《資料結構》

試題評析

第一題：解釋名詞，屬於基本問題，應可簡單拿分。
第二題：二元搜尋樹的基本運算，其中第(二)小題須熟稔刪除的原理，並且對程式（演算法）要了解其做法，才能取得分數。
第三題：圖形表示法的應用問題，並且也考到Dijkstras 單起點最短路徑問題，答題應當不難。
第四題：Shell 排序法稍微冷門的部份，準備完整的考生回答亦不困難。
第五題：應用資料結構表示棋類遊戲的問題，此為近二年常見的考題類型，未考到人工智慧相關內容，能了解題意想出表示方法可簡單處理。
本份試題預測拿到65分是基本的，程度佳、準備完整的應試者應可拿到80分或以上的成績。

一、解釋下列名詞並舉例說明：（每小題5分，共25分）
(一)演算法（algorithm）
(二)時間複雜度（time complexity）
(三)遞迴式的解決問題方法（recursive solution）
(四)雙向佇列（Deque）
(五)最小成本生成樹（minimum cost spanning tree）

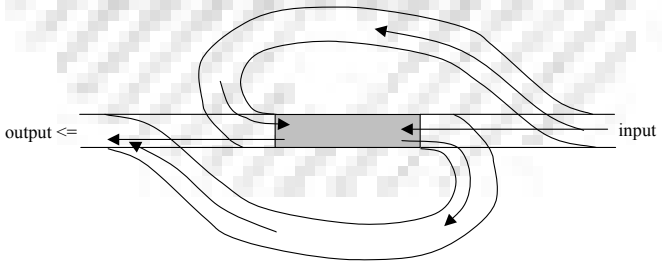
答：

(一)演算法：演算法的定義：為了完成某一特定工作，而設計出的一組有限、有序的指令集合。例如：選擇排序法。

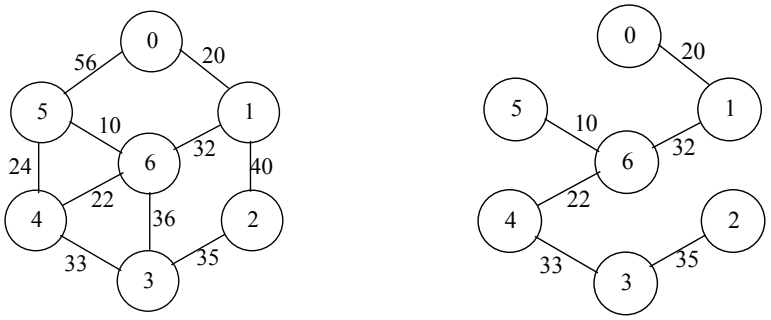
(二)時間複雜度：採用漸近式表示法(Asymptotic Notations)，不計算實際執行指令的個數，而改以計算程式反覆被執行的次數等級，來表示所需時間的成長率現象。例如：f(n)=O(g(n))。

(三)遞迴式的解決問題方法：一個程序(函數)解決問題時，會呼叫自己的比較簡單版本。例如：n的階層。

(四)雙向佇列：一種線性佇列，兩端皆可以加入資料也都可以刪除資料。例如下圖:



(五)最小成本生成樹：一個圖形的生成樹中，成本最低的一棵。例如:



二、請用二元樹（binary tree）針對10筆資料：「陳、劉、王、蘇、高、胡、蔡、何、簡、莊」設計出以鏈結（link）表示的二元樹資料結構，10筆資料的排序方式可自行決定（例如，依據筆劃數、注音符號、拼音或其他）。（每小題5分，共25分）

(一)請用任意程式語言寫出插入（insert）一個節點的演算法。

(二)請用任意程式語言寫出刪除（delete）一個節點的演算法。

(三)請用任意程式語言寫出中序（inorder）尋訪的演算法。

(四)請將「陳、劉、王、蘇、高、胡、蔡、何、簡、莊」及你決定並明確寫出的排序方式，用插入演算法逐一插入二元樹，請畫出最後的二元樹。

(五)請分析二元樹搜尋（searching）的O( )時間複雜度。

答：

(一)

```
1.  void BST_Ins(datatype x, treeptr &t);
2.  {
3.      if (t == NULL)
4.      {
5.          t=(treeptr)malloc(sizeof(struct node));
6.          t→data = x;
7.          t→lchild = NULL;
8.          t→rchild = NULL;
9.      }
10.     else if (x < t→data) BST_ins(x, t→lchild);
11.     else if (x > t→data) BST_ins(x, t→rchild)
12. }
```

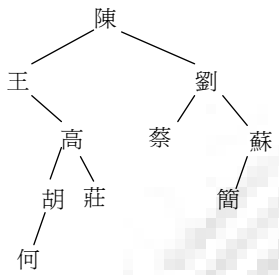
(二)

```
1.  void BST_Delete(datatype x, treeptr &t)
2.  {
3.      if (t==NULL)    NotFound();
4.      else if (x<t→data) BST_Delete(x,t→left);
5.      else if (x>t→data) BST_Delete(x,t→right);
6.      else
7.      {
8.          treeptr q=t;
9.          if (t→left==NULL) t=t→right;
10.         else if (t→right==NULL) t=t→left;
11.         else
12.         {
13.             treeptr p, q;
14.             p=t; q=p→right;
15.             while (q→left!=NULL)
16.             {
17.                 p=q;    q=q→left;
18.             }
19.             t→data=q→data;
20.             p→left=q→right;
21.         }
22.         free(q);
23.     }
24. }
```

(三)

```
1.    void inorder(TreePtr t)
2.    {
3.        if (t!=NULL)
4.        {    inorder(t->lchild);
5.             printf("%d", t->data);
6.             inorder(t->rchild)
7.        }
8.    };
```

(四)按筆劃數排序



(五)

若原來的二元搜尋樹有 n 個節點，插入一項新資料的時間複雜度爲 O(1)～O(n)。
worst case 時間爲 O(n)，可能發生在 skewed binary tree 時。average case 爲 O(logn)。

三、考慮某地區的地圖，地圖上有n個城市，城市之間共有m條相通的公路，每條公路有一個長度（例如，10公里）。某人經常需從城市S出發，開車前往另一城市T送貨，請你設計一個軟體系統的資料結構與演算法，幫忙找出路程最短的建議路徑與該路徑的總長度。（每小題5分，共15分）

(一)請設計一資料結構表示出地圖之n個城市、m條公路及公路長度。

(二)依據你設計的資料結構，寫出Dijkstra演算法，找出路程最短的建議路徑與該路徑的總長度，並舉例說明。

(三)分析Dijkstra的時間複雜度。

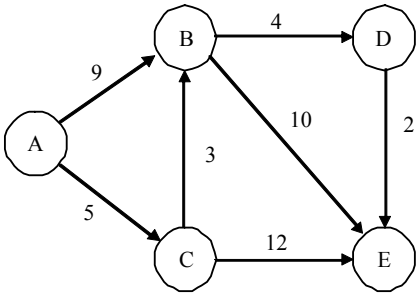
答：

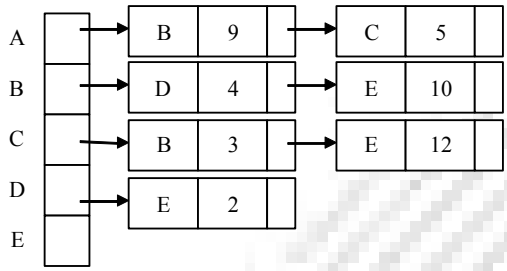
(一)n個城市，建立n 個鏈結串列；m條公路，每條公路建一個鏈串列節點。

節點結構:

```
struct node {
   char city[20];   // 城市名稱
   int length;      // 公路長度
   struct node * link;
};
struct node * header[n];
```

以下圖爲例:

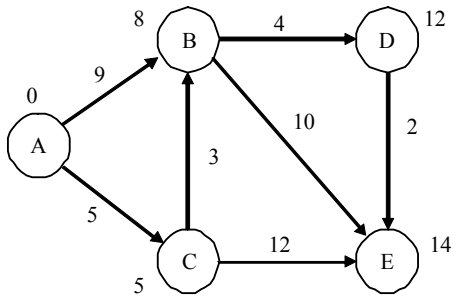




(二)

※ Dijkstra's Algorithm

```
1. procedure shortestpath(v, cost, dist, n, pred);
2. beign
3.     for i ← 1 to n do begin
4.         s[i] ← false;
5.         dist[i] ← cost[v,i];
6.         if cost[v,i]≠∞ then pred[i] ← v
7.         else pred[i] ← 0;
8.     end;
9.     s[v] ← true;    dist[v] ← 0;    prev[v] ← 0;
10.            num ← 2;
11.            while num<n do begin
12.               // choose u with minimum dist[u] and s[u]=false }
13.               u ← choose(dist, n);
14.               s[u] ← true;
15.               num ← num + 1;
16.               for each w adjacent from u do
17.                  if not s[w] then
18.                     if dist[u]+cost[u,w]<dist[w] then begin
19.                        pred[w] ← u;
20.                        dist[w] ← dist[u]+cost[u,w];
21.                     end;
22.            end;
23.        end;
```



(三)

時間複雜度

3~8 行: O(|V|)

13 行:O(|V|log|V|) // 使用 Fibonacci Heaps

16~21行:O(|V|+|E|) // 使用 鄰接串列

總時間爲：O(|E|+|V|log|V|)

四、給予資料：3, 1, 5, 7, 15, 13, 9, 11，

(一)請寫出Shell排序演算法。(15分)

(二)並用Shell排序法，將資料排成由大到小排列，請務必將每一步驟詳細畫出並詳細說明。(10分)

答：

(一)

```
void d_sort(int d, int n)
{ int   i,k,t;
   for (i=d+1; i<=n; i++)
   {          t=a[i]; k=i-d;
              while (k>0 && a[k]>t)
              {     a[k+d]=a[k];
                    k-=d;
              }
              a[k+d]=t
   }
}

void ShellSort(distset d[], int n)
{ int i=0;
   do {       i=i+1;
               d_sort(d[i],n);
   } while (d[i]>1);
}
```

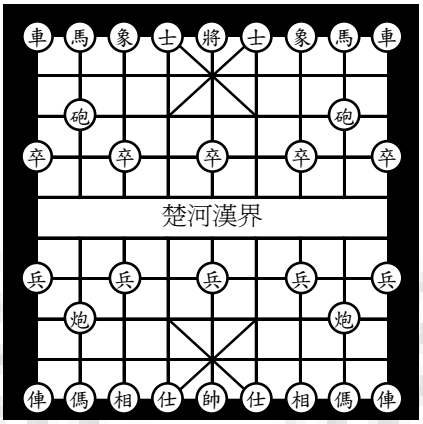
(二)

initial :	3	1	5	7	15	13	9	11
pass 1(d=4):	15	13	9	11	3	1	5	7
pass 2(d=2):	15	13	9	11	5	7	3	1
pass 3(d=1):	15	13	11	9	7	5	3	1

五、考慮設計中式象棋（如圖）電腦程式系統：（每小題5分，共10分）

（一）請設計一資料結構使能隨時表示出棋盤現狀（current state），包含所有棋子的位置、有那些棋子在棋盤上。

（二）寫出一演算法能產生「象」或「相」在任意位置之下一步可前往且合規則的所有位置（next feasible positions），注意，務必考慮其他棋子阻礙的因素。「象」或「相」的移動規則：(1)田字形的對角移動；(2)田字正中央有棋子時，不能移動前往。



答：

(一)使用一個二維陣列記錄棋盤上的狀態：

```
int board[10][9];   // 0:空的位置(沒有棋子)
                    // 1:將, 2,3:士, 4,5:象, 6,7:車, 8,9:馬, 10,11:包, 12~16:卒(有5隻卒)
                    // 17:帥, 18,19:仕, 20,21:相, 22,23:俥, 24,25:傌, 26,27:炮, 28~32:兵(有5隻兵)
int row[33],col[33];    // 記錄每個棋子所在的位置, row值等於 -1 代表該棋子已被吃
```

(二)使用一個二維陣列記錄象可以移動的相對位置

```
int move[4][2]={{-2,-2},{-2,2},{2,2},{2,-2}};
```