# 《資料結構》

一、考慮數字1到n，若將其順序重新排置，每個排列順序都稱作一個排列或置換（Permutation），例如5 1 4 3 2是1 2 3 4 5的一個排列。我們可以將一個數字1到n的排列視為一個順序的映射$P$，則前述例子可表示為$P(5)=1$、$P(1)=2$、$P(4)=3$、$P(3)=4$、$P(2)=5$。當然，1 2 3 4 5也是1 2 3 4 5的一個排列。在一個數字1到$n$的排列$P$中，若一對數字$i$和$j$，$1 \le i < j \le n$，$P(j) < P(i)$，也就是在排列$P$中較大的數字$j$出現在較小的數字$i$左邊（前面），我們稱此對數字為反向（Inversion），而排列$P$的反向數（Inversion number）則定義為排列$P$中反向的總數量。請回答下列問題：

(一)數字1到$n$的何種排列會有最大的反向數？最大反向數是多少？（5分）

(二)若給定一個數字1到$n$的排列$P$，請提出一個線性遞迴（Linear Recursive）的方式來算出排列$P$的反向數，並提供虛擬碼（Pseudo-code）與時間複雜度分析。（10分）

| | |
|---|---|
| **試題評析** | 反向數為排序的一個基礎觀念，本題結合反向數與遞迴程式命題，有一點難度；考生若未能從題意理解反向數的特點，可能不易取得高分。 |
| **考點命中** | 《資料結構》，高點文化出版，王致強編著，頁9-6：精選範例3。 |

**答**：

(一) 當排列為遞減順序時，會有最大的反向數。意即排列順序為

n, n-1,n-2, … , 3, 2, 1

此時，反向數 $= 0 + 1 + 2 + 3 + \cdots + (n-1) = \frac{n(n-1)}{2}$

(二) 若計算一個數字 i 的反向數函數為 Inv(i, j)，其線性遞迴函數如下:

```
Inv(i, j) {
   if (j>n) return 0;
   else if (j>i && P(j)<P(i)) return Inv(i, j+1) + 1;
   else return Inv(i, j+1);
}
```

呼叫 Inv(i, i+1) 就可以計算數字 i 的單獨一項的反向數，其時間複雜度為O(n)；

若要計算一組排列P的反向數，可以用下面程式：

```
Inversion=0;
for (i=1; i<n; i++)
     Inversion += Inv(i, i+1);
```

總時間複雜度為 nxO(n)=O($n^2$)。

二、優先佇列(Priority Queue)是依管理物件的優先權來考量，在此我們考慮管理物件的鍵值(Key)愈小其優先權愈高，兩個主要操作則分別為加入（Insert）與擷取最小者（Delete_Min）。

(一)請說明如何利用優先佇列對n個鍵值進行排序。（6分）

(二)我們使用一個未排序的陣列（Unsorted Array）來管理鍵值以實現一個優先佇列，請回答下列問題：（10分）

(1)若有$n$個鍵值，請說明兩個主要操作（加入（Insert）與擷取最小者（Delete_Min））的時間複雜度。

(2)請判斷下面的敘述是否為真，並請說明原因：

若以此優先佇列進行排序（Sorting），其所對應的排序原理為插入排序（Insertion Sort）。

(三)二元堆積（Binary Heap）是一個優先佇列的資料結構，因為我們考慮鍵值小的物件有高的優先權，所以又可稱為最小堆積（Minimum Heap）。（14分）

(1)在結構上最小堆積為一個完全二元樹（Complete Binary Tree），若使用一個陣列來實作最小堆積，陣列中物件的鍵值放置如下，請描述此陣列對應的完全二元樹（以樹狀結構表示）。

| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Key | 35 | 18 | 42 | 24 | 7 | 14 | 25 | 12 | 38 | 21 |

(2)請說明二元堆積中何謂堆積特性（Heap Property）？

(3)前揭(1)中的完全二元樹並未有堆積特性，請將其進行堆積化（Heapify），並以陣列表示出堆積化後的最小堆積所對應之完全二元樹。

| | |
|---|---|
| **試題評析** | 此題為綜合性質考題，實際上為普通難度，前段先考使用陣列來實作優先權佇列；最後一題考的是使用堆積來實作，以及建立堆積的方法。此題大部份考生應可取得不錯分數。 |
| **考點命中** | 1.《資料結構》高點文化出版，王致強編著，頁7-7：精選範例2。<br>2.《資料結構》高點文化出版，王致強編著，頁9-60：精選範例36。 |

**答**：

(一) 先將資料逐一加入(Insert)優先權佇列，然後持續擷取最小者(Delete_Min)，即可由小而大取出資料，完成排序。

(二)

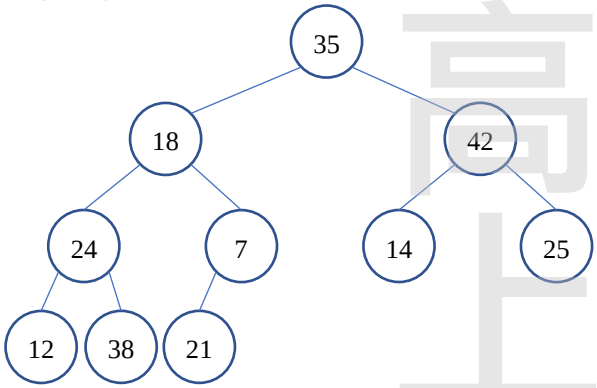
(1)Insert只要將資料加到末端即可，時間複雜度O(1)。

Delete_Min 則必須搜尋整個陣列，才能找出最小的項目，再予以刪除，時間複雜度為O(n)。

(2)此述敘為偽，每次須搜尋剩餘部份的最小項目，原理比較類似選擇排序，而非插入排序。

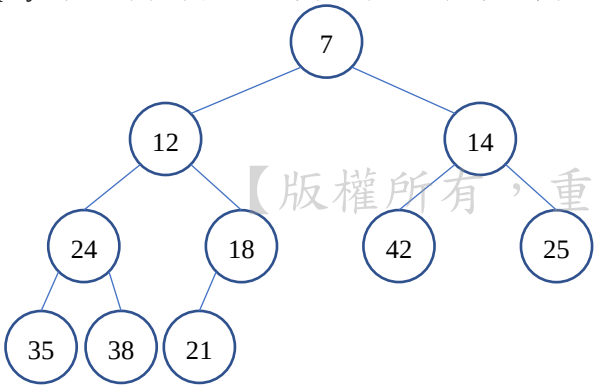
(三)

(1)完全二元樹如下圖




(2)最小堆積除了結構上須為完全二元樹之外，也同時必須是最小樹(min-tree)，也就是個個節點必須小於或等於其兒子(children)。

(3)Hepify就是由下往下做sift，最後逐步調整成最小堆積結構，最後的堆積為



以陣列表示如下：

| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Key | 7 | 12 | 14 | 24 | 18 | 42 | 25 | 35 | 38 | 21 |

三、請回答下列關於AVL樹（AVL Tree）的問題：

（一）我們欲將所管理的鍵值（Key）依序列出，請問是否可以利用一個AVL樹對鍵值來進行排序（Sorting）？若不行，請說明原因；如果可以，請描述方法及時間複雜度。（5分）

（二）請提供一個線性時間的演算法來判斷一個二元搜尋樹是否為AVL樹。（10分）

（三）在AVL樹上進行一個加入（Insert）操作後，是否最多只需要一次的重構（Restructuring）即可恢復其平衡的特性？請說明原因。（10分）

| | |
|---|---|
| **試題評析** | 此題亦為綜合性考題，前半段測驗搜尋樹的排序方法，並使用AVL Tree來進行排序；後半段則測驗AVL Tree本身的特性，以及旋轉的方式。 |
| **考點命中** | 1.《資料結構》高點文化出版，王致強編著，頁6-40，要點：樹排序法。<br>2.《資料結構》高點文化出版，王致強編著，頁11-12，要點：AVL Tree 插入資料方。 |

**答**：

(一)將資料逐一插入 AVL Tree，再對 AVL Tree 做中序追蹤，來輸出資料，即可排序完成。

建立 AVL Tree 需要 O(nlogn)時間，中序追蹤需要 O(n)時間，故總時間為 O(nlogn)。

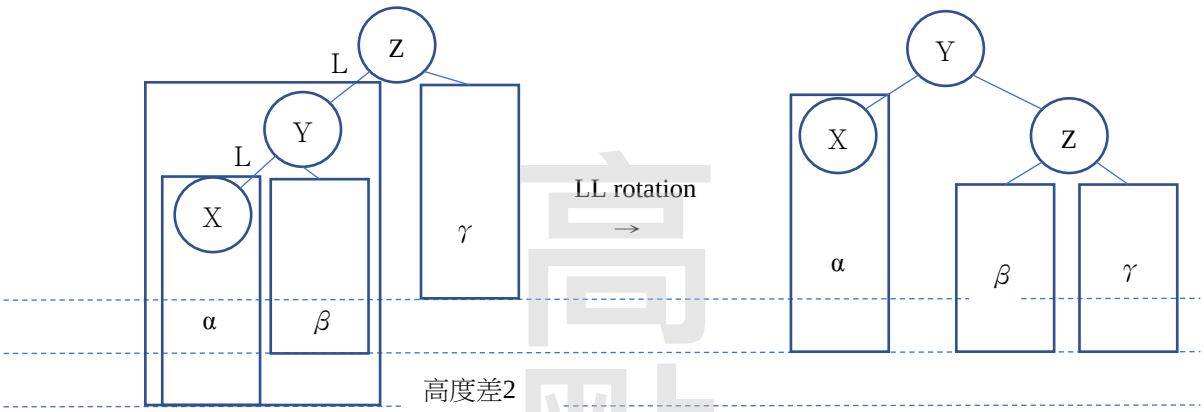
(二)以後序原理(post-order)計算節點的左、右兩 subtree 的 height，再檢查節點的 Balance Factor，遞迴方法如下：

```
bool IsAVLTree(treePtr t, out int height) {
    if (t==NULL) {     height←0; return true;   }
    else {
        if (HeightAndBalance(t→left, Hleft) && HeightAndBalance(t→right, Hright)) {
            BF←Hleft – Hright;
            height←max(Hleft,Hright)+1;
            if (BF>=-1 && BF<=1) return true;
        }
        return false;
}
```

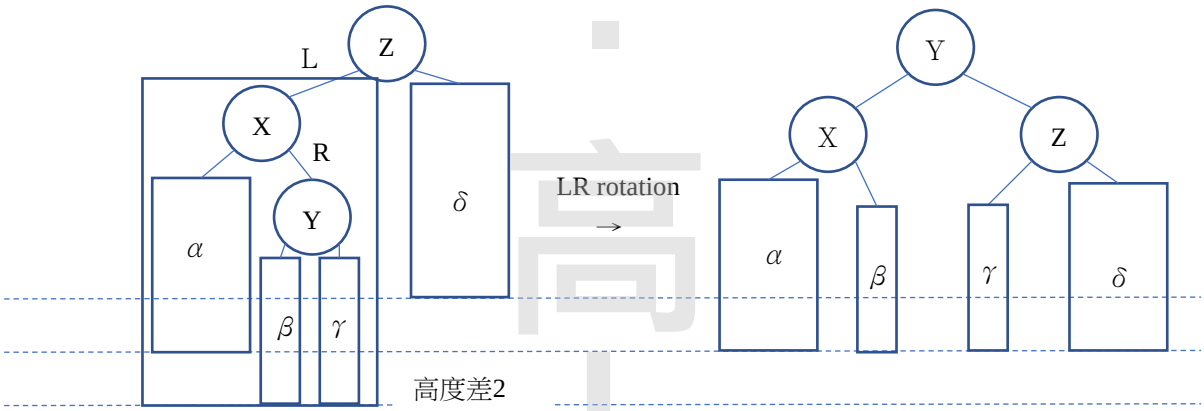
當 AVL Tree 有 n 個節點時，此演算法呼叫次數為 2n+1 次，而每次呼叫 IsAVLTree()花費 O(1)時間，故總時間為 O(n) 線性時間。

(三)是，在插入後，由插入的節點往樹根方向檢查 Balance Factor，如果發現 BF=2 或-2 時，做一次旋轉，即可恢復平衡。

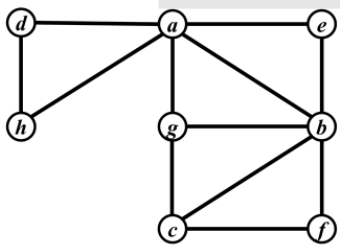
以 LL rotation 為例，旋轉前高度最大差距為 2，旋轉後高度最大差距縮小；RR rotation 的狀況與 LL rotation 對稱。




以 LR rotation 為例，旋轉前高度最大差距為 2，旋轉後高度最大差距為 1；RR rotation 的狀況與 RL rotation 對稱。




四、若我們用相鄰矩陣（Adjacency Matrix）$M$來表示圖一中的無向圖$G=(V,E)$，請考慮下面的問題：




圖一、無向圖$G=(V,E)$



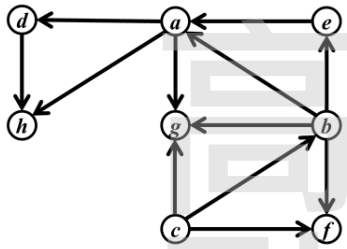
（一）對於無向圖$G=(V,E)$：（12分）

(1)請給出對應的相鄰矩陣$M$。

(2)以字母順序為考量進行深度優先搜尋（Depth-First Search, DFS），請由節點a開始，描述此深度優先搜尋所產生的深度優先樹（DF-tree）。

(二)請說明在用相鄰矩陣（Adjacency Matrix）表示的無向圖上，進行深度優先搜尋的時間複雜度，其中節點與邊的數量分別為$|V|=n$與$|E|=m$。（8分）

(三)若將圖一無向圖$G=(V,E)$中的邊給予方向成為如圖二中的有向圖（Directed Graph）$G'$：（10分）




圖二、有向圖$G'$

(1)有向圖$G'$沒有迴圈（Cycle），是一個無迴圈有向圖（Directed Acyclic Graph, DAG），所以存在節點的拓樸排序(Topological Sort)，請對$G'$給出一個拓樸排序(Topological Sort)。

(2)請給一個方法來判斷一個有向圖是否沒有迴圈。

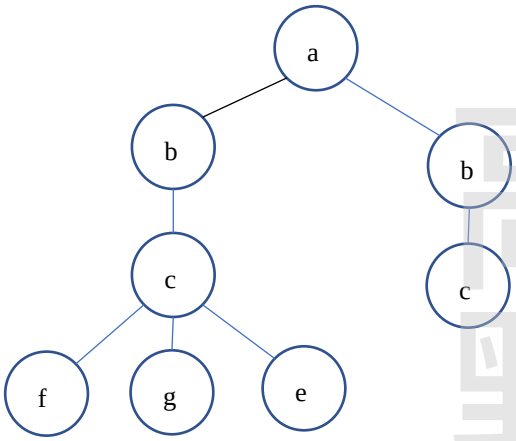
| | |
|---|---|
| **試題評析** | 此次試題大多為綜合型試題，本題為圖形部份的題目。前面考圖形表示法，接著是圖形追蹤，最後是拓樸排序問題。本題在整份試卷中，尚屬較簡單的問題。 |
| **考點命中** | 1.《資料結構》高點文化出版，王致強編著，頁8-8，8-2節要圖形表示法，鄰接矩陣。<br>2.《資料結構》高點文化出版，王致強編著，頁8-22，精選範例13。<br>3.《資料結構》高點文化出版，王致強編著，頁8-84，精選範例64。 |

**答**：

(一)(1)相鄰矩陣如下

| | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|
| a | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| b | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| c | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| d | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| e | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| f | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| g | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| h | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

(2)深度優先樹如下




(二)相鄰矩陣的無向圖上，進行深度優先搜尋的時間複雜度為$O(n^2)$，每個頂點追踪後，必須檢查距陣每一個頂點是否與目前頂點相鄰時間為O(n)，頂點總共有n個，故總時間為$nxO(n)=N(n^2)$。

(三)(1) 其中一個拓樸排序為: c, b, f, e, a, g, d, h

(2) while (G'中存在有in-degree為0的頂點) {
    任選一個in-degree=0的頂點v;
    delete 頂點 v, 以及所有 edge (v,w);
}
if (圖形頂點全被刪除) G'是DAG;
else G'不是DAG;