

Tree

⇒ 非線性階層式資料結構 (non-linear hierarchical)

⇒ 每個節點可以有 0 個或多個子節點，不存在 cycle

⇒ 一般樹轉二元樹：「左 child 右 sibling」轉 90° (P.33~36)

⇒ 子平衡的樹 ⇒ 深度變深 ⇒ 搜尋、插入效率 $O(\log n)$ → $O(n)$ (退化)
→ like linked list

組成元件 {
 Node 节点 (root / internal / leaf / parent / child / sibling)
 Edge 邊
 Subtree 子樹 (left / right)
 Level 层級 ⇒ Root 下 Level 0
 Depth / Height (深度 / 高度) ⇒ Depth (根往下數) / Height (底往上數)
 Branch / Fan out (分支 / 分支度) ⇒ 子節點數量
 → Degree

樹的種類 {
 1. full binary tree → 每個節點有 0 或 2 個子節點
 2. complete binary tree → 除了最後一層其他層全滿，最後一層靠左填充
 3. binary search tree
 4. balanced tree (AVL / Red-Black) → 樹的高度差被控制在一定的範圍內
 5. General tree
 6. N-ary tree → 每個節點最多有 N 個子節點
 7. Trie 字典樹 → 以字元為基礎進行分支
 8. decision tree 決策樹 → 節點代表依序做出的決策
 9. abstract syntax tree → 程式
 10. spanning tree → 連結所有的頂點

- Tree Family
- 1. General Tree → 階層式、不固定子節點數量
 - 2. Binary Tree = 元樹 → 每個節點最多只能有左、右二個子節點 (形狀規則)
 - 3. Binary Search Tree = 元搜尋樹 → 左子樹 < 根節點 < 右子樹 (排序規則) + 形狀
→ 遍迴套用在所有子樹上
→ 大致平衡時可達到高效率搜尋 ($O(\log n)$)
 - 4. Balanced BST 平衡二元搜尋樹 → BST 的特化形式
 - 樹的高度被維持在 $\log n$ (特邊) / 最差 (歪斜) $O(n)$
 - 保證效能穩定、可預期
 - 刪除有子節點的節點
 - 找左子樹最大的
 - 找右子樹最小的
 - 5. AVL tree (一種自我平衡的 BST) → 任何節點的左右子樹高度差不超過 1
 - 形狀 + 排序 + 高度平衡限制
 - Red-Black tree → 形狀 + 排序 + 顏色平衡限制

- DFT (深度优先)
- Preorder → 根左右 \Rightarrow copying tree
 - Inorder → 左根右 \Rightarrow BST
 - Postorder → 左右根 \Rightarrow deletion / freeing memory

ADT - Binary tree \Rightarrow Create / IsEmpty / MakeBT (組合樹) /
Lchild (取左子) / Rchild (取右子) / Data (取值)

concept	time complexity
平衡	$O(\log n)$
DFT	$O(n)$
BFT	$O(n)$

- Binary tree 種類
- 1. Strict (full) → 每個節點有 0 或 1 個子節點
 - 2. Complete → 填滿到最後一層，且最后一層靠左
 - 3. Degenerate (退化成線) → 每個節點只有一個子節點
 - 4. Perfect → 全滿 (三角形)
-
- The diagrams show:
 - Strict (full): A tree where every node has either 0 or 2 children.
 - Complete: A tree where every level is completely filled except for the last level, which is left-justified.
 - Degenerate: A tree where every node has exactly one child, forming a vertical line.
 - Perfect: A tree where every node has exactly two children, forming a full binary tree shape.