《資料結構》

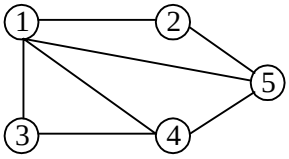
試題評析

本次高考資料結構試題相當平易，考生要拿到高分應不難，試題大多集中在樹、圖形與遞迴相關的問題上。第一題是「圖形表示法」相關問題，應可以簡單拿到分數。第二題則是「霍夫曼編碼樹」的問題，取分也不會困難。第三題是「測驗遞迴程式」的能力，所考的題目都是一般書上常見的範例，因此也十分容易答題。第四題則是「堆積相關表示法」，與「建立堆積」的問題，也都是基本問題，時間複雜度的計算要小心即可。

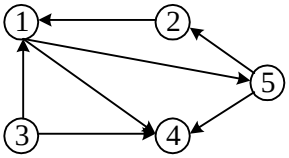
綜觀今年資料結構試題，一般考生要拿到八十分應是輕而易舉；程度好的考生拿到九十分，甚至滿分也不是不可能。

一、圖（graph）的表示法（Graph Representation）

(一)以下面的無向圖（undirected graph）為例，說明圖的鄰接串列（adjacency list）表示法。（10分）



(二)以下面的有向圖（directed graph）為例，說明圖的鄰接矩陣（adjaceny matrix）表示法。（5分）

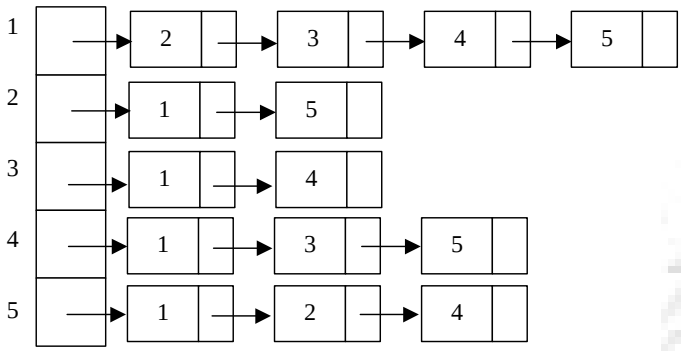


(三)給一n個節點（vertex）的有向圖G的鄰接矩陣，請問計算圖G的一個節點的出分支度（out degree）的時間複雜度為何？（5分）

(四)給一n個節點（vertex）的有向圖G的鄰接矩陣，請簡述判斷圖G是否連通（connected）的演算法。（5分）

答：

(一)



(二)

	1	2	3	4	5
1	0	0	0	1	1
2	1	0	0	0	0
3	1	0	0	1	0
4	0	0	0	0	0
5	0	1	0	1	0

(三) O(n)

(四) $O(n^2)$，使用 DFS 或 BFS 的追蹤法，來檢查連通性。

二、霍夫曼碼（Huffman code）是一種依照字母出現的頻率決定編碼的不定長二進位編碼法（variable-length binary code）。

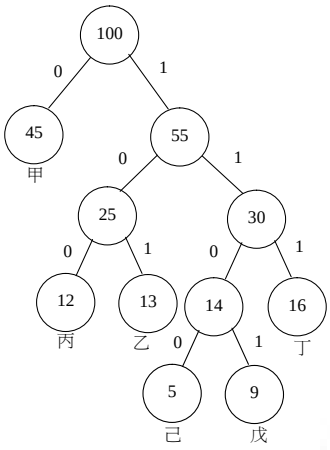
(一)說明霍夫曼碼的編碼與解碼原理。（10分）

(二)假設字母集為{甲、乙、丙、丁、戊、己}，個別字母出現頻率如下表。請填寫每個字母的霍夫曼碼。（15分）

字母	甲	乙	丙	丁	戊	己
出現頻率	45	13	12	16	9	5
霍夫曼碼						

答：

(一)編碼：建立霍夫曼編碼樹，每次挑選出最小的兩項，合併後將頻率相加，直到全部合併爲止，再分配0與1。



解碼：使用編碼樹由樹根開始根據所掃瞄到的霍夫曼碼位元，直到樹葉爲止，即可解出字母。

(二)

字母	甲	乙	丙	丁	戊	己
出現頻率	45	13	12	16	9	5
霍夫曼碼	0	101	100	111	1101	1100

三、遞迴演算法（recursive algorithm）

（一）令A為N個數的整數陣列（Integer array）。請用虛擬碼（Pseudo Code）描述求陣列A中最大值的遞迴演算法。（5分）

（二）令A為N個數的整數陣列（Integer array）。假設A中的數字已經由小到大排列好。請用儘量接近程式語言的虛擬碼（Pseudo Code）描述搜尋整數X是否存在陣列A中的二元搜尋（Binary Search）的遞迴演算法（recursive algorithm）。請說明此一搜尋法的時間複雜度。（10分）

（三）請用儘量接近程式語言的虛擬碼（Pseudo Code）描述計算費氏數列（Fibonacci numbers）第N項的遞迴演算法。請問該遞迴演算法的時間複雜度（time complexity）是否為多項式時間（polynomial time）複雜度？（10分）

答：

(一)
```
int max(int A[], int N)
{   if (N==1) return A[0];
    else
    {   int m=max(A,N-1);
        if (A[N-1]>m) return A[N-1];
        else return m;
    }
}
```

(二)
```
int BinSearch(int A[], int left, int right, int X)
{   if (left>right) return -1;   // 未找到
    else
    {   int mid=(left+right)/2;
        if (A[mid]==X) return mid;
        else if (A[mid]>X) return BinSearch(A, left, mid-1, X);
        else return BinSearch(A, mid+1, right, X);
    }
}
```

時間複雜度爲 O(logn)。

(三)
```
int Fib(int N)
{   if (N<=1) return N;
    else return Fib(N-1)+Fib(N-2);
}
```

時間複雜度爲O(2n)，不是多項式時間。

四、堆積排序（Heap Sort）

（一）堆積排序將堆積樹（heap tree）用一個陣列（array）A儲存。陣列的指標（index）從1到N。請說明堆積樹的根（root）在陣列的中的位置。請說明陣列（array）A第i個位置A[i]所儲存的堆積樹節點的左子節點（left child）、右子節點（right child）、以及父節點（parent）各自在陣列A中的位置。（5分）

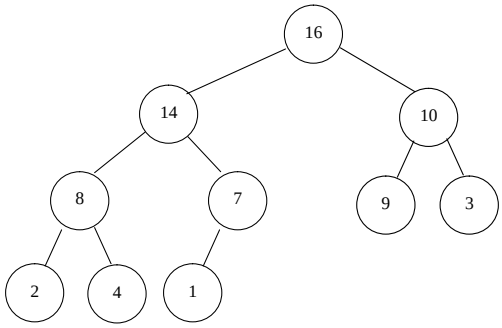
（二）在Max-堆積樹中，除了根節點（root）外，每一個節點所儲存的數小於或等於其父節點所儲存的數。假設陣列A儲存一個十個節點的Max-堆積樹。陣列A中的數字從第一個位置到第10個位置所存數字依序為16, 14,10, 8, 7, 9, 3, 2, 4, 1。請畫出陣列A所儲存的堆積樹以及各節點所儲存的數。（5分）

（三）請以儘量接近程式語言虛凝碼描述如何將一個不符合Max-堆積樹性質的陣列轉換成符合Max-堆積樹性質的陣列。請分析你的演算法的時間複雜度。（15分）

答：

(一) A[i] 的左子節點爲 A[2*i]; A[i] 的右子節點爲 A[2*i+1]; A[i] 的父節點爲 A[i/2]。

(二)



(三)

```
void sift(int a[], int i, int n)
{   // 由第 i 個位置開始調整，但調整不超過第 n 項
    int temp, j;
    temp=a[i]; j=2*i;
    while (j<=n)
    {     if (j<n && a[j]<a[j+1]) j++;
          if (temp>a[j]) break;
          a[i]=a[j];
          i=j;    j=2*i;
    }
    a[i]=temp;
 }
```

主程序

```
void ConstructHeap (int a[], int n)
{ int i;
      for (i=n/2; i>=1; i--)
            sift(a,i,n);
}
```

時間複雜度:

建立 heap 時，worst case 所需之 sifts 次數，可以取 $n=2^k-1$ 項資來計算

$$T(n)=(k-1)\times 1+(k-2)\times 2+(k-3)\times 2^2+\ldots+1\times 2^{k-2} \quad (1)$$

$$T(n)\times 2=(k-1)\times 2+(k-2)\times 2^2+(k-3)\times 2^3+\ldots+1\times 2^{k-1} \quad (2)$$

(2)-(1)

$$T(n)=-k+1+2+2^2+\ldots+2^{k-1}+2^{k-1}=n-k=n-\log_2(n+1)=O(n)$$