104 年特種考試地方政府公務人員考試試題

等　　別：三等考試
類　　科：資訊處理
科　　目：資料結構

一、二元搜尋法（binary search）使用 divide-and-conquer（分而治之）演算法技巧，對一個已排序（sorted）且長度為 n 的陣列 A[0:n-1]，進行資料搜尋，其最差時間複雜度（worst case time complexity）可降到 $\Theta(\log n)$。

㈠請使用 C 或 Java 語言，修改此二元搜尋法，使其能對未排序（unstorted）且長度為 n 的陣列 A[0:n=1]，以 divide-and-conquer 技巧，進行二元化搜尋。（15 分）

㈡請分析修改後的二元搜尋法其最差時間複雜度（worst case time complexity）以 order Θ 的方式表示。（5 分）

（注意：不可將此陣列數值進行排序，請加註解說明程式碼作法）

擬答：

㈠

```
int partition(int A[],int first,int last)
/*  利用 divide and conquer 技巧，將資料分群，直到無法分群為止*/
{
      int i,j,pivot,temp;

      if(first<last)
      {
            pivot=first;
            i=first;
            j=last;

            while(i<j)
            {
                  while(A[i]<=A[pivot]&&i<last)
                        i++;
                  while(A[j]>A[pivot])
                        j--;
                  if(i<j)
                  {
                        temp=A[i];
                          A[i]=A[j];
                          A[j]=temp;
                  }
            }
/*  會有三群資料，左邊群為小於 pivot 之值集合；右邊群為大於 pivot 之值集合*/
            temp=A[pivot];
            A[pivot]=A[j];
            A[j]=temp;
            return j;
/*將 pivot  搬至中間*/
      } else return -1;
}

int ExamSearch(int A[],int n,int key)
{
      int first,last,mid;
```

```
        left = 0;
        right = n-1;
        mid = -1;

    /*分群動作，並且比對 A[pivot]是否為尋找之值，若不是，類似於 Binary Search 方法，往左
    邊群組或是右邊群組尋找*/
        while (first<=last)
        {
            mid = partition(A,first,last);
            if (key == A[mid]) break ;
            else if (key <A[mid]) last = mid - 1;
                else first = mid + 1 ;
        }
        return mid;
    }
```

㈡此種方法，會受限資料原本排列方式，在平均情況可以達到 O(n)，但在最差情況，也就是已經排序好的資料，反而會花費較多時間，退化成為 $O(n^2)$

時間複雜度為 $\Theta(n^2)$

二、請使用 C 或 Java 語言寫一副程式 void merge(int[] A ,int[] B ,int[] C ,int n)，此副程式將對兩個長度為 n 且已依小到大排序的整數列 A 與 B，合併至長度為 2n 且依小到大排序的整數陣列 C，此副程式的時間複雜度需為 Θ (n)。(20 分)

（注意：請加註解說明程式碼作法）

擬答：

合併作法：將陣列 A 及陣列 B 依大小合併成一個新的陣列

若陣列 A 的數值都已填到新的陣列 ⇒ 將陣列 B 中未填過的最小值填入新陣列

若陣列 B 的數值都已填到新的陣列 ⇒ 將陣列 A 中未填過的最小值填入新陣列

將陣列 A 及陣列 B 中，未填過的最小值填到新的陣列

程式碼如下

```
void merge(int A[],int B[],int C[],int n)
{
    int i,j;

    /*將陣列 A 及陣列 B 依大小合併成一個新的陣列 C*/
    while (i <= n -1 && j <= n - 1)
    {
        if (A[i] <= B[j]) C[k++] = A[i++];

        else C[k++] = B[j++];
    }

    /*若陣列 B 的數值都已填到新的陣列，將陣列 A 中未填過的最小值填入新陣列 C*/
    while (i <= m - 1)
        C[k++] = A[i++];

    /*若陣列 B 的數值都已填到新的陣列，將陣列 A 中未填過的最小值填入新陣列 C*/
    while (j <= n - 1)
        C[k++] = B[j++];
}
```

時間複雜度 $\Theta(n)$

三、㈠請說明使用何種資料結構及其演算法，可有效判斷一運算式（expression）中的巢狀（nested）括號是否正確配對 matched)。(10 分)
㈡請以兩個運算實例{A*[B-(C+D)+8]-16}及{A+[B-(C+5)]}，分別說明此演算法判斷的過程及結果。(10 分)
（注意：未說明判斷的過程，不予計分）

擬答：

㈠利用 stack，依照下列步驟：
如果有左邊系列的括號出現，就把左括號 Push 進 Stack
如果有右邊系列的括號出現，就從 Stack 中 Pop 出一個左括號並比較是否為同系列
最後只要再判斷 stack 中還有沒有孤單的括號存在，如果沒有就是配對成功，如果有，當然就是配對失敗
其餘文字皆捨棄不管

㈡

{A*[B(C+D)+8]-16}
Push {
捨棄 A
捨棄 *
Push [
捨棄 B
Push (
捨棄 C
捨棄 +
捨棄 D
Pop (，與)比較，相符
捨棄 +
捨棄 8
Pop [，與]比較，相符
捨棄 -
捨棄 16
Pop {，與 }比較，相符
Stack 為空，配對成功

{A+[B-(C+5])}
Push {
捨棄 A
捨棄 +
Push [
捨棄 B
捨棄 -
Push (
捨棄 C
捨棄 +
捨棄 5
Pop (，與]比較，不相符
跳離判斷式，Stack 不為空，配對不成功

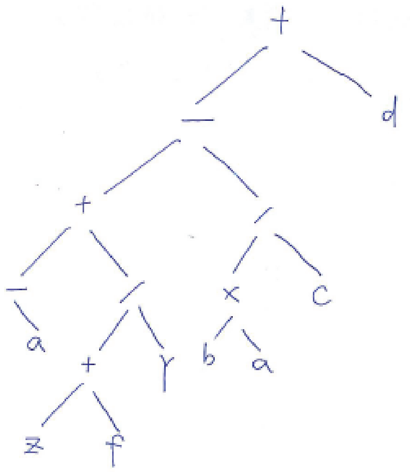
四、㈠一運算式（expression）為：-a+(z+f)/y-b*a/c+d，請依運算元優先順序，繪出其二元樹（binary tree）。（10 分）

㈡請列出此二元樹的前序走訪（preorder traversal）。（5 分）

㈢請列出此二元樹的廣度優先走訪（breadth-first search traversal）。（5 分）

擬答：

㈠



+：加
-：減
×：乘
/：除

㈡

＋ - ＋ - a/+zfy/×bacd

㈢

＋ - d ＋ / - /×ca+ybazf

五、一個圖形（Graph）包含五個頂點（vertex），$V_1,V_2,\ldots,V_5$，其相鄰矩陣（adjacency matrix）

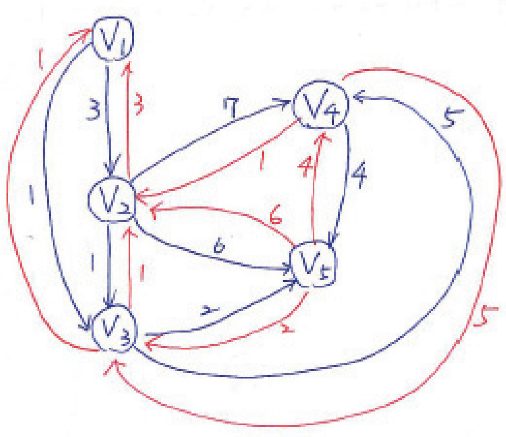
$$A=\begin{bmatrix} 0 & 3 & 1 & \infty & \infty \\ 3 & 0 & 1 & 7 & 6 \\ 1 & 1 & 0 & 5 & 2 \\ \infty & 1 & 5 & 0 & 4 \\ \infty & 6 & 2 & 4 & 0 \end{bmatrix}$$

㈠請使用 Floyd 的方法，計算此圖形的最短路徑長度矩陣（shortest path length matrix），表示任兩頂點間最短路徑長度。請依序列出最短路徑長度矩陣變化過程。（15 分）

㈡請使用 Kruskal 的方法，依序繪出加入此圖形的最小成本擴張樹（minimum cost spanning tree）每一邊的過程。（5 分）

擬答：

㈠ A 之圖形



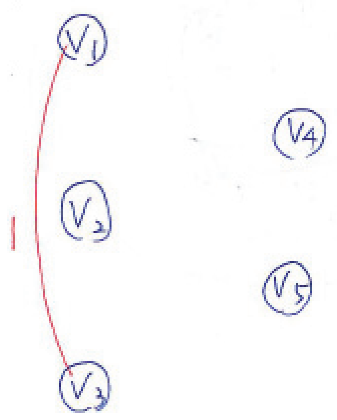
$$A^0 = \begin{bmatrix} 0 & 3 & 1 & \infty & \infty \\ 3 & 0 & 1 & 7 & 6 \\ 1 & 1 & 0 & 5 & 2 \\ \infty & 1 & 5 & 0 & 4 \\ \infty & 6 & 2 & 4 & 0 \end{bmatrix}$$

$$A^1 = \begin{bmatrix} 0 & 3 & 1 & \infty & \infty \\ 3 & 0 & 1 & 7 & 6 \\ 1 & 1 & 0 & 5 & 2 \\ \infty & 1 & 5 & 0 & 4 \\ \infty & 6 & 2 & 4 & 0 \end{bmatrix} \quad A^2 = \begin{bmatrix} 0 & 3 & 1 & 10 & 9 \\ 3 & 0 & 1 & 7 & 6 \\ 1 & 1 & 0 & 5 & 2 \\ 4 & 1 & 2 & 0 & 4 \\ 9 & 6 & 2 & 4 & 0 \end{bmatrix} \quad A^3 = \begin{bmatrix} 0 & 1 & 1 & 6 & 3 \\ 2 & 0 & 1 & 6 & 3 \\ 1 & 1 & 0 & 5 & 2 \\ 3 & 1 & 2 & 0 & 4 \\ 3 & 3 & 2 & 4 & 0 \end{bmatrix}$$

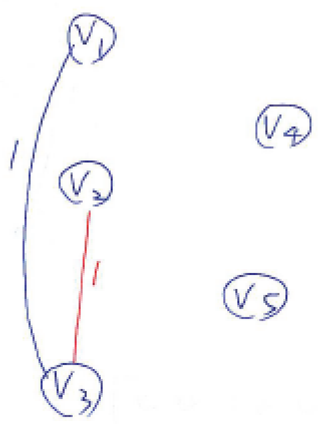
$$A^4 = \begin{bmatrix} 0 & 1 & 1 & 6 & 3 \\ 2 & 0 & 1 & 6 & 3 \\ 1 & 1 & 0 & 5 & 2 \\ 3 & 1 & 2 & 0 & 4 \\ 3 & 3 & 2 & 4 & 0 \end{bmatrix} \quad A^5 = \begin{array}{c} \\ v1 \\ v2 \\ v3 \\ v4 \\ v5 \end{array} \begin{array}{c} \begin{array}{ccccc} v1 & v2 & v3 & v4 & v5 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 6 & 3 \\ 2 & 0 & 1 & 6 & 3 \\ 1 & 1 & 0 & 5 & 2 \\ 3 & 1 & 2 & 0 & 4 \\ 3 & 3 & 2 & 4 & 0 \end{bmatrix} \end{array}$$

㈡使用 kruskal 找出 MCST

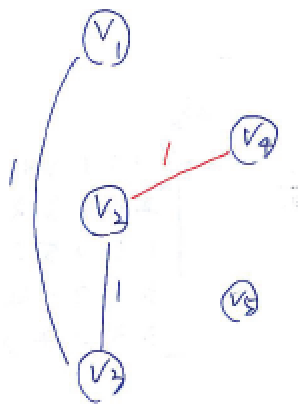
第一條



第二條



第三條



第四條