《資料結構》

一、有位程式設計師在撰寫程式時遇到了一個難解的問題，後來發現有兩個演算法可以解這個難題：演算法A的時間複雜度為$O(n^2\log(n!))$，演算法B的時間複雜度為$O(n^2((\log n)!))$。假設輸入資料的個數n通常都很大，他應該選擇那個演算法比較好，原因何在？（20分）

試題評析	本題為時間複雜度判定的問題，因為資料個數很大，所以要比較兩個演算法時間複雜度等級，選擇等級較低者。
考點命中	《高點資料結構講義》，王致強編著，頁1-16~1-17。

答：

演算法A：$O(n^2\log(n!))=O(n^2\times n\log n)=O(n^3\log n)$

演算法B：取$m=(\log n)!$，而$\log m=\log(\log n)!=\Theta((\log n)(\log\log n))=\Theta(\log n^{\log\log n})$，

故$m=\Theta(n^{\log\log n})$，最後可得時間$O(n^2((\log n)!))=O(n^{2+\log\log n})$

因為演算法B時間複雜度的冪數2+loglogn會隨n成長，故演算法B時間複雜度較高，所以在資料量極大時，選擇演算法A比較省時間，比較好！

二、樹（tree）是一個很常用的資料結構。一個樹是指一個沒有迴圈（cycle）的聯通圖（connected graph）。（每小題10分，共20分）

（一）證明：每個具有n個節點（node）的樹，$n>1$，至少有2個分支度（degree）為1的節點。（分支度就是指有多少邊以此節點為端點。）

（二）用前項結果證明：每個具有n個節點的樹，$n>1$，恰好有$n-1$個邊（edge）。

試題評析	本題為圖形基本特性問題，唯證明部份，平時沒有經常思考相關問題者，考試時比較不易想出來，本題大部份考生可能不易取分。
考點命中	《高點資料結構講義》，王致強編著，頁8-41。

答：

(一) 因為樹是非循環圖(acyclic graph)，故邊的個數 e≦n-1…(1)。

另外，所有節點分支度的總和$\sum_{v\in V}\deg(v)=2e$…(2)，可以假設分支度為1的節點有 x 個，

故$\sum_{v\in V}\deg(v)\ge x\times1+(n-x)\times2$…(3)。

綜合(1)(2)與(3)，得$2(n-1)\ge 2e\ge x+2(n-x)$，可解得$x\ge2$。

(二) 使用歸納法證明

basis：n=2，兩個節點的連通非循環圖形，明顯可見邊的個數=1，符合題意。

hypothesis：假設連通非循環圖形有n個節點，則邊的個數為n-1。

induction：當連通非循環圖形的節點個數增加為n+1時，由(一)的結論，分支度為1的節點至少會有2個。取其中一個分支度為1的節點刪除，其連接的邊也一併移除，則剩下來的會是一棵n個節點的連通非循環圖形，由hypothesis，其邊的個數為n-1。

所以，未刪除節點與邊之前，會有n+1個節點，其邊的個數為n=(n-1)+1，故得證。

三、給定一個權重圖（weighted graph），$G=(V,E,w)$，其中每個邊（edge）e的權重$w(e)$都是正整數，為了簡單，假設$V=\{1,2,\ldots,n\}$。任意點v與起始點s的距離可以用一個矩陣$d[1..n]$來表示。（每小題10分，共20分）

(一)設計一個只需 $O(n)$ 空間的方法來記錄從 s 出發，到達每個點的最短路徑。

(二)說明計算與印出從起始點 s 到任意點 $t \in V$ 的最短路徑的演算法。(解此小題時可參考Dijkstra或其他演算法來設計，且不須將Dijkstra或別的演算法做詳細的描述。)

試題評析	本題為 Dijkstra 演算法考題，主要測驗演算法的資料結構與最短路徑表示法的設計，對Dijkstra演算法相關內容，上課內容有詳細了解的考生，應可拿到不錯分數。
考點命中	《高點資料結構講義》，王致強編著，頁8-61~8-63。

答：

(一)使用一個prev陣列，用prev[v]記錄節點v在最短路徑上的前一節點，這樣可以構成一棵sink tree，使用此一sink tree，每個節點可以逆向回溯到s的路徑。再使用堆疊或遞迴程式，將路徑反轉，即可得到s到v的路徑。

因為陣列大小即為節點總數n，故空間複雜度為O(n)。

(二)計算s到t的最短路徑：直接使用Dijkstra演算法，直到節點t找到最短路徑時，即可提前結束演算法。

印出起始點到t的路徑：使用一個堆疊來回溯最短路徑，演算法如下：

```
createStack();
p←t;
while (p≠s) {
      push(p);
      p←prev[p];
}
push(s);
print "shortest path:";
while (not stackEmpty())    print pop();
```

四、有個矩陣 $A[1..n]$，n 的值很大。在矩陣 A 中存有 n 個正整數，且從小到大排列。給定某個整數 x，二分搜尋法(binary search)可以在 $O(\log n)$ 的時間內找出 x 在矩陣 $A[1..n]$ 的位置，或宣告在 $A[1..n]$ 中沒有 x。在某個應用中，已知絕大部分的 x 都會出現在矩陣 $a[1..n]$ 的前面 m 個元素，且 m 的值遠小於 n，但是無法預知 m 的範圍。設計一個演算法，可以在 $O(\log m)$ 的時間內完成搜尋。(20分)

試題評析	本題為二分搜尋法的變化題，除了利用二分搜尋法之外，還須先設法快速找到搜尋的範圍。本題有一定的難度，有點屬於演算法的問題，除非想出相關技巧，否則拿分不易。
考點命中	《高點資料結構講義》，王致強編著，頁10-3~10-5。

答：

方法如下：

(1) 先找尋x在陣列A中，可能的範圍m。

(2) 在A[1]~A[m]進行二分搜尋。

演算法虛擬碼如下：

1. m←1;
2. while (m<n and A[m]<x) m←m*2; //有需要時，將m值加倍，以找到適當的m。
3. if (m>n) m←n;
4. 在A[1]~A[m]之間，以二分搜尋法找出 x；

時間分析

第1行O(1)

第2行O(logm)，因為是以兩倍兩倍方式增加，由1增加到適當的m，只要O(logm)時間即可做到。

第3行O(1)

第4行O(logm)，因為陣列已經排序好，所以可以使用二分搜尋法。
故總時間O(1)+O(logm)+O(1)+O(logm)=O(logm)。

五、假設有個矩陣 $A[1..n]$ 儲存 n 個整數。Quick sort是一個排序演算法。假設有個副程式partition(A,l,r) 其輸入參數 A 是一個矩陣，$l,r,l<r<n$，是兩個指標。其回傳的值 m 也是一個指標。這個副程式可將矩陣中從 l 到 r 的這一段資料 $A[l..r]$ 區分成兩段：$A[l..m]$ 和 $A[m+l..r]$，使得在 $A[l..m]$ 中的元素都小於或等於 x，而在 $A[m+l..r]$ 中的元素都大於或等於 x，其中 x 是從 $A[l..r]$ 中隨機選擇的一個整數。接下來要在此兩段資料遞迴執行partition。避免這些遞迴計算可以用一個堆疊（stack）來處理。假設partition(A,l,r) 回傳 m，則執行：

if $(l<m)$ push (l,m) into stack
if $(m+1<r)$ push $(m+1,r)$ into stack

一開始，堆疊中只有一組資料，$(1,n)$ 表示 $A[1..n]$ 需要排序。如此反覆將堆疊最上面的資料 (l,r) 移出，執行partition(A,l,r)，直到堆疊沒有資料為止。（每小題10分，共20分）
(一)證明在最糟情況下，堆疊的高度可以達到 $n/2$。
(二)設計一個好的演算法以降低stack的高度，並證明堆疊的高度最多只需要 $\log n+1$。

試題評析	本題為Quick Sort 變化題，以堆疊取代遞迴方式，實作Partition，測驗空間複雜度的分析，同時也測驗選擇基準值的一些技巧。
考點命中	《高點資料結構講義》，王致強編著，頁9-36。

答：
(一)最糟情況，發生在每次partition選擇的x，恰好都是群組中第二小的資料，partition後A[l..m]只有最小兩項，會先push 到堆疊較底部；而其餘資料皆在A[m+1..r]，會Push到堆疊較靠頂端。
故n項資料最糟情況的堆疊高度T(n)，可以用下面關係式來推算

$$T(n)=\begin{cases}1 & ,n<2\\ 1+T(n-2) & ,n\geq 2\end{cases}$$

可解得T(n)=n/2。

(二)演算法如下
假設 partition (A,l,r) 傳回 m 後，先 psuh 較長的分割，再 push 較短的分割，做法如下：

```
if (m-l+1 > r-m) {
    if (l<m) push (l,m) into stack
    if (m+1<r) push (m+1,r) into stack
} else {
    if (m+1<r) push (m+1,r) into stack
    if (l<m) push (l,m) into stack
}
```

這樣就可以有效降低 stack 的高度，n 項資料進行 Quick sort 最糟情況時的高度 T(n) 可定義如下：

$$\mathrm{T(n)}=\begin{cases}1+T(n/2) & ,n\geq 1\\ 0 & ,n<1\end{cases}$$

取 $n=2^k$，即可證明

$$T(n)=T(2^k)\leq 1+T(2^{k-1})\leq 2+T(2^{k-2})\leq ...\leq k+T(2^0)\leq k+1=\log n+1$$ 。