《資料結構》

試題評析	今年檢事官資料結構的命題十分靈活，以測驗應考人員是否能活用所學過的資料結構，所考範圍皆爲常見的資料結構，主要集中在 AVL-tree、Binary Search Tree、Heap、Disjoint Set 的 trees 表示法，以及運算式的處理。雖然涵蓋範圍是一些基本的資料結構，但應考者未必習慣此類命題方式，臨場反應很重要，因此今年此一科目的分數可能會偏低，預測可能會在 40~60 之間。

一、考慮某種資源分享系統，該資源可供使用時間共分成 n 個時段，依序編號為 1、2、…、n。每一個使用者只能使用一個時段，而每一個時段最多只能有一個使用者。使用者要預約一個時段時，需透過指令 reserve(i, j)來完成，這裡 i≤j 代表使用者希望使用時段的範圍。若這個範圍的時段都已經被預約了，這指令會回傳 0 代表預約失敗，反之若這個範圍有空的時段，這指令會回傳一介於 i、j 間的整數 k，代表使用者將可以使用時段 k。

一個簡單的作法為用一個一維陣列 a[1.. n]來表示預約的狀況，a[i]=1 代表時段 i 已經被預約，a[i]＝0 代表時段 i 未被預約。實做指令 reserve(i, j)時，只需一一檢查 a[i]到 a[j]是否有值為 0 即可。因此這個指令的執行時間為 0(j－i)，在最差的狀況下這指令的執行時間可高達 0(n)。事實上有在最差狀況下執行時間為 0(log n)或更好的作法。

請(一)設計一個能表示整個預約狀況的資料結構，與(二)寫出一實做 reserve(i, j)的演算法，使得在最差情況下 reserve(i, j)的執行時間為 0(log n)或更好。(20 分)

【擬答】

一、使用 AVL-tree 來記錄尚未被預約的時段，將每一個未被預約的時段 k，分別建立一個節點 k，並將其記錄在 AVL-tree 中。一開始時，因爲時段 1~n 都未被預約，故建立 n 個節點分別記錄 1~n，並建立起一棵 AVL-tree。

要進行 reserve(i,j) 時，只要在 AVL-tree 中找到一個節點 k(i≦k≦j)，就表示有符合要求的時段被找到，然後將節點 k 自 AVL-tree 中刪除即可。

(一)

```
int reserve(treeptr tree, int i, int j)
{    if (tree==NULL) return 0;
     else if (tree->timeperiod < i) return reserve(tree->rightchild, i, j);
     else if (tree->timeperiod > j) return reserve(tree->leftchild, i, j);
     else
     {     int t=tree->timeperiod;
           deleteAVLtree(tree);
           return t;
     }
}
```

二、二元搜尋樹(binary search tree)是一種基本的資料結構，在實做上有時需將兩個二元搜尋樹 T_1、T_2 合併成一個新的二元搜尋樹，一個單純的作法是：先建一個新的且為空的二元搜尋樹 T，然後將 T_1、T_2 裡的數字一個一個依任意次序拿出，再插入(insert)到 T 裡面。這個作法的缺點是，在最差的情況下，運算時間會高達 $O(n^2)$，其中 n 為 T 的大小。實際上有一個運算時間為 O(n)的作法：

(一)請描述一個線性時間的作法，可以將一個二元搜尋樹裡的數字依小到大的次序排出來，並放在一個一維陣列上。(5 分)

(二)利用(一)的特性，請設計出一個線性時間的演算法將 T_1、T_2 裡的數字依小到大的次序排出來，並放在一個一維陣列上。(5 分)

(三)給 n 個已經排好的數，請設計出一個線性時間的演算法建構出以這 n 個數為鍵值(key)的二元搜尋樹，同時我們要求這個樹的高度必須為 O(log n)。(10 分)

【擬答】

一、使用二元樹的中序追踪法，來追踪二元搜尋樹，其追踪的次序正是由小而大的順序；在追踪時，每次追踪到一個節點，就將節點資料依序置入一維陣列即可。因爲二元樹的中序追踪所需的時間，與節點個數成正比，故時間爲線性時間。

二、分別對兩棵二元樹進行中序追踪，得到兩組由小而大的序列，再將這兩個序列合併。若兩棵二元搜尋樹分別有 n1 與 n2 個節點，則中序追踪時間爲 O(n1)與 O(n2)，合併所需時間爲 O(n1+n2)，因爲 n=n1+n2，故總時間爲 O(n)。

三、使用 divide-and-conquer 方式，取中間項做爲樹根，中間項之前的資料以遞迴的方式建立成左子樹；中間項之後的資料以遞迴的方式建立成右子樹。

```
treeptr CreateTree(int a[], int low, int high)
{       if (low>high) return NULL;
        else
        {   int mid=(low+high)/2;
            treeptr p=(treeptr)malloc(sizeof(node));
            p->data=a[mid];
            p->leftchild=CreateTree(a, low, mid-1);
            p->rightchild=CreateTree(a, mid+1, high);
            return p;
        }
}
```

三、一個監測器(sensor)可看成是一台計算機，只是其中央處理器速度比一般計算機慢，且記憶體也較小。有一監測器每隔一段時間需從外界讀入一個監測值，一段時間下來總共讀入 n 個數值，而這監測器的任務為記性這 n 個數裡面最小的 k 個，一般來講 k 遠小於 n。現在監測器的記憶體剛好只能放 k 個數值，外加一個暫存器存放剛從外界讀入的監測值。同時由於監測器的處理速度不快，根據估計，若在監測器讀取兩個監測值的間隔時間內，處理器需做 k/2 次以上的比較，那麼監測器可能會錯失一個監測值，長期下來會造成紀錄結果不正確。這裡我們假設 k≥100。請設計出一個可行的作法，可以讓這個監測器正確的完成其任務。(20 分)

【擬答】

(1)使用最大堆積(max-heap)記錄最小的 k 個值。

(2)每次讀入一個值後，與 root 比較，若新讀入的值大於 root，則將新讀入值捨棄。

(3)若新讀入之值小於 root，則以新讀入之值取代掉原來的 root 的值，並且由 root 向下進行調整，使其回復爲最大堆積之情況。

(4)按此方法，每次讀入一個值，最多只需要做 $2\lfloor \log_2 k \rfloor$ 次比較。

四、假設有一個數學運算式(expression)裡面每一個運算元(operand)都是一個正整數，而運算子(operator)只有加法與乘法兩種。請就以下每一數學運算式的格式限制，寫出一段程式算出這個數學運算式的值。(20 分)

(一)數學運算式為前序表示(prefix expression)

(二)數學運算式為後序表示(postfix expression)

這裡你可以假設已經有一個前處理程式 get-next-token()可以運用，其規格為：同傳值為正整數，表示 get-next-token()讀到一個運算元，其大小即為回傳值。

回傳值為 0，表示 get-next-token()讀到運算式的結尾。

回傳值為－1，表示 get-next-token()讀到加號。

回傳值為－2，表示 get-next-token()讀到乘號。

【擬答】

一、

```
int PrefixEval()
{   int x=get_next_token();
    if (x==0) Error();
    else if (x==-1) return PrefixEval()+PrefixEval();
```

```
        else if (x==-2) return PrefixEval()*PrefixEval();
        else return x;
    }
```

(一)
```
int PostfixEval()
{       int x;
        InitStack();
        x=get_next_token();
        while (x!=0)
        {
            if (x>0) Push(x);
            else if (x==-1)
            {   int p=Pop();
                Push(Pop()+p);
            }
            else if (x==-2)
            {   int p=Pop();
                Push(Pop()*p);
            }
            x=get_next_token();
        }
        return Pop();
}
```

五、一個軟體系統內的程式，常需檢查其輸入資料格式是否正確，以確保後續處理不出問題。現有一組輸入資料放在一個陣列 p[1.. n]上代表一棵有根樹(rooted tree)，其中 n 為節點數，且節點編號為 1, 2, …, n。若 p[i]存的值為 j，表示節點 i 的父節點為 j。若節點 i 為根節點則 p[i]存的值為 i。請寫出一個線性時間的演算法檢查這陣列是否表示一棵有根樹。(20 分)

【擬答】

一、使用 disjoint set 的運算 find(i)，並用 path compression 來將所有節點直接指向其 root，最後再檢查所有的節點是否具有相同的 root。 演算法如下：

```
for i ← 1 to n do
  begin
    k←i; count←0;
    while p[k]≠k and count≦n do
      begin
          count←count+1;
          k←p[k];
      end;
    if count>n then Error();
    else
      begin
```

```
            root←k;
            k←i;
            while p[k]≠root do
               begin
                  m←k;
                  k←p[k];
                  p[m]←root;
               end;
         end;
   end;
if p[1]~p[n] are identical then O.K.
else Error();
```