n* 等欄位用來存放節點的鏈結，這是指向子節點的指標。

data	link1	link2	…	link*n*

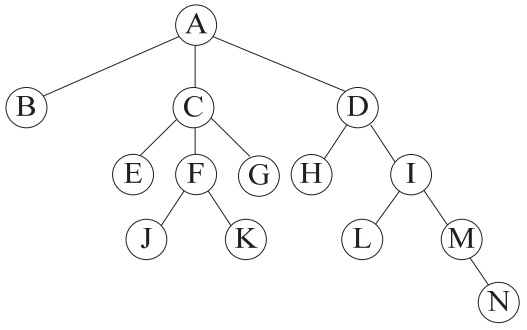
顯然鏈結欄位的個數取決於樹的分支度，一個節點個數為 V、分支度為 n 的樹將有 n×V 個鏈結欄位，但事實上，有使用到的鏈結欄位卻只有 V - 1 個 (即等於邊數)，這表示有 n×V - (V - 1) 個鏈結欄位是空的，相當沒有效率。

為了節省記憶體，我們通常會將一般樹轉換為**二元樹** (binary tree)，以便將節點的欄位限制為三個，如下，其中 data 欄位用來存放節點的資料，lchild 欄位用來存放節點的左鏈結 (指向左子樹)，lchild 欄位用來存放節點的右鏈結 (指向右子樹)，下一節有進一步的討論。

lchild	data	rchild

隨堂練習

1. 針對下圖的樹回答下列問題：



(1) 該樹的節點個數為何？邊數為何？這是否符合範例 6.1 的 V = E + 1？

(2) 該樹的終端節點個數為何？非終端節點個數為何？兩者之和是否等於該樹的節點個數？

(3) 該樹的高度為何？分支度為何？

(4) 節點 L 的父節點為何？祖先為何？

(5) 節點 F 的階度為何？分支度為何？

(6) 節點 B 的兄弟為何？

(7) 使用串列表示該樹。

2. 假設使用串列表示某棵樹為 (A(B(D, E(G, H)), C(F(I)))，請描繪該樹。

3. 假設某棵樹的節點個數為 15、分支度為 5，試問，當我們使用如下結構存放節點時，總共需要幾個鏈結欄位？其中有幾個鏈結欄位是空的？

data	link1	link2	⋯	link*n*

6-2 二元樹

二元樹（binary tree）是每個節點最多有兩個子節點的樹，節點的左邊稱為**左子樹**（left child），節點的右邊稱為**右子樹**（right child），圖 6.4 即為一例。

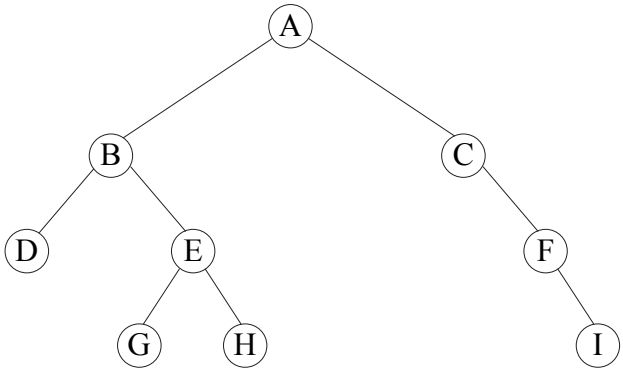


圖 6.4 二元樹

二元樹和樹的差別如下：

- 二元樹可以是空集合，而樹必須至少有一個樹根，不可以是空集合。
- 二元樹的節點分支度為 $0 \le d \le 2$，而樹的非終端節點分支度為 $d \neq 0$。
- 二元樹會以左右子樹或左右節點來區分順序，而樹無順序之分。

以圖 6.5(a)、(b) 為例，這雖然是兩棵相同的樹，卻是兩棵不同的二元樹，因為二元樹會以左右子樹或左右節點來區分順序。

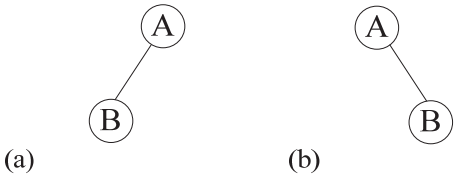


圖 6.5 兩棵不同的二元樹

範例 6.2：證明二元樹第 i 階的最多節點個數為 2^{i-1}，$i \geq 1$。

解答：我們使用數學歸納法來加以證明：

1. 當 $i = 1$ 時，第 1 階最多只有樹根一個節點，故 $2^{i-1} = 2^{1-1} = 2^0 = 1$ 成立。
2. 假設當 $1 \leq i < k$ 時，第 i 階的最多節點個數為 2^{i-1} 成立，故第 k - 1 階的最多節點個數為 2^{k-2}。
3. 證明當 $i = k$ 時，第 i 階的最多節點個數為 2^{i-1} 亦成立。由於第 k - 1 階的最多節點個數為 2^{k-2}，而二元樹的每個節點最多有兩個子節點，故第 k 階的最多節點個數為 $2^{k-2} \times 2 = 2^{k-1}$。

範例 6.3：證明高度為 h 之二元樹的最多節點個數為 $2^h - 1$，$h \geq 1$。

解答：由於二元樹第 i 階的最多節點個數為 2^{i-1}，因此，對高度為 h 的二元樹來說，全部存滿的話，總共有 $2^0 + 2^1 + \cdots + 2^{h-1} = 2^h - 1$ 個節點。

例如圖 6.6 是一個高度為 4 且全部存滿的二元樹，它的節點個數為 $2^4 - 1 = 15$，編號為 0 ~ 14，而且第 1、2、3、4 階的節點個數分別為 2^{1-1}、2^{2-1}、2^{3-1}、2^{4-1}。事實上，這種全部存滿的二元樹就叫做**完滿二元樹**（full binary tree）。

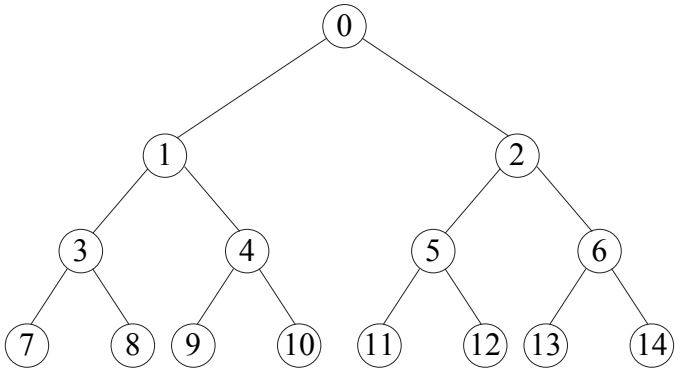


圖 6.6 高度為 4 的完滿二元樹

範例 6.4：證明高度為 h 之二元樹的最少節點個數為 h，$h \geq 1$。

解答：當二元樹的每一階都各只有一個節點時，其節點個數最少，故高度為 h 之二元樹的最少節點個數為 h，例如圖 6.7 是兩個高度為 4 且節點個數最少的二元樹，它的最少節點個數等於其高度 4。事實上，這種向左或向右傾斜的二元樹就叫做**傾斜二元樹**（skewed binary tree）。

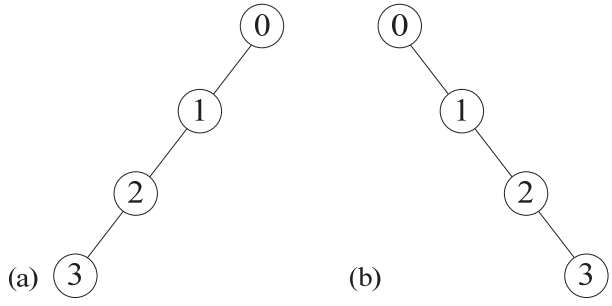


圖 6.7 高度為 4 的傾斜二元樹

範例 6.5：假設有一個非空的二元樹 T，其分支度為 0 的節點個數為 n_0，其分支度為 2 的節點個數為 n_2，試證明 $n_0 = n_2 + 1$ (例如圖 6.6 的 n_0、n_2 為 8、7，圖 6.7 的 n_0、n_2 為 1、0，兩者均符合 $n_0 = n_2 + 1$)。

解答：首先，假設二元樹 T 的節點個數為 V、邊數為 E、分支度為 1 的節點個數為 n_1，則：

$V = E + 1$ ····· ①

$V = n_0 + n_1 + n_2$ ····· ②

接著，分支度為 1 的節點有一個邊，而分支度為 2 的節點有兩個邊，則：

$E = n_1 + 2n_2$ ····· ③

繼續，將③代入①，則：

$V = n_1 + 2n_2 + 1$ ····· ④

最後，將②代入④，則：

$n_0 = n_2 + 1$，故得證。

6-2-1 完滿二元樹 V.S. 完整二元樹

完滿二元樹（full binary tree）指的是高度為 h 且節點個數為 $2^h - 1$ 的二元樹，也就是全部存滿的二元樹，例如圖 6.8(a) 是高度為 3 且節點個數為 $2^3 - 1$ (7) 的完滿二元樹，編號為 0 ~ 6，而圖 6.8(b) 是高度為 4 且節點個數為 $2^4 - 1$ (15) 的完滿二元樹，編號為 0 ~ 14。

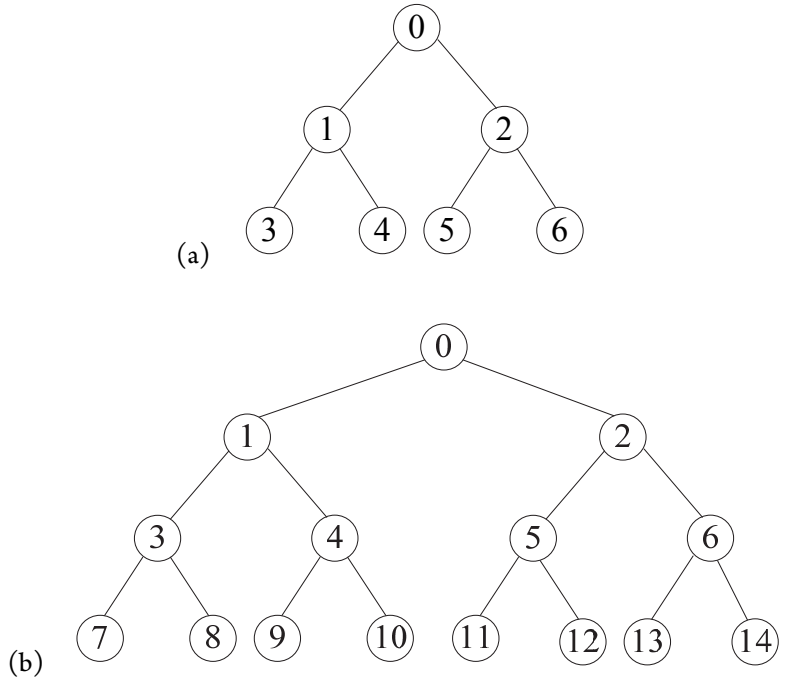


圖 6.8(a) 高度為 3 的完滿二元樹 (b) 高度為 4 的完滿二元樹

完整二元樹（complete binary tree）指的是高度為 h、節點個數為 n 且節點順序對應至高度為 h 之完滿二元樹的節點編號 0 ~ n - 1，例如圖 6.9(a) 是高度為 3、節點個數為 6 的完整二元樹，其節點順序對應至高度為 3 之完滿二元樹的節點編號 0 ~ 5，而圖 6.9(b) 是高度為 4、節點個數為 12 的完整二元樹，其節點順序對應至高度為 4 之完滿二元樹的節點編號 0 ~ 11。

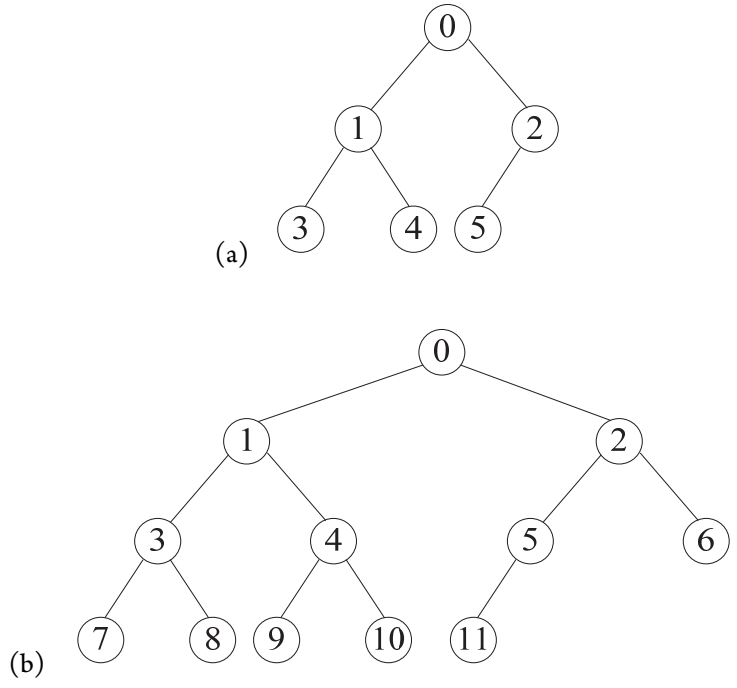


圖 6.9(a) 高度為 3 的完整二元樹 (b) 高度為 4 的完整二元樹

假設完整二元樹的節點個數為 n 且節點依照 0 ~ n - 1 的順序編號，則其任意節點 i ($0 \le i \le n - 1$) 具有下列特質：

- 若 $i = 0$，表示節點 i 為樹根，沒有父節點，否則節點 i 的父節點為 $\lfloor (i - 1)/2 \rfloor$ (即小於等於 $(i - 1)/2$ 的最大正整數)，例如節點 5 的父節點為 $\lfloor (5 - 1)/2 \rfloor$ (即節點 2)。
- 若 $2i + 1 \ge n$，表示節點 i 沒有左子節點，否則節點 i 的左子節點為 $2i + 1$。
- 若 $2i + 2 \ge n$，表示節點 i 沒有右子節點，否則節點 i 的右子節點為 $2i + 2$。

範例 6.6：證明節點個數為 n 之完整二元樹的高度為 $\lfloor \log_2 n \rfloor + 1$。

解答：假設完整二元樹的高度為 h，已知完整二元樹的最少節點個數為高度為 h - 1 之完滿二元樹的節點個數加 1，即 $2^{h-1} - 1 + 1 = 2^{h-1}$ (圖 6.10(a))，最多節點個數為高度為 h 之完滿二元樹的節點個數 $2^h - 1$ (圖 6.10(b))，則：

$$2^{h-1} \le n \le 2^h - 1$$
$$\Rightarrow 2^{h-1} \le n < 2^h$$
$$\Rightarrow \log_2(2^{h-1}) \le \log_2(n) < \log_2(2^h)$$
$$\Rightarrow h - 1 \le \log_2(n) < h$$
$$\Rightarrow \lfloor \log_2 n \rfloor = h - 1$$
$$\Rightarrow h = \lfloor \log_2 n \rfloor + 1$$

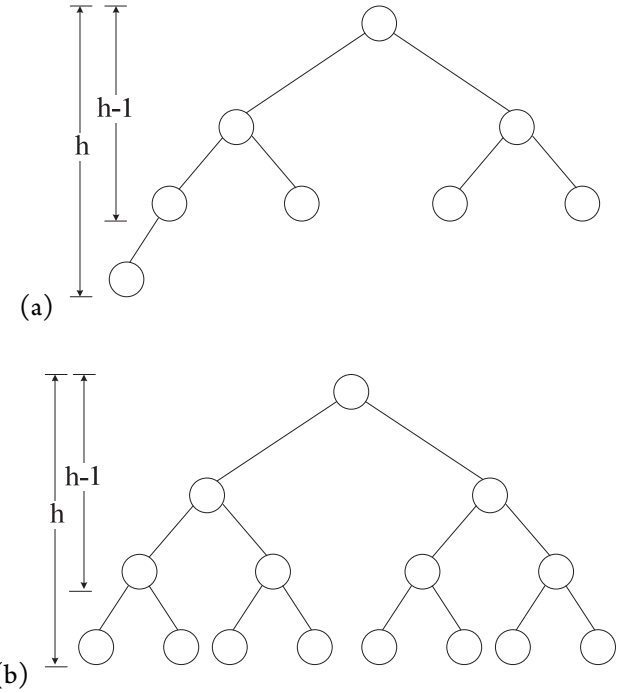
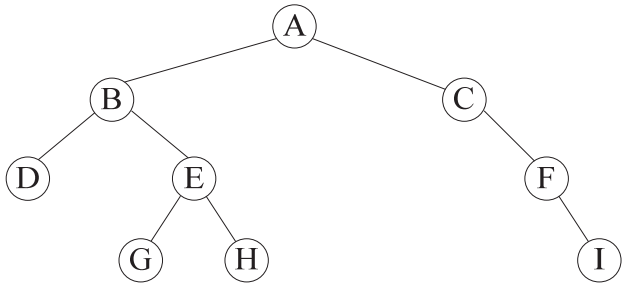


圖 6.10(a) 節點最少的情況 (b) 節點最多的情況

6-2-2 二元樹的表示方式

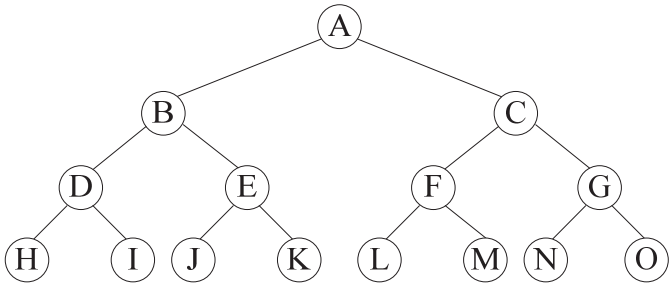
使用陣列實作二元樹

我們可以使用陣列存放二元樹，然後依照二元樹的高度由上到下、由左到右，第一個位置存放第一個階度的節點 (樹根)，第二個位置存放第二個階度的左節點，第三個位置存放第二個階度的右節點，第四 ~ 七個位置存放第三個階度由左到右的節點，…，依此類推，下圖是一個例子，其中 -- 表示缺的節點。



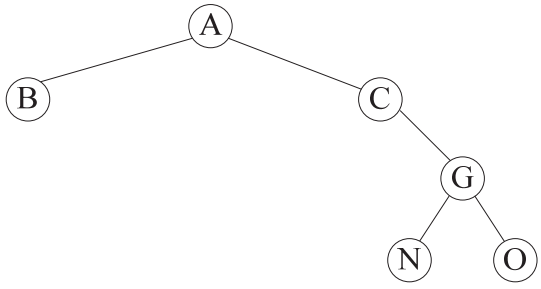
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
A	B	C	D	E	--	F	--	--	G	H	--	--	--	I

當二元樹呈現完整或左右平衡 (樹根的兩個子樹高度相同) 時，這種方式就比較不會浪費記憶體，下圖是一個例子。



[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

相反的，當二元樹呈現稀疏或左右不平衡時，這種方式就比較浪費記憶體，下圖是一個例子。



[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
A	B	C	--	--	--	G	--	--	--	--	--	--	N	O

使用陣列存放二元樹的優點如下：

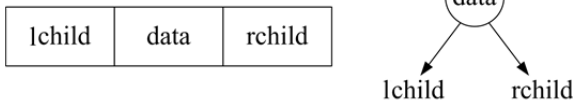
- 容易實作。
- 可以快速找出任意節點的父節點、左右子節點及兄弟節點存放在哪個位置，以上圖為例，已知節點 G 在編號 6 的位置，則其父節點在編號 $\lfloor(6-1)\rfloor/2=2$ 的位置 (即節點 C)，左子節點在編號 $2\times6+1=13$ 的位置 (即節點 N)，右子節點在編號 $2\times6+2=14$ 的位置 (即節點 O)，而兄弟節點的位置則取決於節點的編號為奇數或偶數，當節點的編號為奇數時，其兄弟節點是在加 1 的位置，當節點的編號為偶數時，其兄弟節點是在減 1 的位置。

使用陣列存放二元樹的缺點如下：

- 可能會浪費記憶體，尤其是當二元樹呈現稀疏或左右不平衡時。
- 增加或刪除節點時可能需要搬移多個節點。

使用鏈結串列實作二元樹

我們也可以使用鏈結串列存放二元樹，此時，每個節點的結構如下，其中 data 欄位用來存放節點的資料，lchild 欄位用來存放節點的左鏈結 (指向左子樹)，rchild 欄位用來存放節點的右鏈結 (指向右子樹)。




```
/*宣告 tree_node 是二元樹的節點*/
typedef struct node{
   struct node *lchild;          /*節點的左鏈結欄位*/
   char data;                    /*節點的資料欄位*/
   struct node *rchild;          /*節點的右鏈結欄位*/
}tree_node;

/*宣告 tree_pointer 是指向節點的指標*/
typedef tree_node *tree_pointer;
```

有了前述的節點結構後，我們可以使用鏈結串列將圖 6.11 的二元樹表示成如圖 6.12。

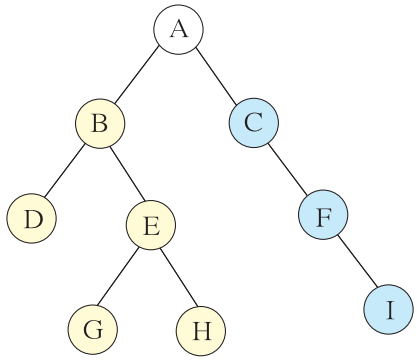


圖 6.11 二元樹

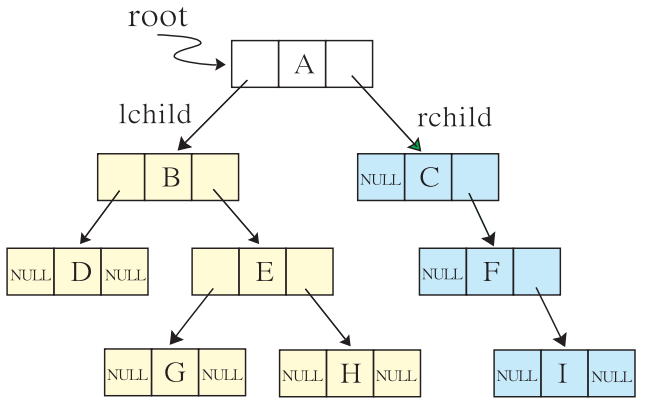


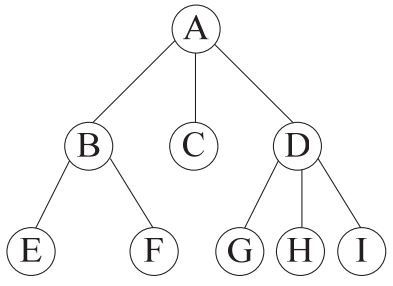
圖 6.12 使用鏈結串列表示圖 6.11 的二元樹

6-2-3 將樹轉換為二元樹

我們可以將樹轉換為二元樹，其步驟如下：

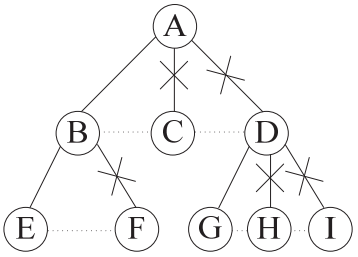
1. 將節點的兄弟節點連接在一起。
2. 除了最左邊子節點的邊予以保留之外，其它子節點的邊均刪除。
3. 順時針旋轉 45 度。

範例 6.7：將下面的樹轉換為二元樹。

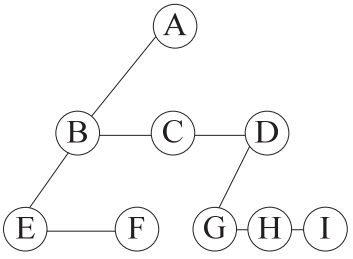


解答：

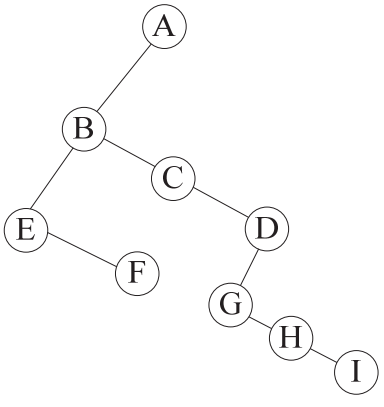
1. 將節點的兄弟節點連接在一起。



2. 除了最左邊子節點的邊予以保留之外，其它子節點的邊均刪除。

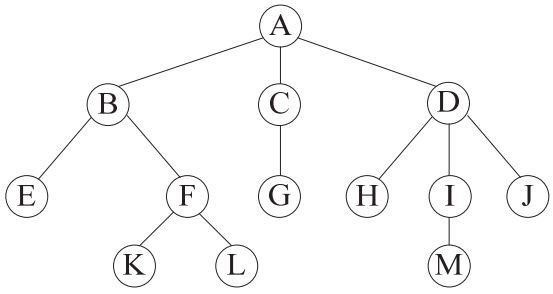


3. 順時針旋轉 45 度。



隨堂練習

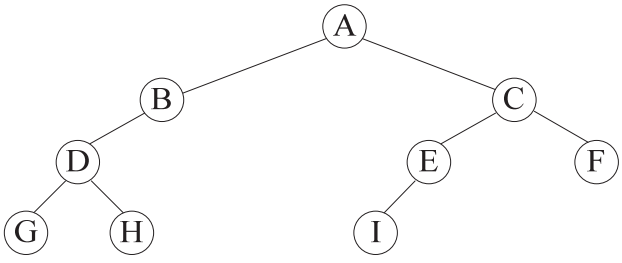
1. 下圖是二元樹嗎？若不是，其理由何在？請試著將它轉換成二元樹。



2. 節點個數為 31 及 32 之完整二元樹的高度分別為何？

3. 假設在一棵二元樹中，分支度為 0 的節點有 40 個，那麼分支度為 2 的節點有幾個？

4. 針對下圖的二元樹回答問題：



(1) 該樹有幾個分支度為 0 的節點？該樹有幾個分支度為 2 的節點？這是否符合範例 6.5 所證明的 $n_0 = n_2 + 1$？

(2) 畫出使用陣列存放該樹的結果。

(3) 畫出使用鏈結串列存放該樹的結果。

6-3 二元樹的運算

在決定如何存放二元樹後，我們還要提供相關運算的定義及函數，原則上，一個完整的二元樹資料結構應該要提供新增節點、刪除節點、走訪等運算的定義及函數。

6-3-1 走訪二元樹

走訪（traversal）指的是將樹狀結構的每個節點都拜訪一次，而且僅限一次，至於走訪的方式則有下列三種：

- **中序走訪**（inorder traversal）：若樹根不是空節點，則走訪步驟如下：
  1. 以中序走訪的方式拜訪左子樹。
  2. 拜訪樹根。
  3. 以中序走訪的方式拜訪右子樹。
- **前序走訪**（preorder traversal）：若樹根不是空節點，則走訪步驟如下：
  1. 拜訪樹根。
  2. 以前序走訪的方式拜訪左子樹。
  3. 以前序走訪的方式拜訪右子樹。
- **後序走訪**（postorder traversal）：若樹根不是空節點，則走訪步驟如下：
  1. 以後序走訪的方式拜訪左子樹。
  2. 以後序走訪的方式拜訪右子樹。
  3. 拜訪樹根。

以下圖的二元樹為例，其走訪結果如下：

學習評量

一、選擇題

( )1. 下列哪種資料不適合使用樹狀結構來存放？

A. 族譜　　B. 機關組織表

C. 遞迴函數　　D. 運動賽程表

( )2. 在二元樹中，第 i 階的最多節點個數為何？

A. i　　B. 2^{i-1}

C. $\log_2 n + 1$　　D. $2^i - 1$

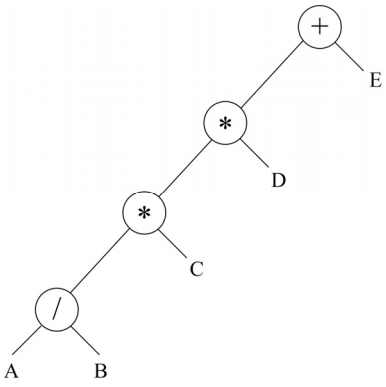
( )3. 針對下圖的運算式樹，下列敘述何者錯誤？

A. 前序走訪結果為 +*E*D/CAB

B. 前序走訪結果為 +**/ABCDE

C. 中序走訪結果為 A/B*C*D+E

D. 後序走訪結果為 AB/C*D*E+



( )4. 一個高度為 5 且全部存滿的二元樹，它的節點個數為何？

A. 31　　B. 15

C. 16　　D. 63

( )5. 某二元樹的每個節點各自儲存一個英文字母，其後序走訪為DBEFCA、中序走訪為 DBAECF，則其前序走訪為何？

A. ABDCEF　　B. ADBECF

C. ADBCEF　　D. ADBECF

( )6. 假設二元樹有 n 個節點，試問，其最大高度為何？(假設當此樹只有一個節點時，其高度為 1)

A. 2^{n-1}　　B. n

C. $2^n - 1$　　D. $\log_2 n$

( )7. 下列關於樹的敘述何者錯誤？

A. 假設樹的節點個數為 V、邊數為 E，則 V = E + 2

B. 終端節點指的是沒有子樹的節點

C. 節點的分支度指的是節點有幾棵子樹

D. 父節點相同的節點稱為兄弟

( )8. 下列關於二元樹的敘述何者錯誤？

A. 二元樹可以是空集合

B. 高度為 h 之二元樹的最少節點個數為 h

C. 高度為 h 之二元樹的最多節點個數為 $2^h - 1$

D. 二元樹的左右子樹並無順序之分

( )9. 下列關於二元搜尋樹的敘述何者正確？

A. 左子樹的鍵值必須大於其樹根的鍵值

B. 右子樹的鍵值必須小於其樹根的鍵值

C. 每個節點包含唯一的鍵值 (key)

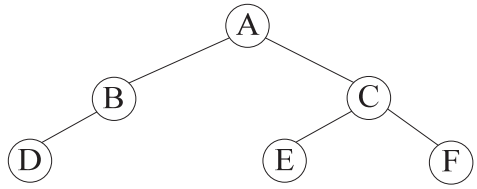
D. 二元搜尋樹的前序走訪結果剛好會使得資料由小到大排序

( )10.中序運算式(1+3)*5 轉換成前序運算式的結果為何？

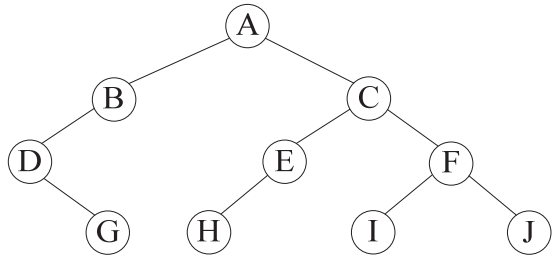
A. 1+35*　B. 1+3*7　C. *+135　D. +*135

二、練習題

1. 簡單說明何謂樹？請您想想看，生活中有哪些例子屬於樹的應用呢？
2. 以鏈結串列表示下列二元樹。



3. 我們可以從前序走訪結果或後序走訪結果決定唯一的二元樹嗎？
4. 寫出下列二元樹的中序、前序與後序走訪結果。



5. 假設運算式的中序表示法為 a / b ** c * d - e，試將它建構為運算式樹。
6. 假設二元樹的中序走訪結果為 AIBHCGDFE，後序走訪結果為 ABICHDGEF，試問，其前序走訪結果為何？
7. 假設數字串列為 (14, 15, 5, 9, 8, 19, 2, 6, 16, 4, 20, 17, 10, 13, 7)，請回答下列問題：

   (1) 將該數字串列建構為二元搜尋樹。

   (2) 寫出該二元搜尋樹的後序走訪結果。

   (3) 寫出該二元搜尋樹包含幾個樹葉與高度。
8. 假設二元樹的前序走訪結果為 ABCDE，中序走訪結果為 CBDAE，試據此推算出該二元樹。