

# 補充：進階資料結構與演算法

TYIC 桃高資訊社

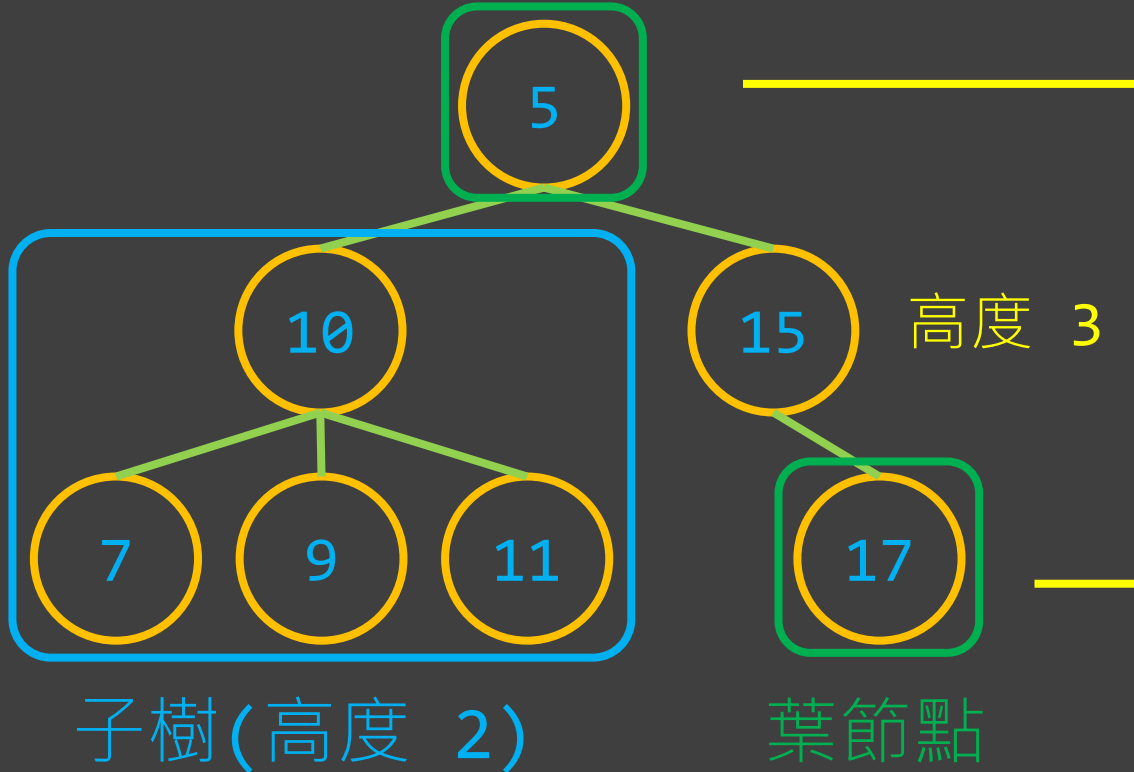
# 樹與二元樹

樹(**tree**)是一種資料結構，樹中的每個資料稱為節點(**node**)  
樹中的資料(節點)不可重複，且節點會連結其他的節點  
連結的節點之間會形成父子關係，但連結不可成環  
每個節點只有一個父節點，但可以有很多個子節點  
相同父節點的節點為兄弟節點，父節點的兄弟節點為叔伯節點  
父節點的父節點為祖父節點，父節點為兄弟節點的節點為堂兄弟節點  
樹中的首個資料為根節點(**root**)，無子節點的資料為葉節點(**leave**)  
從任一葉節點到根節點的最大節點數為該樹的高度(**height**)  
一個節點和其所有子節點可被視為一棵新的樹(子樹)  
且該節點即為新樹(子樹)的根節點  
二元樹(**binary tree**)是一種樹，但每個節點最多只有兩個子節點

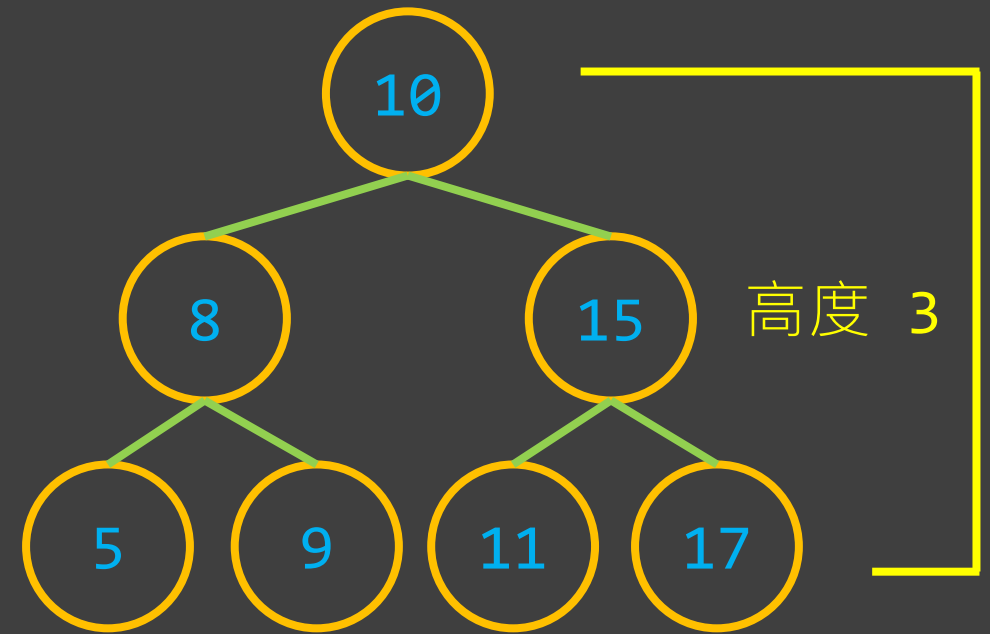
# 樹與二元樹

樹

根節點



二元樹



# 二元搜尋樹

二元搜尋樹(binary search tree)是一種特殊的二元樹

在二元搜尋樹中，比根節點小的資料會放到左子樹中

比根節點大的資料會放到右子樹中

這樣使用二分搜尋法就會非常快速

所以其存取(access)效率較鏈結串列高、較陣列低

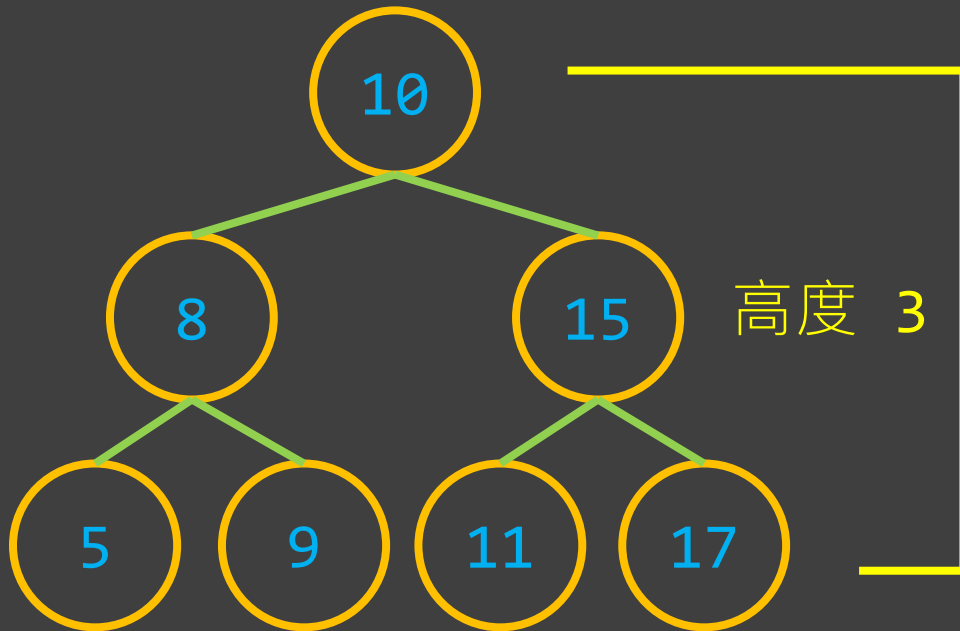
搜尋(search)效率較陣列和鏈結串列高

插入(insert)、刪除(delete)效率較陣列高、較鏈結串列低

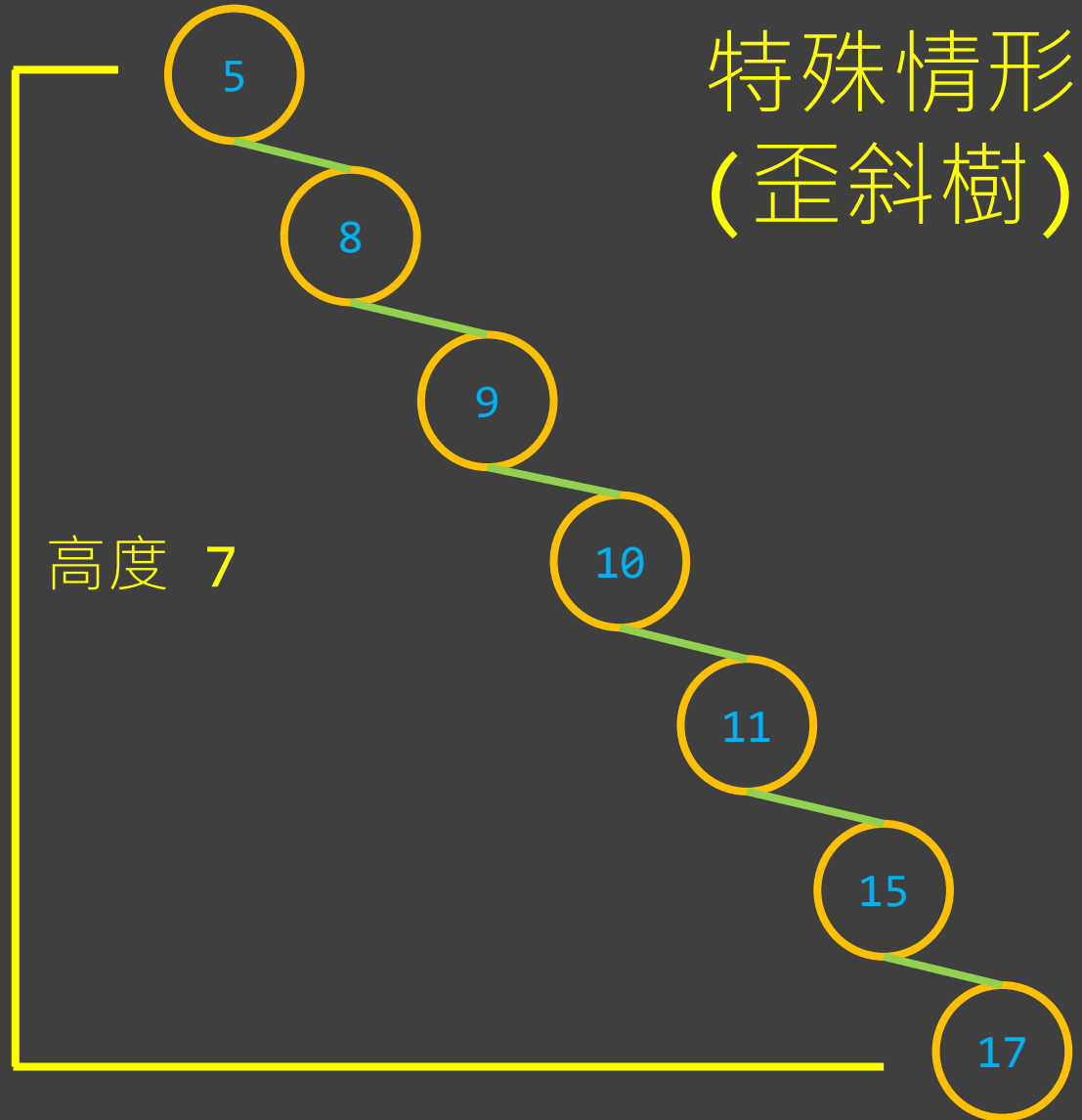
若資料已排序，則放入二元搜尋樹後會變為歪斜樹(skewed tree)

# 二元搜尋樹

正常情形



特殊情形  
(歪斜樹)



# 紅黑樹

紅黑樹(**red-black tree**)是一種二元搜尋樹

紅黑樹會自平衡(**self-balancing**)，避免出現剛剛的特殊情形

紅黑樹的葉節點皆為空資料(**null** 或 **nil**)，並定義了幾條規則：

1. 節點是紅色或黑色
2. 根節點是黑色
3. 所有的葉節點都是黑色
4. 相接節點不能皆為紅色
5. 從根節點到任一葉節點的黑色節點數量皆相同

這樣的規則使得最長路徑長度不超過最短路徑長度的兩倍



紅黑樹線上模擬器

# 紅黑樹

