目錄

第一章　習作解答 …… 2
第二章　習作解答 …… 3
第三章　習作解答 …… 4
第四章　習作解答 …… 7
第五章　習作解答 …… 9
第六章　習作解答 …… 13
第七章　習作解答 …… 14
第八章　習作解答 …… 17
第九章　習作解答 …… 19
第十章　習作解答 …… 22

第一章　習作解答

一、填充解答

1. '0b1111101'、bin()、'0o175'、oct()、'0x7d'、hex()
2. 數值轉換爲整數、數值轉爲浮點數
3. 實數、虛數、Decimal()
4. (1)59、(2)564、(3)11、(4)3
5. (1)0, 1, 2, 3, 4、(2)2, 3, 4, 5、(3)5, 9, 13, 17,21
6. 中斷迴圈的執行，跳過目前的敘述，回到上一層迴圈
7. 取得輸入的資料、兩數相除的商和餘數、將字串格式化、回傳物件的型別
8. def、return
9. 基底類別、父類別、衍生類別、子類別

第二章　習作解答

1. 演算法描述解決問題的方法是以程序式的描述爲主，讓「人」一看就知道是怎麼一回事，所以表達的對象以人爲主，要能閱讀。大部分的演算法都能夠利用程式流程圖表現，不過並非所有程式流程圖都可用演算方法表示，因爲程式流程圖可包含無窮迴路，與演算法的定義相違背。

2.

Step 1.　定義函式，以名稱和5科成績爲參數。

Step 2.　函式中，輸出名稱。

Step 3.　函式中，以sum()函式來統計總分。

Step 4.　函式中，算出平均分數。

Step 5.　函式中，加入條件判斷，大於或等於60分者，印出總分和平均分數。

Step 6.　函式條件判斷中，小於60分者，印出不及格和平均分數。

Step 7.　主程式，以Tuple物件儲存名稱和5科分數。

3. 執行次數爲5次，由於每次在迴圈裡會把k值會從100000、1000、100、10、1；由於「k = 1」沒有大於條件判斷就結束while迴圈。

4.

(1) $f(n) = (n^2+1)\log n = O(n^2 \log n)$

(2) $f(n) = 8\log\log n = O(\log\log n)$

(3) $f(n) = \log n^2 = 2\log n = O(\log n)$

(4) $f(n) = 4\log\log n = O(\log\log n)$

(5) $f(n)=n/100+1000/n^2 <= n/100$（當n>=1000時）$=O(n)$

(6) $f(n) = n! = 1*2*3*4*5\cdots*n <= n*n*n*\cdots n*n <= n^n$（n>=1時） $= O(n^n)$

第三章　習作解答

1. Python的Tuple與List若以資料結構的觀點來看，皆屬於動態資料結構。不同之處在於「可變性」，產生Tuple物件就無法改變其項目，無論是新增、插入或刪除皆無法使用。List所建的物件，無論是新增、插入、刪除或清空項目皆有List物件提供的方法來配合。

   所以，若希望陣列的內容不要被任意刪除或修改就以Tuple物件來處理，若要有更彈性的操作就找List物件來幫忙。

4. (1) 假設A陣列m列×n行，則m = 1 – (–100) + 1 = 102, n = 100 – 1 + 1 = 100

   Loc(A[1, 12]) = 100 + (1 – (–100))*100*4 + (12 – 1)*4 = 40544(#)

   (2) 假設A陣列有m列×n行，則m = 10 – 5 + 1 = 6, n = 20 – (–10) + 1 = 31

   Loc(A[5, –5]) = 100 + (5 – 5)*31*4 + (–5 – (–10))*4 = 120(#)

5. 因爲陣列元素$A_{3,2}$、$A_{2,3}$的第1個索引值3 > 2，第2個索引值2 < 3，而儲存位址1110 < 1115；故可推知此陣列是以行爲主（Column-Major）的方式儲存。

$A_{3,2}$ 的位址計算如下：	$A_{2,3}$ 的位址計算如下：
=>α + (2 – 1)×m×1 + (3 – 1)×1 = 1110	=>α + (3 – 1)×m×1 + (2 – 1)×1 = 1115
=>α + m=1108 ----------(1)	=>α + 2m=1114 ----------(2)

   (1) – (2) => m = 6　　代入(1)式得a = 1102

6. 由Loc(A(3, 3)) = 121，Loc(A(6, 4)) = 159，得知陣列A的配置是以「以行爲主」的方式，所以起始位址α，單位空間1，則陣列A(1:m, 1:n)

=>α + (3 – 1)*1 + m*(3 – 1)*1 = α + 2*(1 + m) = 121 => α + 2 + 2m = 121……①	α + (6 – 1)*1 + (4 – 1)*m = α + 3m + 5 = 159 => α + 3m + 5 = 159……②

由①，②式可得α = 49，m = 35

=>Loc(A(4, 5)) = 49 + 4*35 + 3 = 192(#)

7. 首先我們知道本題所提供的資料是屬於註標表示法，則計算出實際陣列的列數及行數：

m = 5 – (–3) + 1 = 9 => 9列

n = 2 – (–4) + 1 = 7 => 7行

再代入以列爲主的位址計算公式中，可得下面的式子：

A(1, 1)的位址 = 100 + ((1 – (–3) + 1) – 1)*7*1 + ((1 – (–4) + 1) – 1)*1

= 13

8. 因爲陣列元素$A_{3,2}$、$A_{8,6}$的第1個索引值3 < 8，第2個索引值2 < 8，而儲存位址168 <344，故此陣列無法決定是以何種的方式儲存。

首先假設以行爲主（Column-Major）：第一個元素的位址爲α，元素的大小d = 4；則

$A_{3,2}$ 的位址計算如下： =>α + (2 – 1)×m×4 + (3 – 1)×4 = 168 =>α + 4m = 160 ----------(1)	$A_{8,6}$ 的位址計算如下： =>α + (6 – 1)×m×4 + (8 – 1)×4 = 344 =>α + 20m = 316 ----------(2)

(1) – (2) => 16m = 156 m = 9.75（不合理）

因此假設陣列以列爲主（Row-Major）：第一個元素的位址爲α，元素的大小d = 4，則

$A_{3,2}$ 的位址計算如下： =>α + (3 – 1)×n×4 + (2 – 1)×4 = 168 =>α + 8n = 164 ----------(1)	$A_{8,6}$ 的位址計算如下： =>α + (8 – 1)×n×4 + (6 – 1)×4 = 344 =>α + 28m = 324 ----------(2)

(1) – (2) => n = 8 代入(1)式得a = 100

則$A_{5,6}$的位址爲α + (5 – 1)×n×4 + (6 – 1)×4 = 248

9. 直接代入公式：

Loc(A(1, 2, 3)) = 100 + (1 – 1)*4*5*1 + (2 – 1)*5*1 + (3 – 1)*1 = 107

10. m = 2 – (–3) + 1 = 6　n = 3 – (–2) + 1 = 6　o = 4 – 0 + 1 = 5

Loc(A(1, 2, 3)) = 300 + (3 – 0)*6*6*1 + (2 – (–2))*6*1 + (2 – (–3))*1

= 437

第四章　習作解答

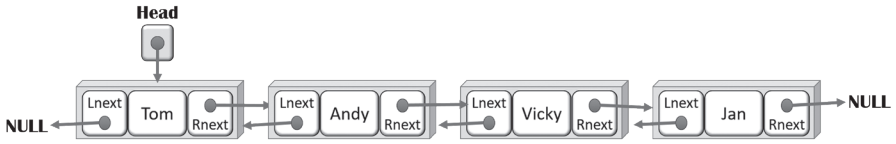
1. 單向鏈結串列（Single Link List）、環狀鏈結串列（Circular Link List）、雙向鏈結串列（Double Link List）。
2. 鏈結串列的第一個節點再附設一個「首節點」（Head Node），但是它不儲存任何資訊；有了首節點，表示從它開始就能找到第一個節點，也能藉由它儲存的「鏈結」（或指標）往下一個節點走訪。

   由於是單向鏈結串列，欲加到末端的新節點或者要刪除最後一個節點，必須從第一個節點開始，直到最後一個節點才完成加入動作；若有尾節點的指標就能提其效能。
3. 使用單向鏈結串列可以插入新的項目，有三種方式可供選擇：①從首節點插入；②從尾節點插入；③從中間的節點插入。不過，我們一定要知道，無論是那一種方式都是把鏈結的指標指向新的物件。
4. 因爲雙向鏈結串列有兩個指標分別指向節點本身的前後兩個節點，所以能夠很輕鬆的找到它前後節點，同時從串列中的任一節點也可以找到其他節點而不需經過反轉或比對節點等處理，執行速度較快。另外如果有任一節點的鏈結斷裂，可輕易的經由反方向的串列走訪，快速的完整重建鏈結。

   缺點：

   由於它有兩個鏈結，所以在加入節點或刪除節點時都得花更多的時間移動指標，且雙向串列較爲浪費空間。另外在雙向鏈結串列與單向鏈結串列的演算法中，我們知道雙向串列在加入一個節點時需改變四個指標，而刪除一個節點也要改變兩個指標。不過單向串列中加入節點，只要改變兩個指標，而刪除節點只要改變一個指標即可。

6.



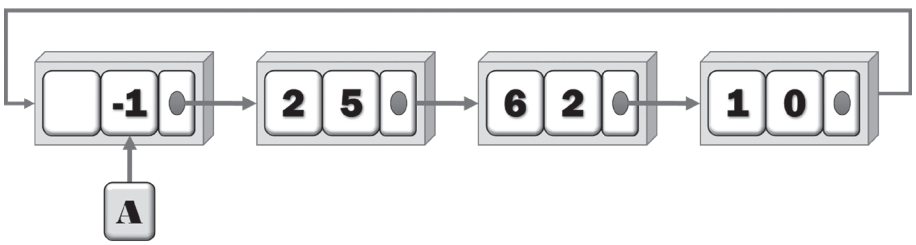
7. 關於環狀串列的特點，我們大致做出以下的結論：

優點：①回收整個串列所需時間是固定的，與長度無關。②可以從任何一個節點追蹤所有節點。

缺點：①需要多一個鏈結空間。②插入一個節點需要改變兩個鏈結。③環狀串列讀取資料比較慢，因爲必須多讀取一個鏈結指標。

8. 可加一串列首（Head Node）於環狀串列之上，此串列首爲

A = 2×5 + 6×2 + 1，可表示如下：：



第五章　習作解答

一、填充題

1.	先進後出、pop、push
2.	-A/*B+CDE、ABCD+*E/-
3.	運算子＋運算元、運算元＋運算子
4.	ISP堆疊內優先權」ICP輸入優先權
5.	由右而左、「)」（右括號）、由左而右、「(」（左括號）
6.	4, 5

二、實作與問答

1. 遞迴呼叫、副程式的呼叫、CPU的中斷處理（Interrupt Handling）、中序法轉換成後序法、堆疊計算機（Stack Computer）

2. CREATE：建立一個空堆疊

   PUSH()：從頂端推入資料，並傳回新堆疊

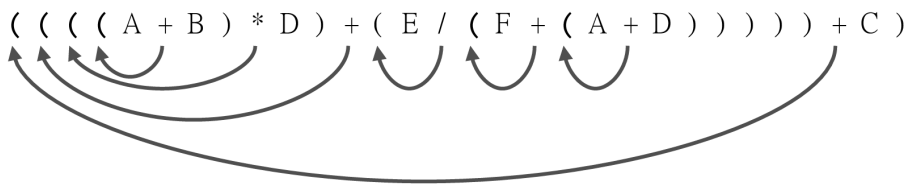
   POP()：刪除頂端資料，並傳回新堆疊

   PEEK()：查看堆疊項目，回傳其值

   IsEmpty()：判斷堆疊是否爲空堆疊，是則傳回true，不是則傳回false

4.

(A+B)* D + E / (F+A*D) + C　前序++*+ABD/E+F*ADC



(A+B)* D + E / (F+A*D) + C　後序：AB+D*EFAD*+/+C+

5.

-A*/+BC-DEF　前序式：A-B+C/(D-E)*F

AB*CD+E/-　後序式：A*B-C+D/E

6. A/B+(C+D)*E-A*C轉爲前序式

讀入字元	堆疊內容	輸出
C	Empty	C
*	*	C
A	*	AC
-	-	*AC
E	-	E*AC
*	*-	E*AC

讀入字元	堆疊內容	輸出
)	* -)	E*AC
D	* -)	DE*AC
+	* -) +	DE*AC
(	* -	+CDE*AC
+	+ -	*+CDE*AC
B	+-	B*+CDE*AC
/	/+-	/B*+CDE*AC
A	/+-	AB*+CDE*AC
None	Empty	-+/AB*+CDE*AC

A/B+(C+D)*E-A*C轉爲後序式

讀入字元	堆疊內容	輸出
A	Empty	A
/	/	A
B	/	AB
+	+	AB/
(	+(	AB/
C	+(	AB/C
+	+(+	AB/C
D	+(+	AB/CD
)	+	AB/CD+
*	+*	AB/CD+
E	+*	AB/CD+E
-	+-	AB/CD+E*
A	+-	AB/CD+E*A
*	*+-	AB/CD+E*A

讀入字元	堆疊內容	輸出
C	*+-	AB*+CDE*AC*+-
None	Empty	AB*+CDE*AC*+-

7. 所謂「尾歸遞迴」（Tail Recursion）就是程式的最後一個指令爲遞迴呼叫，因爲每次呼叫後，再回到前一次呼叫的第一行指令就是return，所以不需要再進行任何計算工作，因此也不必保存原來的環境資訊（如參數儲存、控制權轉移）。例如N!

第六章　習作解答

1. CPU的排程，播放器待播放的曲目，列表機待列印的文件、I/O設備的配置。

3. front指標會指向第一個元素，而rear指標則指向最後一個元素。新增元素時rear指標會隨著新增元素來變更位置。當佇列前端第一個元素被刪除時，指標front原本會由第一個位置而指向下一個位置。所以，指標front會隨著前端元素的移除向後方移動。

4. 實作佇列，新增（enqueue）、刪除（dequeue）、佇列是否已滿（isFull）、佇列是否空的（isEmpty）、取得佇列前端元素（front）。

6. 環狀佇列的操作

	front指標	rear指標
①	4	5
②	4	7
③	4	0
④	5	0
⑤	5	2

第七章　習作解答

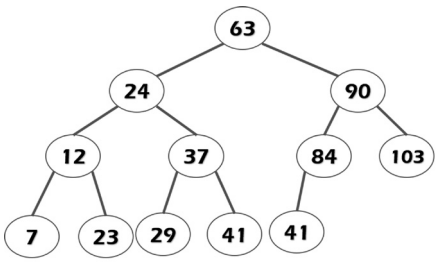
一、填充題

1.	A	A、B、C	D、E、F	兄弟節點	堂兄弟	C、A	B、C、D、E、F、G、H
2.	(B)因爲環狀串列會造成循環現象，不符合樹的定義						
3.	有序	2					
4.	完滿二元樹		歪斜樹			5.	(A)
6.	空集合	0 ≦ d ≦ 2		沒有次序關係			
7.	葉節點	一個子節點		左、右皆有子節點			

二、實作與問答

1. 不是，因爲會造成無出口的迴圈。

2. 二元搜尋樹



3. 提示：各位可先行假設n是節點總數，是分支度等於1的節點數，可得$n = n_0 + n_1 + n_2$，再行證明。

4. 利用數學歸納法證明：

   - 當i=1時，只有樹根一個節點，所以$2^{i-1} = 2^0 = 1$成立。
   - 假設對於j，且$1 \leqq j \leqq i$，階度爲j的最多點數2^{j-1}個成立，則在j = i階度上的節點最多爲2^{i-1}個。

   當j = i + 1時，因爲二元樹中每一個節點的分支度都不大於2，因此在階

度j = i + 1時的最多節點數 ≤ $2*2^{i-1} = 2^i$，由此得證。

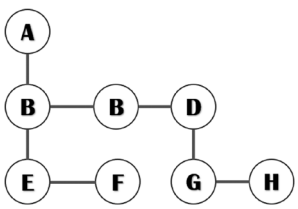
5. 中序：A/B**C+D*E-A*C；後序：ABC**/DE*+AC*-；前序：-+/A**BC*DE*AC

6. 中序：A/B↑C*D-E、前序：-*/A↑BCDE、後序：ABC↑/D*E-

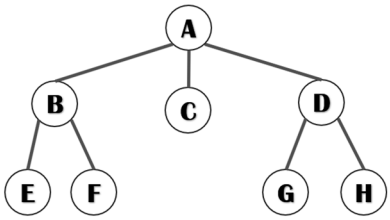
7. 樹林的走訪方式則由樹的走訪衍生過來，步驟如下：

中序走訪（InOrder traversal）	①如果樹林爲空，則直接返回。 ②以中序走訪第一棵樹的子樹群。 ③中序走訪樹林中第一棵樹的樹根。 ④依中序法走訪樹林中其它的樹。
前序走訪（Preorder traversal）	①如果樹林爲空，則直接返回。 ②走訪樹林中第一棵樹的樹根。 ③以前序走訪第一棵樹的子樹群。 ④以前序法走訪樹林中其它的樹。
後序走訪（Postorder traversal）	①如果樹林爲空，則直接返回。 ②以後序走訪第一棵樹的子樹。 ③以後序法走訪樹林中其它的樹。 ④走訪樹林中第一棵樹的樹根。
樹林走訪的結果如下：	①中序走訪：EBCDAGHFI ②前序走訪：ABECDFGHI ③後序走訪：EBCDGHIFA

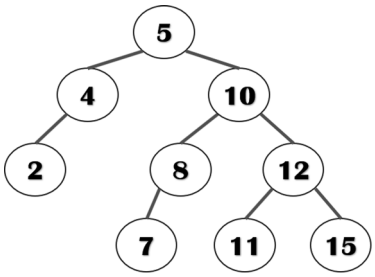
8. 這就是樹化爲二元樹的反向步驟，方法也很簡單。首先是逆時針旋轉45度，如下圖所示：



另外(ABE)(DG)左子樹代表父子關係；(BCD)(EF)(GH)右子樹代表兄弟關係



9.

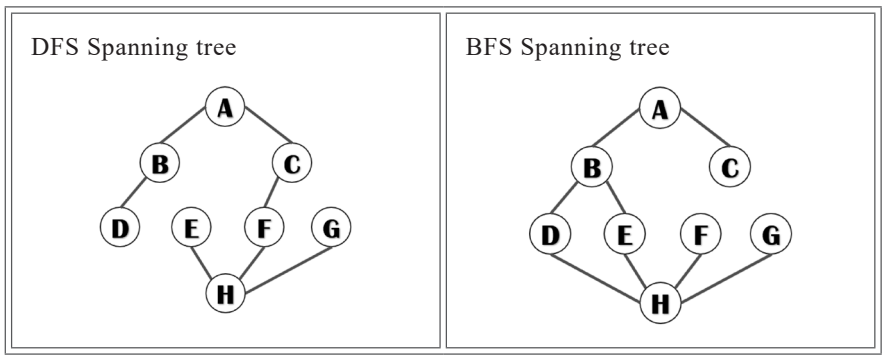


10.如果二元樹的高度爲h，樹的節點數爲2h – 1，h>=0，則我們稱此樹爲「完滿二元樹」（full binary tree）。

如果二元樹的深度爲h，所含的節點數小於2h – 1，但其節點的編號方式如同深度爲h的完滿二元樹一般，從左到右，由上到下的順序一一對應結合。

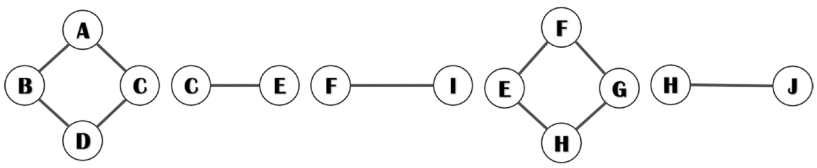
第八章　習作解答

1.



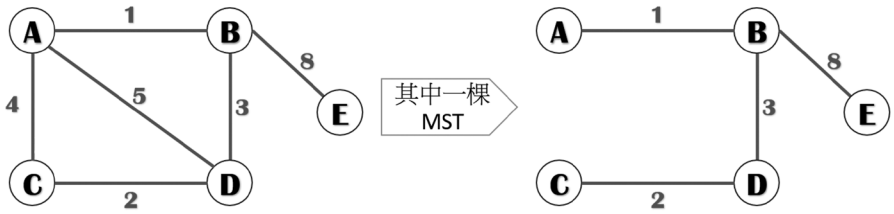
2. (3)　3.(3)、(5)、(8)

4. 對於一個頂點V，如果將V上所連接的邊都去除所產生的G'，如果G'最少有兩個連通單元，則稱此頂點V爲所謂的「連結點」（Articulation Point）。而一個沒有連結點的圖形，就是「雙連通圖形」（Biconnected Graph）。而這個有4個連結點C、E、F、H，因此並不是「雙連通圖形」。而此圖的連通單元，有下列五種：



5. DFS：先深後廣的走訪順序爲：頂點A、頂點B、頂點C、頂點D、頂點E　BFS：頂點A、頂點B、頂點E、頂點C、頂點D

6. 圖G中的最大加權邊是8，但仍然在MST中。



7.

$$A^0=\begin{bmatrix}0&4&11\\6&0&2\\3&\infty&0\end{bmatrix}\ A^1=\begin{bmatrix}0&4&11\\6&0&2\\3&7&0\end{bmatrix}$$

$$A^2=\begin{bmatrix}0&4&6\\6&0&2\\3&7&0\end{bmatrix}\ A^3=\begin{bmatrix}0&4&6\\5&0&2\\3&7&0\end{bmatrix}$$

8. 相鄰矩陣法、相鄰串列法、相鄰多元串列法、索引表格法。

9. 答：一個圖形G = (V, E)，存在某一頂點v?V，我們希望從v開始，經由此節點相鄰的節點而去拜訪G中其它節點，這稱之爲「圖形追蹤」。

10.答：

Prim演算法又稱P氏法，對一個加權圖形G = (V, E)，設V={1,2,……n}，假設U={1}，也就是說，U及V是兩個頂點的集合。然後從U-V差集所產生的集合中找出一個頂點x，該頂點x能與U集合中的某點形成最小成本的邊，且不會造成迴圈。然後將頂點x加入U集合中，反覆執行同樣的步驟，一直到U集合等於V集合（即U=V）爲止。

第九章　習作解答

一、

題目	1	2	3	4	5	6	7	8	9
答案	D	D	A	B	B、C	B	C	B	D

二、實作與問答

1. 將下列資料「85、34、11、73、65、42、126」繪製出氣泡法由大而小的交換方式，並利用程式碼輸出其過程。

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	說明
原始資料	85	34	11	73	65	42	126	
A[0]> A[1]	85	34						不交換
A[1]> A[2]		34	11					不交換
A[2]> A[3]			73	11				交換
A[3]> A[4]				65	11			交換
A[4]> A[5]					42	11		交換
A[5]> A[6]						126	11	交換
第一回合得	85	34	73	65	42	126	11	

繼續第二回合的資料交換

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	說明
第二回合	85	34	73	65	42	126	11	
A[0]> A[1]	85	34						不交換
A[1]> A[2]		73	34					交換
A[2]> A[3]			65	34				交換
A[3]> A[4]				42	34			交換
A[4]> A[5]					126	34		交換
第二回合	85	73	65	42	126	34	11	

繼續第三回合的資料交換

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	說明
第三回合	85	73	65	42	126	34	11	
A[0]> A[1]	85	73						不交換
A[1]> A[2]		73	65					不交換
A[2]> A[3]			65	42				不交換
A[3]> A[4]				126	42			交換
第三回合得	85	73	65	126	42	34	11	

繼續第四回合的資料交換

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	說明
第四回合	85	73	65	126	42	34	11	
A[0]> A[1]	85	73						不交換
A[1]> A[2]		73	65					不交換
A[2]> A[3]			126	65				交換
第四回合得	85	73	126	65	42	34	11	

繼續第五回合的資料交換

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	說明
第五回合	85	73	126	65	42	34	11	
A[0]> A[1]	85	73						不交換
A[1]> A[2]		126	73					交換
第五回合得	85	126	73	65	42	34	11	

繼續第六回合的資料交換

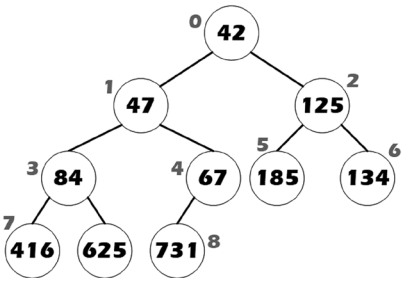
	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	說明
第六回合	85	126	73	65	42	34	11	
A[0]> A[1]	126	85						交換
第六回合得	126	85	73	65	42	34	11	完成排序

3. 選擇排序法-找出最大者與第一個元素交換

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	說明
原始資料	185	625	134	47	731	125	42	416	
第一回合	731	625	134	47	185	125	42	426	731、185互換
第二回合	731	625	134	47	185	125	42	426	625未換
第三回合	731	625	426	47	185	125	42	134	426、134互換
第四回合	731	625	426	185	47	125	42	134	185、47互換
第五回合	731	625	426	185	134	125	42	47	134、47互換
第六回合	731	625	426	185	134	125	47	42	互換
完成排序	731	625	46	185	134	125	47	42	完成排序

4. 解答：比較式排序是每次以整個鍵值作爲比較的依據，例如Bubble Sort。而分配式排序每次比較僅比較鍵值的某一位數，例如Radix Sort。

5. 堆積樹

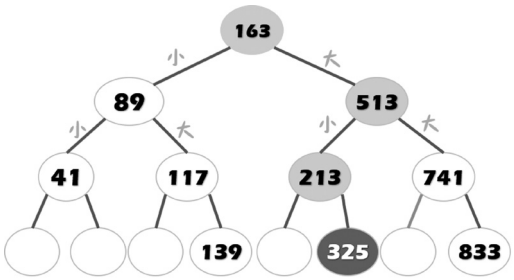


APPENDIX A

第十章　習作解答

2. 查找過程如下：

```
mid = (0 + 9) // 2 = 4, 爲163
325 > 163, 向右
mid = (4 + 1) + 9 // 2 = 7, 爲513
325 < 513, 向左
mid = 5 + (7 - 1) // 2 = 5, 爲213
325 > 213, 向右
mid = (5 + 1) + 6 // 2 = 6, 找到key「325」
```



3.將搜尋數列表示如下：

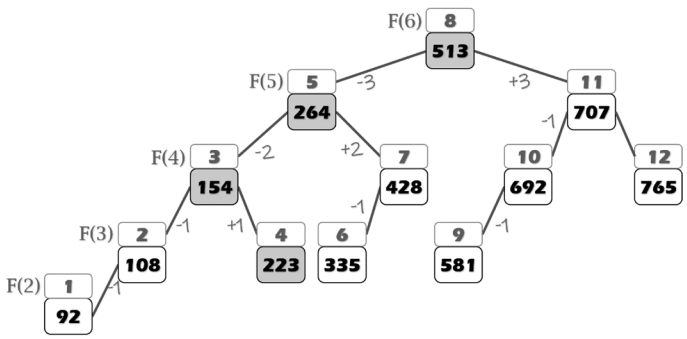
41	92	117	125	223	264	325	478	513	692	787
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]

使用「內插法」的搜尋過程如下：

次數	low	high	mid	key與A[mid]比較	範圍
1	0	10	$0 + \frac{513 - 41}{787 - 41} * 10 = 6$	513 > 325	向右
2	6 + 1 = 7	10	$7 + \frac{513 - 478}{787 - 478} * 3 = 7$	513 > 478	向右

次數	low	high	mid	key與A[mid]比較	範圍
3	7 + 1 = 8	10	$8 + \frac{513 - 513}{787 - 513} * 2 = 8$	mid = low = 513	找到

4. 費氏樹



「N = 11」所以費氏樹「F(k) – 1 = 12, F(k) = 13」，得「k = 7」以費氏樹搜尋key「223」的過程如下：

次數	樹根(r)	子樹(s)	差值(d)	比較	範圍
開始	F(7 – 1) = 8	F(7 – 2) = F(5) = 5	F(7 – 3) = F(3) = 3	223 < 152	向左
2	r – d = 8 – 3 = 5	s = d = 3	s – d = 5 – 3 = 2	223 < 64	向左
3	r – d = 5 – 2 = 3	s = d = 2	s – d = 3 – 2 = 1	223 > 154	向右
4	r + d = 3 + 1 = 4	s – d = 2 – 1 = 1	d – s = 1 – 1 = 0	223 = 223	找到

5. 令雜湊函數爲h(key)=key mod B，其中B=11爲一質數，這個函數的計算結果介於0~10之間（即0,1,2,3,4,5,6,7,8,,9,10這11種可能的餘數），則

```
h(345) = 4、h(348) = 7、h(80) = 3、h(119) = 9、
h(83) = 6、h(89) = 1、h(297) = 0
```

索引	0	1	2	3	4	5	6	7	8	9	10
儲存值	297	89		80	345		83	348		119	

6. 利用移動折疊法的原理可得到：

   => 743+280+321=1344、=> 1344即爲索引位址

7. 平方測探法

```
h(431) = 431 % 11 = 2 → 3
h(597) = 597 % 11 = 3 → 4
h(343) = 343 % 11 = 2 → 6 → 0 → 7 → 5
h(385) = 385 % 11 = 0 → 1
```

雜湊表如下：

索引	0	1	2	3	4	5	6	7	8	9	10	11
鍵值	583	385	365	431	597	343	534	128	459	680		