國立臺北教育大學資科所 111-114 年度計概考題學習指南

本指南根據 111 至 114 年度國立臺北教育大學資訊科學系碩士班「計算機概論」考科試題進行深度分析與整理，旨在提供考生一個全面、系統化的複習框架。內容涵蓋了考題的領域分佈統計，並針對各核心題型提供詳盡的解題思路與作答模板，以期幫助考生掌握關鍵概念，並能精準、完整地應對申論題型。

歷屆題型統計分析

下表依據考題索引，統計了 111 至 114 年度各知識領域的出題頻率，有助於考生了解命題重點與趨勢。| 主要領域 | 細分主題 | 相關考題 | 總題數 || ------ | ------ | ------ | ------ || 作業系統與計算機組織 | CPU 效能計算 | 111 年第 2 題 | 8 || | 馮紐曼模型 (von Neumann model) | 111 年第 9 題 | || | 動態連結 (Dynamic linking) | 112 年第 3 題 | || | RISC vs. CISC 架構 | 112 年第 4 題 | || | 多元程式處理 (Multiprogramming) | 112 年第 5 題 (2) | || | 死結 (Deadlock) 概念與處理 | 113 年第 1 題 | || | 虛擬記憶體 (Virtual Memory) | 113 年第 2 題 | || | 死結 (Deadlock) 資源分配判斷 | 114 年第 9 題 | || 資料結構與演算法 | 排序演算法 (Insertion & Selection) | 111 年第 4 題 | 6 || | 霍夫曼編碼 (Huffman coding) | 111 年第 5 題 | || | 二元樹走訪 (Binary Tree Traversal) | 111 年第 7 題 | || | 二元搜尋樹 (BST) | 114 年第 4 題 | || | 選擇排序法 (Selection Sort) | 114 年第 5 題 | || | 霍夫曼編碼 (Huffman coding) | 114 年第 10 題 | || 程式設計 **(C/C++)** | 最大公因數 (GCD) 程式實作 | 112 年第 1 題 | 3 || | C 語言程式碼追蹤與除錯 | 113 年第 5 題 | || | C 語言遞迴階層 (Factorial) | 114 年第 3 題 | || 電腦網路 | DNS 名稱解析 | 111 年第 6 題 (1) | 5 || | 網路備援方案 | 111 年第 6 題 (2) | || | 網路設備功能判斷 | 112 年第 5 題 (4) | || | 無線網路技術 | 113 年第 3 題 | || | 星狀網路 (Star LAN) 容錯 | 114 年第 8 題 | || 資訊安全 | 智慧財產權 (IPR) | 112 年第 5 題 (1) | 5 || | 電子交易安全 | 112 年第 5 題 (3) | || | 災難復原 | 112 年第 5 題 (5) | || | 網路安全技術整合 | 112 年第 6 題 | || | Wi-Fi 安全協定 | 113 年第 3 題 (2) | |
分析洞見：

- 作業系統與計算機組織 是最核心的命題領域，幾乎每年都有深入的申論題，特別是 死結 **(Deadlock)** 與 記憶體管理 相關主題。
- 資料結構與演算法 穩定出現，著重於基本觀念的實作過程展示，如 排序 、樹 的建立與走訪，以及 霍夫曼編碼 。
- 電腦網路 與 資訊安全 題目佔比相當，經常結合出現，涵蓋從底層協定到上層應用的安全機制。
- 程式設計 題目較少，但考驗基本功，包含演算法實作、程式碼追蹤與除錯能力。

核心題型解題模板

一、作業系統 / CPU / 記憶體 / 排程

1. CPU 效能比較 (111 年第 2 題)

題型核心： 給定不同 CPU 的時脈 (Clock Rate)、每指令週期數 (CPI) 與指令數 (Instruction Count)，計算並比較其執行時間。解題模板：

- 寫出公式： 首先明確寫出 CPU 執行時間的計算公式：執行時間 (Execution Time) = 指令數 (Instruction Count) × CPI / 時脈頻率 (Clock Rate)
- 代入計算： 依序計算各個處理器的效能指標。可以直接計算執行時間，或計算每秒可執行指令數 (Instructions Per Second, IPS = Clock Rate / CPI)。
- **P1:** 時脈 3GHz, CPI 1.5 → IPS = 3 / 1.5 = 2 GIPS (每秒 2x10^9 個指令)
- **P2:** 時脈 2.5GHz, CPI 1.0 → IPS = 2.5 / 1.0 = 2.5 GIPS

- **P3:** 時脈 4GHz, CPI 2.0 → IPS = 4 / 2.0 = 2 GIPS
- 比較結果：根據計算結果，IPS 越高代表效能越好。因此 P2 效能最高。
- 回答延伸問題：針對題目的第二部分（效能提升），根據目標建立方程式求解。
- 目標：將 P1, P2, P3 執行時間降低 30%，即執行時間變為原來的 70%。同時 P1, P2, P3 的 CPI 分別上升 20%，即 CPI 變為原來的 1.2 倍。
- 新執行時間 = 0.7 × 原執行時間
- 指令數 × (1.2 × 原CPI) / 新時脈 = 0.7 × (指令數 × 原CPI / 原時脈)
- 1.2 / 新時脈 = 0.7 / 原時脈 → 新時脈 = (1.2 / 0.7) × 原時脈 ≈ 1.714 × 原時脈
- 分別計算 P1, P2, P3 的新時脈。

2. 馮紐曼模型 *(von Neumann Model) (111* 年第 *9* 題*)*

題型核心：解釋馮紐曼模型的架構，並說明在此模型下程式如何被電腦執行。解題模板：

- 定義馮紐曼模型：
- 核心思想：提出「儲存程式 (Stored-program Computer)」的概念，即指令 (Instructions) 和資料 (Data) 都以二進位形式存放在同一塊記憶體中，CPU 可以像讀取資料一樣讀取指令。
- 主要組件：包含五大單元：
- 算術邏輯單元 **(ALU):** 執行算術與邏輯運算。
- 控制單元 **(CU):** 解讀指令，並指揮電腦各單元運作。
- 記憶體 **(Memory):** 存放指令與資料。
- 輸入單元 **(Input):** 接收外部資訊。
- 輸出單元 **(Output):** 呈現處理結果。
- (ALU 和 CU 常合稱為中央處理器 CPU)
- 解釋程式執行流程 **(**取指**-**解碼**-**執行循環**, Fetch-Decode-Execute Cycle)**：
- 取指 **(Fetch):** 控制單元 (CU) 根據程式計數器 (Program Counter, PC) 指向的記憶體位址，從記憶體中取出下一個要執行的指令。
- 解碼 **(Decode):** CU 對取出的指令進行解碼，分析其操作類型（如加法、載入、儲存等）以及操作數的來源。
- 執行 **(Execute):** CU 發出控制信號給相關單元（主要是 ALU）來執行指令。這可能涉及從記憶體或暫存器中讀取資料、進行計算，並將結果寫回記憶體或暫存器。
- 循環：此循環不斷重複，直到程式結束。

3. 死結 *(Deadlock) (113* 年第 *1* 題*, 114* 年第 *9* 題*)*

題型核心：(1) 解釋死結的定義、發生條件、預防與避免策略。(2) 根據資源分配情況，判斷是否發生死結。解題模板 **(**概念題 **113-1)**：

- 何謂 **Deadlock**：一組行程 (Processes) 處於僵持狀態，其中每個行程都在等待一個僅能由該組中另一個行程才能釋放的資源。
- 發生的四個必要條件 **(Coffman's Conditions)**：
- 相互排斥 **(Mutual Exclusion):** 資源一次只能被一個行程使用。
- 佔有與等待 **(Hold and Wait):** 行程已持有一個資源，同時又在等待其他行程持有的資源。
- 不可搶佔 **(No Preemption):** 資源不能被強制從持有它的行程中搶走，只能由該行程自願釋放。
- 循環等待 **(Circular Wait):** 存在一個行程集合 {P0, P1, ..., Pn}，使得 P0 等待 P1 的資源，P1 等待 P2 的資源，...，Pn 等待 P0 的資源。
- **Prevention vs. Avoidance**：

- 預防 **(Prevention):** 透過破壞上述四個必要條件之一來確保死結永遠不會發生。例如:要求行程一次請求所有資源(破壞 Hold and Wait),或允許系統搶佔資源(破壞 No Preemption)。此法較嚴格,可能導致資源利用率降低。
- 避免 **(Avoidance):** 系統在分配資源前,會先檢查這次分配是否會導致系統進入「不安全狀態 (Unsafe State)」。若會,則延遲分配。最著名的演算法是「銀行家演算法 (Banker's Algorithm)」。此法需要行程預先宣告其最大資源需求。
- **Atomic Instruction**:
- 定義: 不可中斷的指令,一旦開始執行,就會完整執行完畢,期間不會被任何其他行程或事件打斷。
- 用途: 主要用於實現同步機制,如號誌 (Semaphores) 或鎖 (Locks)。透過確保「檢查並設置」等操作的原子性,可以防止競態條件 (Race Condition),從而正確地實現相互排斥,這是管理共享資源、避免死結問題的基礎。解題模板 **(實作題 114-9)**:
- 分析資源請求: 列出每個行程持有 (Hold) 與請求 (Request) 的資源。
- 行程 A: 持有檔案 1,請求檔案 2。
- 行程 B: 持有檔案 3,請求檔案 1。
- 行程 C: 持有檔案 2,請求檔案 3。
- 繪製資源分配圖 **(Resource-Allocation Graph)**:
- 用圓圈代表行程 (A, B, C),用方框代表資源 (檔案 1, 2, 3)。
- 從資源指向行程代表「持有」關係 (e.g., 檔案 1 → A)。
- 從行程指向資源代表「請求」關係 (e.g., A → 檔案 2)。
- 判斷是否存在循環: 根據繪製的圖形,尋找是否存在循環路徑。
- 圖中存在一個清晰的循環:A → 檔案 2 → C → 檔案 3 → B → 檔案 1 → A。
- 結論: 明確指出「死結已經發生」。理由是資源分配圖中存在一個循環,且每個資源類型只有一個實例 (Instance)。在此循環中的每個行程都在等待下一個行程所持有的資源,導致所有行程都無法繼續執行。

4. 虛擬記憶體 *(Virtual Memory) (113* 年第 *2* 題*)*

題型核心: 解釋虛擬記憶體的運作原理、分頁與分段的區別、頁表的作用以及頁面錯誤的處理流程。解題模板:

1. 虛擬記憶體運作原理: 說明虛擬記憶體是一種記憶體管理技術,它為每個行程提供一個獨立的、連續的邏輯位址空間,而這個邏輯位址空間可以遠大於實際的實體記憶體。作業系統將行程的邏輯位址空間分割成塊,並將目前需要的部分載入實體記憶體,其餘部分則存放在磁碟上。
2. 分頁 **(Paging) vs.** 分段 **(Segmentation)**: | 特性 | 分頁 (Paging) | 分段 (Segmentation) | | :--- | :--- | :--- | | 分割單位 | 將邏輯位址空間切成固定大小的「頁 (Page)」 | 將邏輯位址空間切成不同大小、具邏輯意義的「段 (Segment)」,如程式碼段、資料段 | | 使用者觀點 | 使用者不可見,由硬體處理 | 使用者可見,程式設計師可指定段 | | 記憶體碎片 | 產生內部碎片 (Internal Fragmentation) | 產生外部碎片 (External Fragmentation) | | 位址對應 | 透過「頁表 (Page Table)」進行一維對應 | 透過「段表 (Segment Table)」進行二維對應 |
3. 頁表 **(Page Table)** 的作用:
4. 定義: 一個資料結構,用於儲存邏輯頁 (Page) 與實體記憶體頁框 (Frame) 之間的對應關係。
5. 運作方式: 當 CPU 產生一個邏輯位址時,記憶體管理單元 (MMU) 會將其拆分為「頁碼 (Page Number)」和「頁內位移 (Offset)」。MMU 以頁碼為索引查詢該行程的頁表,找到對應的實體頁框號碼。最後,將實體頁框號碼與頁內位移組合,形成最終的實體記憶體位址。
6. 頁面錯誤 **(Page Fault)** 的處理:

7. 發生時機：當行程試圖存取一個在頁表中標記為「無效 (invalid)」的頁面時（表示該頁面目前不在實體記憶體中）。
8. 處理流程：
9. CPU 產生一個 trap，將控制權轉交給作業系統。
10. 作業系統檢查此記憶體存取是否合法。
11. 在實體記憶體中尋找一個空閒的頁框 (Frame)。若無空閒頁框，則執行頁面置換演算法 (e.g., LRU) 選擇一個犧牲頁框。
12. 將所需的頁面從磁碟讀取到找到的頁框中。
13. 更新頁表，將該頁的對應頁框號碼填入，並將有效位 (valid bit) 設為「有效 (valid)」。
14. 將控制權交還給原行程，並重新執行導致錯誤的指令。

二、資料結構與演算法

1. 排序演算法過程展示 *(111* 年第 *4* 題*, 114* 年第 *5* 題*)*

題型核心：展示給定數列使用 Insertion Sort 或 Selection Sort 的每一輪 (pass) 排序過程。解題模板 **(**以 **Selection Sort** 為例**,** 資料**: 5, 4, 3, 2, 1)**：

- 簡述演算法原理：選擇排序法的核心是在每一輪中，從尚未排序的數列中找到最小值，並將其放置到已排序部分的末尾。
- 逐步展示過程：清晰地列出每一輪的結果。用 | 符號標示已排序和未排序部分。
- 原始資料**:** 5, 4, 3, 2, 1
- **Pass 1:** 在 5, 4, 3, 2, 1 中找到最小值 1，與第一個元素 5 交換。
- 結果: 1 | 4, 3, 2, 5
- **Pass 2:** 在 4, 3, 2, 5 中找到最小值 2，與第一個元素 4 交換。
- 結果: 1, 2 | 3, 4, 5
- **Pass 3:** 在 3, 4, 5 中找到最小值 3，與第一個元素 3 交換（不移動）。
- 結果: 1, 2, 3 | 4, 5
- **Pass 4:** 在 4, 5 中找到最小值 4，與第一個元素 4 交換（不移動）。
- 結果: 1, 2, 3, 4 | 5
- 最終結果**:** 1, 2, 3, 4, 5解題模板 **(**以 **Insertion Sort** 為例**,** 資料**: 100, 47, 23, 3, ...)**
- 簡述演算法原理：插入排序法的核心是將數列分為已排序和未排序兩部分，每一輪從未排序部分取出第一個元素，插入到已排序部分的正確位置。
- 逐步展示過程：
- 原始資料**:** 100, 47, 23, 3, 35, 14, 9
- **Pass 1:** 將 47 插入 100
- 結果: 47, 100 | 23, 3, 35, 14, 9
- **Pass 2:** 將 23 插入 47, 100
- 結果: 23, 47, 100 | 3, 35, 14, 9
- 依此類推，直到所有元素排序完成。

2. 霍夫曼編碼 *(Huffman Coding) (111* 年第 *5* 題*, 114* 年第 *10* 題*)*

題型核心：根據字元出現的頻率，建立霍夫曼樹並產出對應的編碼。解題模板：

- 列出頻率並建立節點：將每個字元及其頻率視為一個獨立的樹節點。
- (111-5): A(11), B(8), C(9), D(20), E(31), F(14), G(10)
- (114-10): A(1), B(1), C(1), ..., J(1)
- 逐步建構霍夫曼樹：
- 重複以下步驟，直到只剩下一個根節點：
- 從所有節點中，選出頻率最小的兩個節點。
- 建立一個新的父節點，其頻率為這兩個子節點頻率之和。

- 將這兩個最小頻率節點作為新父節點的左右子節點。
- （重要）最好能畫出樹的合併過程或最終的樹狀結構圖。
- 標示邊緣並產生編碼：
- 從根節點開始，為所有左分支標示為 0，右分支標示為 1（或反之，但需保持一致）。
- 每個字元的霍夫曼碼即為從根節點到該字元葉節點的路徑所組成的 0 和 1 字串。
- 列出結果：以表格形式清晰呈現每個字元對應的編碼。
- 範例：| 字元 | 頻率 | 編碼 | | :--- | :---: | :--- | | E | 31 | 00 | | D | 20 | 01 | | ... | ... | ... |

三、網路與資安

1. DNS 名稱解析流程 (111 年第 6 題)

題型核心：說明從輸入網域名稱到取得 IP 位址的完整流程。解題模板：

- 使用者輸入：使用者在瀏覽器輸入網域名稱，例如 www.ntue.edu.tw。
- 本機快取查詢：作業系統首先檢查自身的快取 (cache) 中是否有該網域的紀錄。
- 遞迴查詢 **(Recursive Query)** 至 **Local DNS Server**：
- 若本機快取沒有，作業系統會向設定好的 Local DNS Server（通常由 ISP 提供）發出查詢請求。
- **Local DNS Server** 的迭代查詢 **(Iterative Query)**：
- Local DNS Server 檢查自身快取。若無，它會代替使用者進行一系列查詢：
- **Step 1:** 查詢根 **(Root)** 伺服器：Local DNS Server 問 Root Server：「誰知道 .tw？」
- **Step 2: Root** 回應 **TLD** 伺服器位址：Root Server 回答：「我不知道，但你可以去問負責 .tw 的頂級域 (Top-Level Domain, TLD) 伺服器，它的位址是...」
- **Step 3:** 查詢 **TLD** 伺服器：Local DNS Server 接著問 .tw TLD 伺服器：「誰知道 edu.tw？」
- **Step 4: TLD** 回應權威伺服器位址：TLD Server 回答：「我不知道，但你可以去問負責 edu.tw 的權威名稱伺服器 (Authoritative Name Server)，它的位址是...」
- **Step 5:** 查詢權威伺服器：Local DNS Server 再問 edu.tw 伺服器：「www.ntue.edu.tw 的 IP 位址是什麼？」
- **Step 6:** 權威伺服器回應 **IP**：權威伺服器查詢自己的紀錄，回覆最終的 IP 位址。
- 回傳與快取：
- Local DNS Server 將取得的 IP 位址回傳給使用者的電腦。
- 同時，Local DNS Server 會將這個對應關係快取起來，以加速未來的查詢。

2. 無線網路技術 (113 年第 3 題)

題型核心：解釋無線網路的各種模式、安全協定、SSID 及常見問題。解題模板：

- **Ad Hoc vs. Infrastructure** 網路模式：
- **Ad Hoc (點對點模式)**：一個由多個無線裝置直接相互連接組成的去中心化網路，不需透過中央存取點 (Access Point, AP)。適用於臨時性、小範圍的資料交換。
- **Infrastructure (基礎架構模式)**：所有無線裝置都透過一個中央的 AP 連接。AP 負責管理網路通訊，並可將無線網路橋接至有線網路（如網際網路）。這是最常見的 Wi-Fi 使用模式。
- **WEP / WPA / WPA2** 的比較：
- **WEP (Wired Equivalent Privacy):** 最早的 Wi-Fi 安全協定，使用 RC4 加密演算法與靜態金鑰。因存在嚴重設計缺陷，現已極不安全，可輕易被破解。
- **WPA (Wi-Fi Protected Access):** 為取代 WEP 的過渡方案。引入 TKIP (Temporal Key Integrity Protocol) 動態金鑰機制，雖仍使用 RC4，但安全性遠高於 WEP。

- **WPA2 (Wi-Fi Protected Access II):** 目前最廣泛使用的安全標準。採用更強大的 AES (Advanced Encryption Standard) 加密演算法與 CCMP 協定，提供非常高的安全性。
- **SSID (Service Set Identifier)**:
- 定義：無線網路的名稱，是一個長度最多 32 字元的字串，用於識別一個特定的無線區域網路 (WLAN)。無線基地台 (AP) 會廣播其 SSID，讓客戶端裝置可以發現並連接到該網路。
- **Hidden Node Problem (**隱藏節點問題**)**:
- 問題描述：在一個無線網路中，當節點 A 和節點 C 都在 AP 的通訊範圍內，但 A 和 C 卻互相聽不到對方的信號。如果 A 和 C 同時向 AP 傳送資料，將會在 AP 端發生訊號碰撞 (Collision)，導致傳輸失敗。
- 解決方案：使用 RTS/CTS (Request to Send / Clear to Send) 機制。當一個節點要傳送資料前，先向 AP 發送一個短的 RTS 請求。AP 收到後，會廣播一個 CTS 封包給所有在範圍內的節點。即使是隱藏節點（如 C），收到 CTS 後也會知道媒體正忙，從而延後自己的傳送，避免碰撞。

3. 網路安全技術整合 *(112* 年第 *6* 題*)*

題型核心：解釋對稱/非對稱加密、雜湊函數、數位簽章、數位憑證等核心資安技術。解題模板：

- 對稱式加密 **(Symmetric)**:
- 原理：加密和解密使用同一把「共享金鑰 (Shared Key)」。
- 優點：速度快，效率高。
- 缺點：金鑰配送困難，若金鑰被竊取，安全性即被破壞。
- 非對稱式加密 **(Asymmetric)**:
- 原理：使用一對「公鑰 (Public Key)」與「私鑰 (Private Key)」。公鑰可對外公開，用於加密；私鑰由本人保管，用於解密。
- 優點：解決了金鑰配送問題，安全性高。
- 用途：除了加密外，私鑰加密、公鑰解密可用於實現「數位簽章」。
- 雜湊函數 **(Hash Function)**:
- 原理：將任意長度的輸入資料，轉換成固定長度的輸出值（雜湊值）。
- 特性：單向性（不可逆）、輸入的小改變會導致輸出的巨大改變、難以找到兩個不同輸入有相同輸出（抗碰撞）。
- 應用：主要用於驗證「資料完整性 (Integrity)」，確保資料在傳輸過程中未被竄改。
- 數位簽章 **(Digital Signature)**:
- 目的：提供資料來源的「身份認證 (Authentication)」、「完整性 (Integrity)」與「不可否認性 (Non-repudiation)」。
- 運作機制：
- 傳送方先對訊息內容進行 雜湊 運算，得到訊息摘要。
- 傳送方再用自己的 私鑰 對此摘要進行 加密，產出數位簽章。
- 接收方收到訊息與簽章後，用傳送方的 公鑰 對簽章 解密，還原出原始摘要 A。
- 接收方對收到的訊息本身重新進行一次 雜湊 運算，得到摘要 B。
- 比較 A 和 B 是否相符，若相符則簽章有效。
- 數位憑證 **(Digital Certificate)**:
- 目的：解決公鑰的信任問題，將一個公鑰與一個實體（如個人、網站）的身分綁定。
- 機制：由一個受信任的第三方機構，稱為「憑證頒發機構 (Certificate Authority, CA)」，來簽發。憑證內含持有者的身分資訊、持有者的公鑰，並由 CA 用其自身的私鑰進行數位簽章。使用者可透過驗證 CA 的簽章來確認憑證的真實性。