一、 基礎概念 (Introduction)

資料結構 (Data Structure)：一種組織與儲存資料的方式，旨在提高資料的使用效率。

演算法 (Algorithm)：解決特定問題或執行計算的有限、嚴謹指令序列。

演算法的五大準則：

1. **輸入 (Input)**：0 個或多個外部輸入。
2. **輸出 (Output)**：至少產生一種結果。
3. **明確性 (Definiteness)**：每條指令必須清晰。
4. **有限性 (Finiteness)**：必須在有限步驟內結束。
5. **有效性 (Effectiveness)**：指令必須簡單到可以用紙筆手算執行。

二、 線性資料結構 (Linear Data Structures)

1. 陣列 (Array)

- **特性**：記憶體空間連續、元素類型相同、支援**隨機存取**。
- **優點**：讀取速度極快(O(1))。
- **缺點**：固定大小；插入或刪除元素時需要移動大量資料(O(n))。

2. 鏈結串列 (Linked List)

- **結構**：由節點 (Node) 組成，每個節點包含「資料」與「指向下一個節點的指針」。
- **類型**：單向、雙向、環狀鏈結串列。
- **優點**：動態大小；插入/刪除僅需修改指針 (O(1))。
- **缺點**：不支援隨機存取，搜尋需從頭開始(O(n))。

3. 堆疊 (Stack)

- **原理**：**後進先出 (LIFO, Last-In-First-Out)**。
- **操作**：Push (推入)、Pop (推出)。
- **應用**：遞迴呼叫 (Call Stack)、括號匹配、瀏覽器後退功能。

4. 隊列 (Queue)

- **原理**：**先進先出 (FIFO, First-In-First-Out)**。
- **操作**：Enqueue (入隊)、Dequeue (出隊)。
- **環狀隊列 (Circular Queue)**：利用模除運算 (Modulo) 重複利用陣列空間，避免空間浪費。
- **應用**：印表機工作排程、CPU 排程。

三、 進階資料結構 (Advanced Data Structures)

1. 雜湊表 (Hash Table)

- **機制**：透過「雜湊函數」將鍵 (Key) 轉換為索引。
- **衝突處理**：
    - **鏈結法 (Chaining)**：在同一個索引位置使用鏈結串列。
    - **開放定址法 (Open Addressing)**：如線性探測 (Linear Probing)、雙重雜湊 (Double Hashing)。
- **效能**：平均情況下的搜尋、插入、刪除均為 O(1)。

2. 樹 (Tree)

- **二元搜尋樹 (BST)**：左子節點 < 根節點 < 右子節點。
- **AVL 樹**：一種自平衡二元搜尋樹，確保左右子樹高度差不超過 1，搜尋效率穩定在 O(logn)。
- **走訪 (Traversal)**：

- **前序** (Root-L-R)、**中序** (L-Root-R)、**後序** (L-R-Root)。

3. 堆積 (Heap)

- **定義**：一種完全二元樹。
  - **最大堆積 (Max Heap)**：根節點為最大值，父節點大於等於子節點。
  - **最小堆積 (Min Heap)**：根節點為最小值，父節點小於等於子節點。
- **應用**：優先隊列 (Priority Queue)、堆積排序 (Heapsort)。

4. 圖形 (Graph)

- **表示法**：相鄰矩陣 (Adjacency Matrix)、相鄰串列 (Adjacency List)。
- **走訪演算法**：
  - **廣度優先搜尋 (BFS)**：使用 **Queue** 實作，適合找最短路徑。
  - **深度優先搜尋 (DFS)**：使用 **Stack/遞迴** 實作，適合探索所有可能路徑。
- **關鍵點**：圖形可能有環，必須記錄「已造訪 (Visited)」節點。