tree

Tree

4.1 樹的基本概念

一、什麼是樹？

看到標題你可能會這麼問：電腦裡頭也有樹？原來電腦也可以種樹嗎？先別著急，這邊說的樹不是我們所知道大自然的樹，這裡所指的樹只是一種資料結構。大自然的樹是樹根在下，由下往上長出樹葉 （圖 4-1）；而資料結構的樹則是樹根在上，由上往下長出樹葉，此資料結構稱為**樹狀結構** （圖 4-2）。

圖 4-1-1 大自然的樹

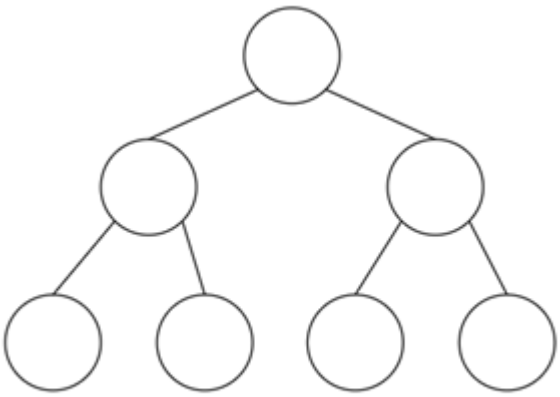


圖 4-1-2 樹狀結構

樹狀結構簡稱為**樹**，它是一種**階層式**的資料結構。日常生活中有許多資料可以用樹狀結構表示，例如：家庭圖（圖 4-2-1）、決策模型（圖 4-2-2）等，在 4.4 章會陸續介紹。

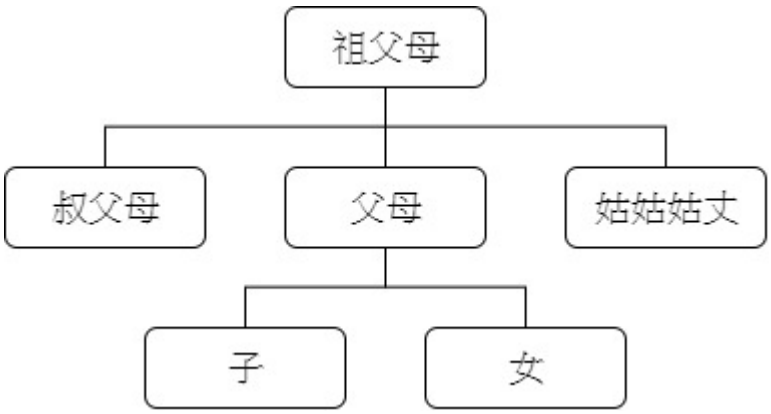


圖 4-2-1 家庭圖

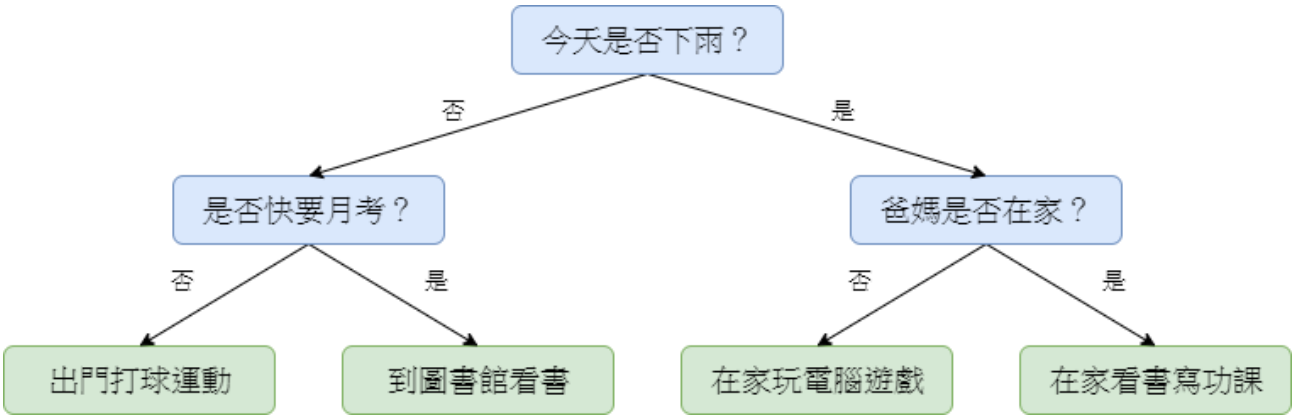


圖 4-2-2 決策模型

二、樹的基本定義與特點

在資料結構上，樹是由一個或多個節點所組成，它具有以下的特點：

1. 每一個節點只有有限個子節點或者沒有子節點。
2. 沒有父節點的節點稱為**根節點**。
3. 每一個棵樹一定有一個唯一的根節點。
4. 除了根節點之外，可以分成多棵互不相交的**子樹，**每一棵子樹也是一個樹。

樹的任兩個節點可相通，但只會有一條路徑，不會出現**循環**的迴路。生活中許多資料都是以**一對多**的形式呈現，一個班級有 25 個人、一個遊戲角色有好幾個基本屬性，這些資料都適合使用樹來做儲存與處裡。

三、樹的基本術語

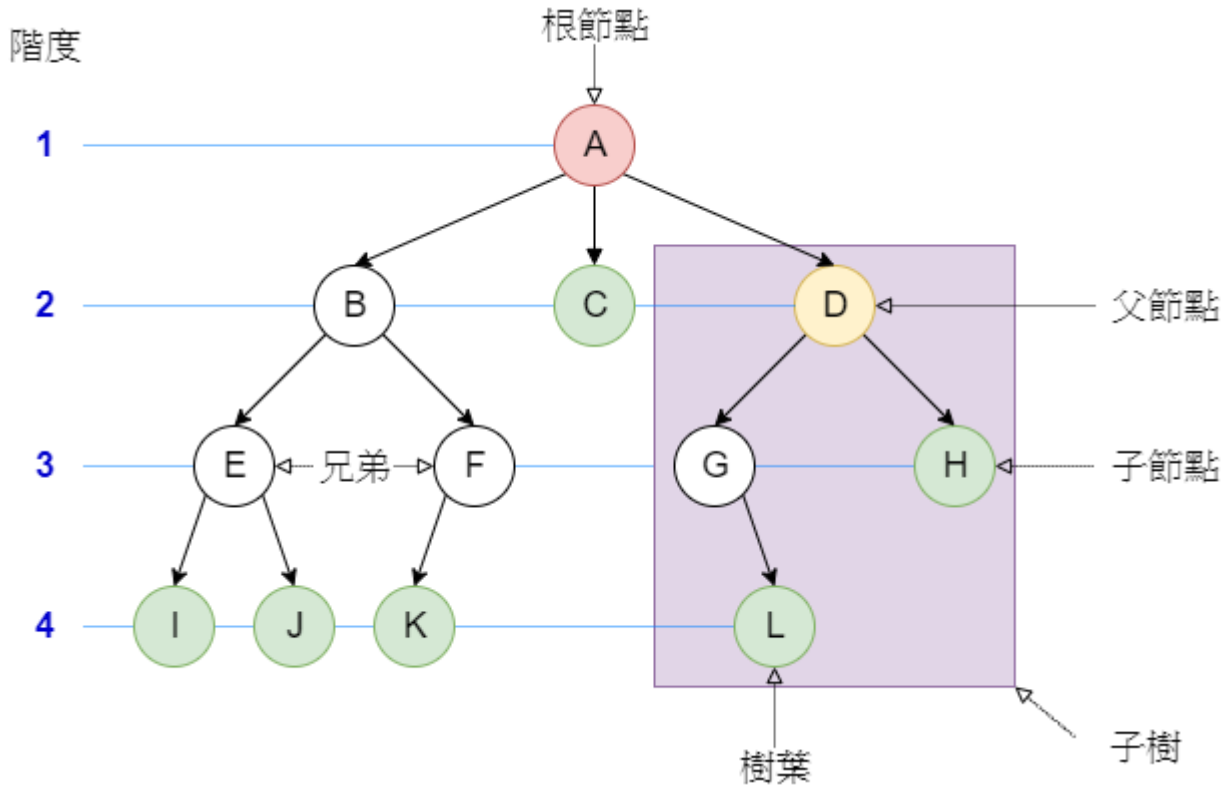


圖 4-3 樹與基本術語

以圖 4-3 為例，樹的基本術語如下：

1. 分支度：

指一個節點含有的子樹個樹，例如：節點 A 的分支度為 2 ，D 為 2，H 為 0。

2. 樹葉：
指分支度為 0 的節點，例如：節點 C, H, I, J, K, L。

3. 子節點：
指某個節點下一層節點，例如：節點 E, F 為 B 的子節點。

4. 父節點：
指某個節點上一層節點，例如：節點 E, F 的父節點為 B。而唯一沒有父節點的為根節點是節點 A 。

5. 兄弟節點：
指具有相同父節點的節點互稱為兄弟節點，例如：節點 E, F 為兄弟節點。

6. 階度：
從根節點開始為第 1 層，根的子節點為第 2 層，以此類推，例如：節點 F 階度為 3。

7. 高度（深度）：
指樹中節點的最大階度為樹的高度，例如：此樹的高度為 4。

四、牛刀小試

1. 下列圖形哪些是樹？正確請畫 ✓，如果錯誤請說明原因。

4.2 二元樹

一、什麼是二元樹?

依照樹的**最大分支度**，樹可以分成很多種，如最大分支度為 2 的樹則稱為二元樹，二元樹也是最廣泛被使用的樹狀結構。

二元樹的特點就是每個節點最多只會有兩個分支，或者說最多只有兩個子樹，在習慣上左邊的子樹稱左子樹，右邊的子樹稱作右子樹。更加嚴格的定義是：二元樹要麼為空，要麼就是由根節點、左子樹和右子樹所組成，而左右子樹各別也是一棵二元樹，如下圖 4-4。

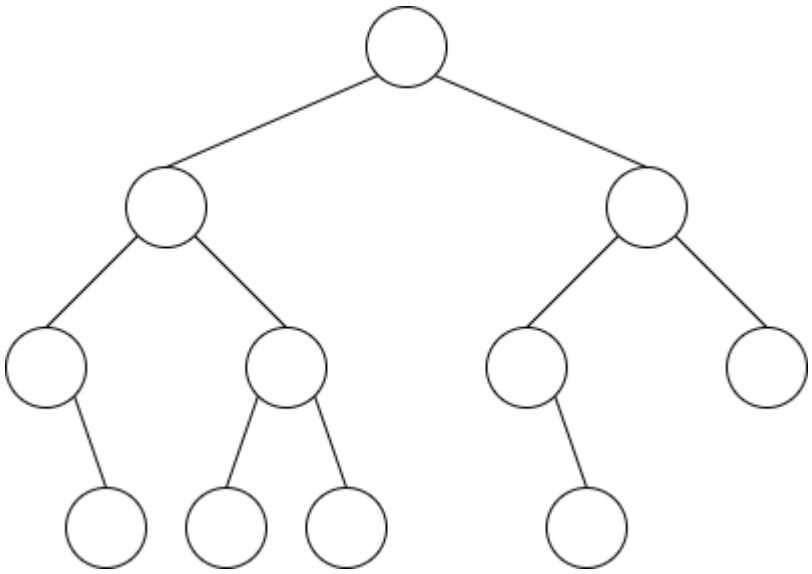


圖 4-4 二元樹

二、特殊的二元樹

二元樹還有三種特殊的二元樹，分別叫做**完滿二元樹**、**完全二元樹**與**完美二元樹**。

1. 完滿二元樹：除了樹葉以外，每個節點必定都有兩個小孩。
2. 完全二元樹：各層節點全滿，除了最後一層，且最後一層節點全部靠左。
3. 完美二元樹：葉子都在最後一層，且其餘每個節點都有兩個小孩，也就是同時滿足完滿二元樹與完全二元樹。

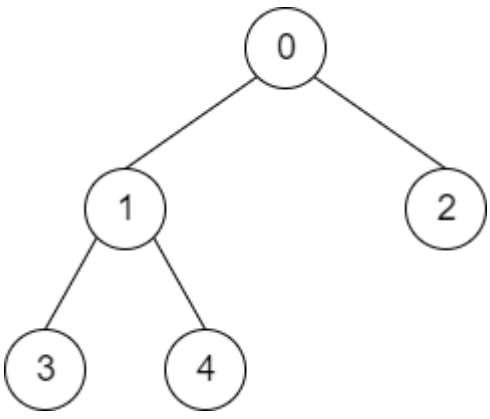


圖 4-5-1 完滿二元樹

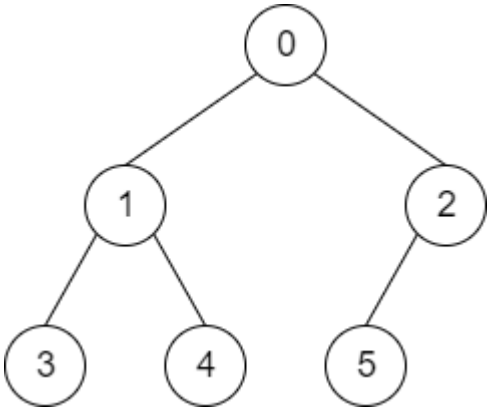


圖 4-5-2 完全二元樹

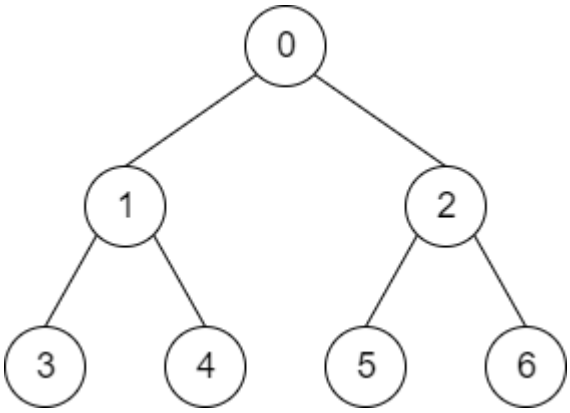


圖 4-5-3 完美二元樹

三、二元樹的儲存

如果想儲存二元樹，可以選擇使用**一維陣列**來做儲存。一棵樹高 H 的二元樹最多可擁有 $2^H - 1$ 個節點，因此使用**一維陣列**來儲存二元樹只需配置 $2^H - 1$ 個元素，就可以儲存樹高 H 的二元樹。在儲存前先將**空子樹**補齊，使它成為完美二元樹，再依序編號，存入陣列對應位置。例如：下圖 4-6 的二元樹樹高為 3 ，最多能擁有 $2^3 - 1 = 7$ 個節點，所以可配置一維陣列 btree [7] 來儲存此樹。

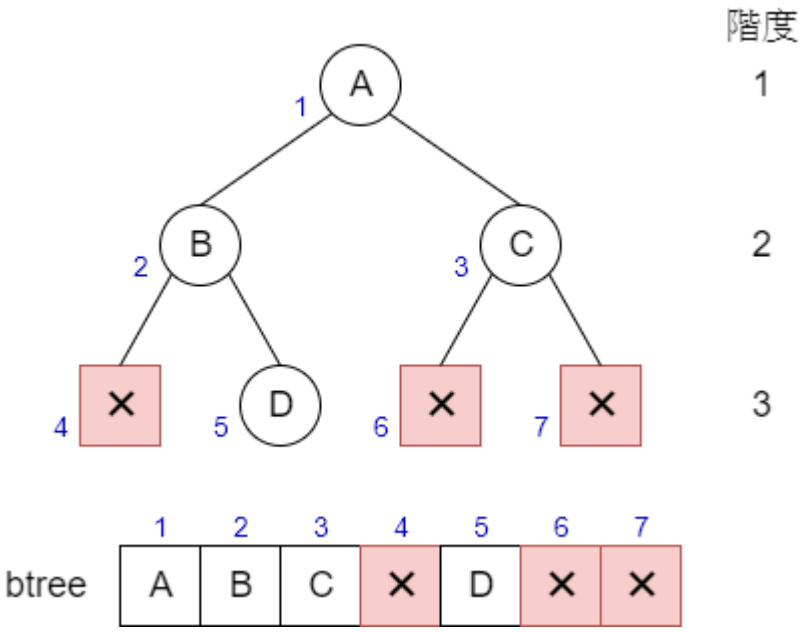


圖 4-6 使用陣列儲存二元樹

由圖 4-6 可以發現三個特點：

1. 左子節點編號是父節點編號乘上 2。
2. 右子節點編號是父節點編號乘上 2 加 1。
3. 父節點編號則是子節點除以 2。

例如：2 號節點的左子節點為 4 號節點(空節點)，右子節點為 5 號節點(D 節點)， 4 號節點和 5 號節點的父節點為 2 號節點。

四、二元樹的走訪

二元樹的**走訪**即是拜訪樹每一個節點，而依據拜訪**根節點**的前中後順序，走訪方式可以分為**前序走訪**、**中序走訪**和**後序走訪**三種方式。

前序走訪	中序走訪	後序走訪
根節點 → 左子樹 → 右子樹	左子樹 → **根節點** → 右子樹	左子樹 → 右子樹 → **根節點**

`if (當前根節點非空) {`
`  拜訪根節點;`
`  用前序走訪拜訪左子樹;`
`  用前序走訪拜訪右子樹;`
`}`
	`if (當前根節點非空) {`
	`  用中序走訪拜訪左子樹;`
	`  拜訪根節點;`
	`  用中序走訪拜訪右子樹;`
	`}`
		`if (當前根節點非空) {`
		`  用後序走訪拜訪左子樹;`
		`  用後序走訪拜訪右子樹;`
		`  拜訪根節點;`
		`}`

如圖 4-7 為例，依前序、中序、後序走訪結果如下：

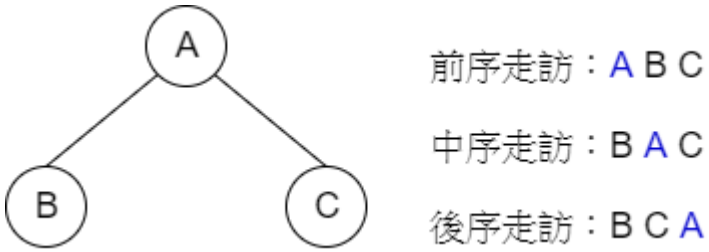
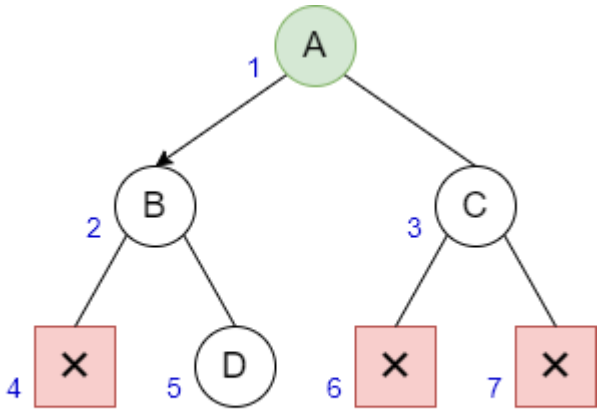


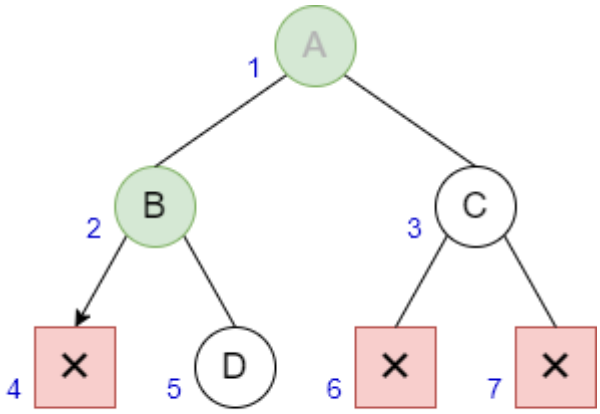
圖 4-7 走訪結果

以圖 4-6 為例，其前序走訪方式的步驟如下：

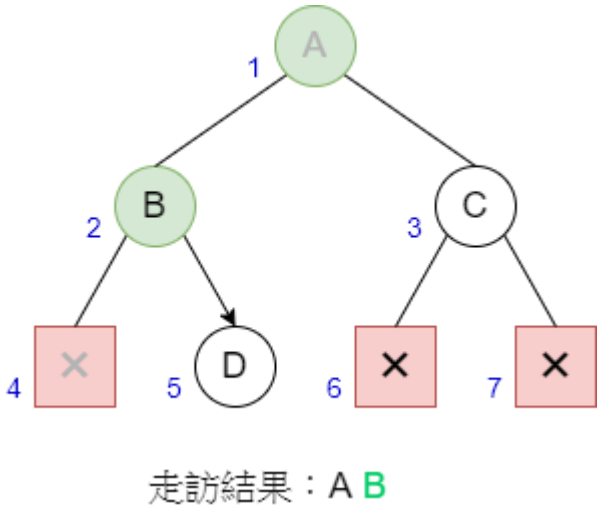
1. 走訪根節點，走訪完畢後，走訪左子樹。



2. 因為左子樹存在，所以走訪其根節點，走訪完畢後，走訪左子樹。

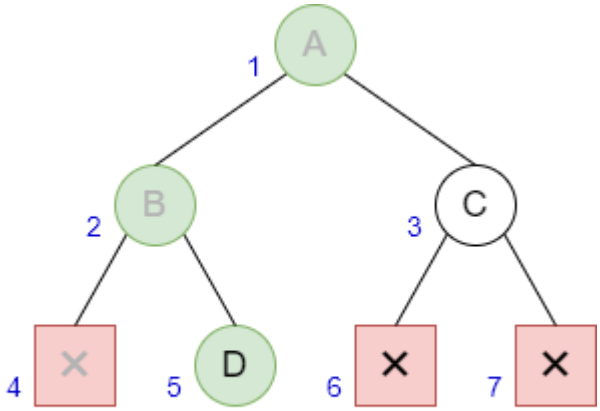


3. 因為左子樹不存在，所以走訪右子樹。



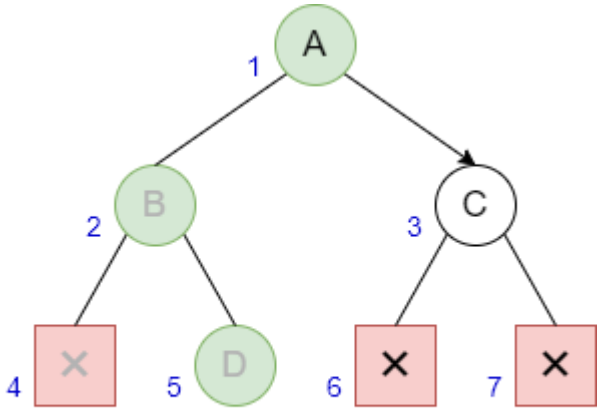
走訪結果：A B

4. 因為右子樹存在，所以走訪其根節點，走訪完畢後，走訪左子樹。



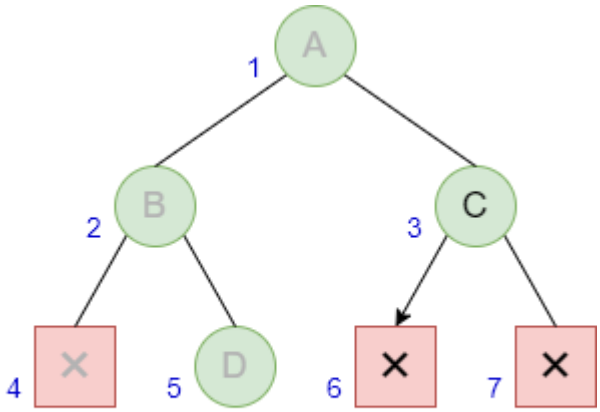
走訪結果：A B D

5. 因為左子樹不存在，因此改往右子樹走訪，結果都不存在，此時對於 A 節點來說，其左子樹的走訪結束，改往右節點走訪。



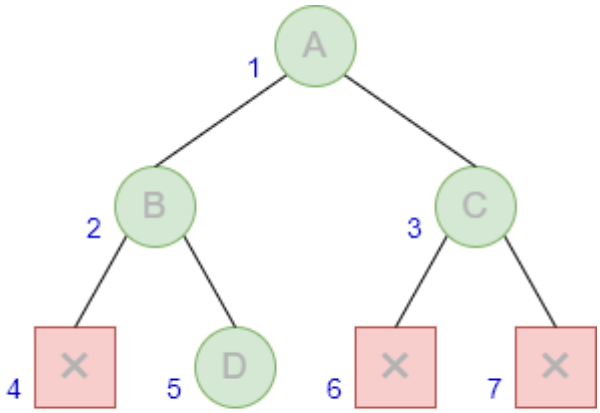
走訪結果：A B D

6. 因為右子樹存在，所以走訪其根節點，走訪完畢後，走訪左子樹。



走訪結果：A B D C

7. 因為左子樹不存在，因此改往右子樹走訪，結果都不存在，此時對於 A 節點來說，其右子樹的走訪結束，中序走訪結束。



走訪最終結果：A B D C

4.3 二元搜尋樹

一、二元搜尋樹的定義

二元搜尋樹，是一種二元樹，必須滿足以下特性：
1. 二元樹的每一個節點值都不相同，不會有重複值。
2. 左子樹節點值 > 根節點值
3. 右子樹的節點值 < 根節點值
4. 左右子樹都是二元搜尋樹

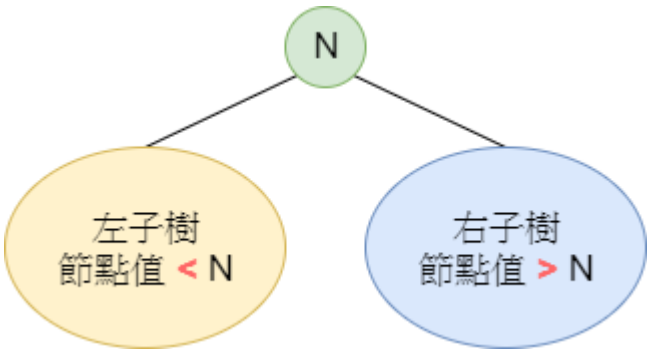


圖 4-8 二元搜尋樹的定義

圖 4-9 A 是一棵二元搜尋樹，根結點為 27，左子樹的值 5, 12, 13 都小於 27；右子樹的值 30, 36, 44 都大於 27。同樣地，此二元搜尋樹的左右子樹也都為二元搜尋樹。

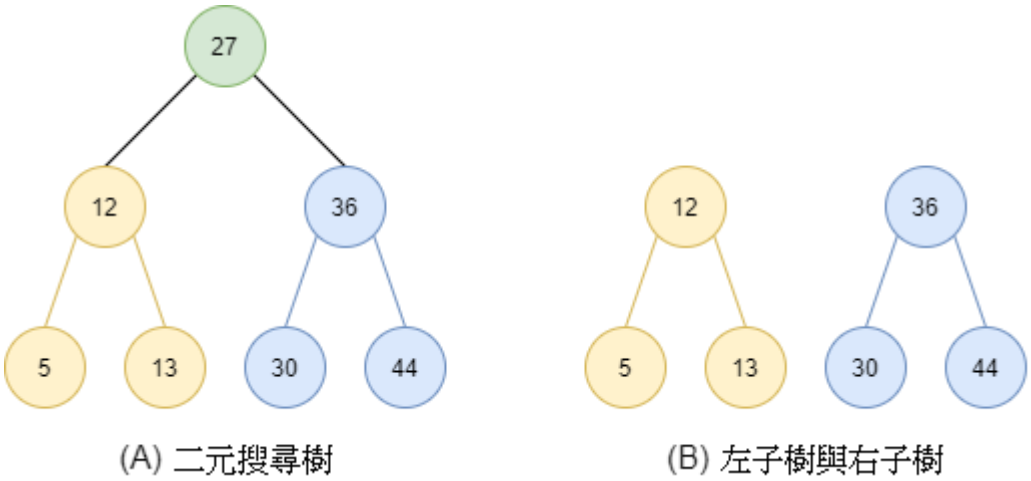


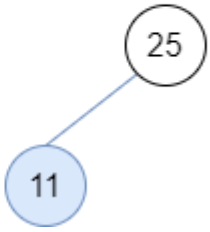
圖 4-9 二元搜尋樹的例子

二、二元搜尋樹的建立

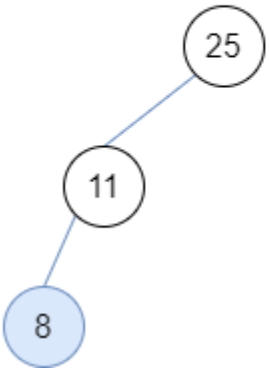
應用二元搜尋樹的定義，可以將資料建成一棵二元搜尋樹。例如資料「25, 11, 8, 16, 32, 28, 44, 15」，將資料建立成一棵二元搜尋樹的步驟如下：

(1) 建立一個節點 25，作為根節點。

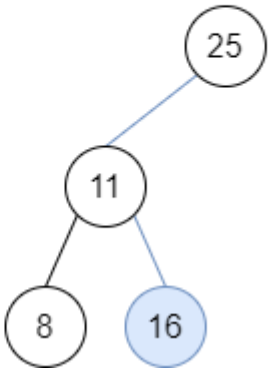
(2) 插入 11。因 11 < 25，所以插入左子樹。



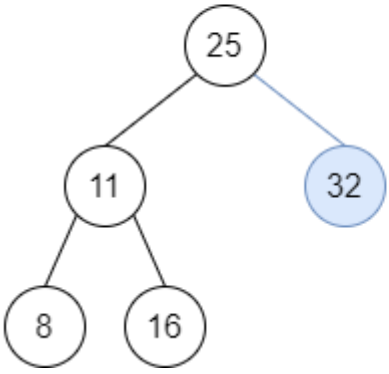
(3) 插入 8。8 < 25，往左子樹。8 < 11，所以插入左子樹。



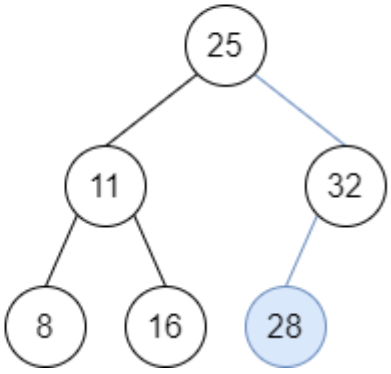
(4) 插入 16。16 < 25，往左子樹。16 > 11，所以插入右子樹。



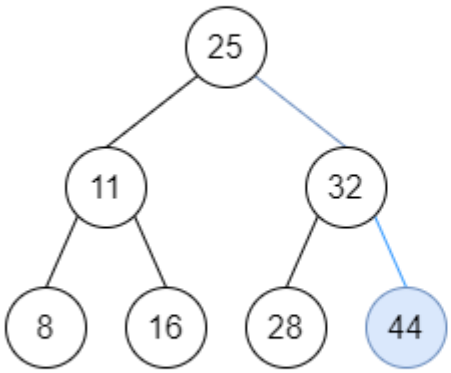
(5) 插入 32。因 32 > 25，所以插入右子樹。



(6) 插入 28。28 > 25，往右子樹。28 < 32，所以插入左子樹。



(7) 插入 44。44 > 25，往右子樹。44 > 32，所以插入右子樹。



(8) 插入 15。15 < 25，往左子樹。15 > 11，往右子樹。 15 < 16，所以插入左子樹。