《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Θ(log n)。

(一)請使用C或Java語言，修改此二元搜尋法，使其能對未排序（unsorted）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）

(二)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）

試題評析	未排序資料要做二分搜尋法，原本不可能，但題意應該是要用快速排序的partition來進行，很像是select k-th smallest的方法，分成兩群之後再決定要往前或往後縮小範圍搜尋，此題可能讓考生較難掌握到題意。
考點命中	《資料結構》，高點出版，王致強編撰，頁9-34～9-36。

答：

(一)divide-and-conquer 是利用partition，將資料分成<pivot,pivot與>pivot三種，再判斷是否已找到，或是要往那一群繼續遞迴搜尋下去。

```
#include <stdio.h>
void swap(int A[], int x, int y)
{ int temp;   /* 交換 A[x]與A[y]的函數 */
      temp=A[x]; A[x]=A[y]; A[y]=temp;
}
/* partition 可以將資料分成兩群，第一群<pivot；第二群>pivot */
int partition(int A[], int left, int right) {
      int pivot, i, j;
      if (left<=right) { /* 還有資料，繼續 partition */
            pivot=A[left];   /* 取最開頭一項做為 pivot */
            i=left;   /* i由左而右, 找<pivot的項集中到A[0]~A[i] */
            for(j=left+1; j<=right;j++)
                  if (pivot>A[j]) {
                        i++;
                        swap(A,i,j);
                  }
            A[left]=A[i];/* 將pivot移到兩群之間 */
            A[i]=pivot;
            return i;      /* 傳回pivot的位置 */
      } else return -1; /* 沒有資料，傳回-1 */
}
int BinarySearch(int A[], int n, int x)
{     /* 在A[0]~A[n-1]之間找尋 x */
      int left=0, right=n-1, mid=-1, k;
      while (left<=right) { /* 只要還有資料，繼續 partition */
            mid=partition(A,left,right);
            if(x==A[mid]) left=right+1; /*找到x，將搜尋資料設為空的*/
            else if (x<A[mid]) right=mid-1;/* x<pivot,往前一群搜尋*/
            else left=mid+1; /* x>pivot,往後一群搜尋*/
```

```
    }
    return mid; /* 未找到，傳回-1 */
  }
```

(二)最差情況：在遇到原始資料是由小而大的狀況，partition 總時間=(n-1)+(n-2)+…+3+2+1=O(n2)。

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陣列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）

試題評析	兩排序串陣列合併，屬於合併排序中的基本處理，小心撰寫程式，拿分應該不難。
考點命中	《資料結構》，高點出版，王致強編撰，頁9-38。

答：

```
void merge(into A[], int B[], int C[],int n) {
    int i=0,j=0,k=0;
    while(i<n && j<n) {
        /* 比較A與B最小的項目，較小的先搬到C的結尾，搬動完繼續取下一項比較 */
        if (A[i]>B[j]) C[k++]=B[j++];
        else C[k++]=A[i++];
    }
    /* 若A陣列還有資料沒搬，則逐一搬到C */
    while (i<n) C[k++]=A[i++];
    /* 若B陣列還有資料沒搬，則逐一搬到C */
    while(j<n) C[k++]=B[j++];
}
```

時間複雜度為：Θ(n)

三、(一) 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）

(二)請以兩個運算式實例{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）

試題評析	本題考的括號匹配檢查，使用堆疊記錄左括號，再取出與右括號比對即可，屬於堆疊的基本應用題。
考點命中	《資料結構》，高點出版，王致強編撰，頁4-67。

答：

(一) 使用stack來將左括號push到stack中，遇到左括號時，由stack pop左括號，檢查其是否匹配，演算法如下：

```
init. Stack is empty;
while (not EOF) {
    x <- read();
    if (x is 左括號) push(x);
    else if (x is 右括號) {
        if (stack empty) { output "Not Match"; exit; }
        y <- pop();
        if (x 與 y 不是同一類型的左右括號) { output "Not Match"; exit; }
    }
}
```

if (stack empty) output “Match”;
else output “Not Match”;

(二)實例1:

{ ：push stack 為 { (底部)
A : 丟棄
* : 丟棄
[: push stack 為 [{ (底部)
B : 丟棄
- : 丟棄
(: push stack 為 ([{ (底部)
C + D : 丟棄
) : pop up (, 匹配 and continue， stack 為 [{ (底部)
+ 8 : 丟棄
] : pop up [, 匹配 ，stack: { (底部)
- 16: 丟棄
} : pop up { , 匹配 stack :空的,
檢查沒有錯誤，匹配成功 output “Match”

實例2:

{ : push, stack 為 { (底部)
A: 丟棄
＋: 丟棄
[: push, stack 為 { [(底部)
B : 丟棄
-: 丟棄
(: push, stack 為 { [((底部)
C : 丟棄
+ : 丟棄
5 : 丟棄
] : pop up (, output “Not Match”

四、(一) 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）

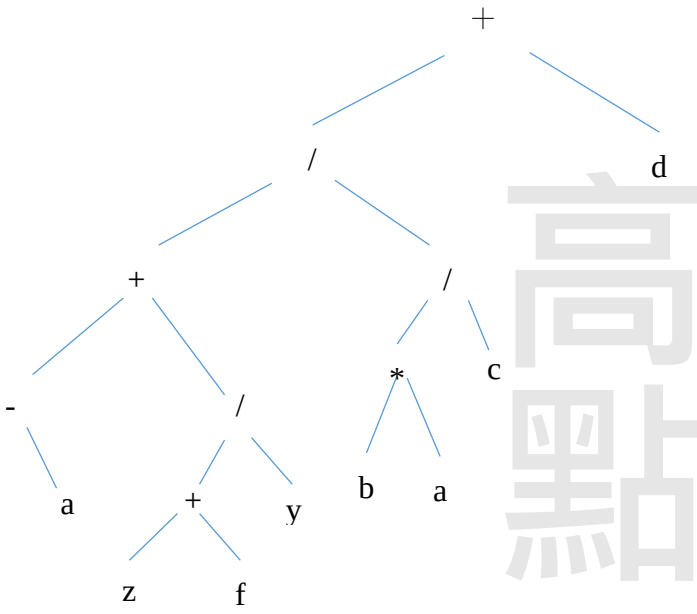
(二) 請列出此二元樹的前序走訪（preorder traversal）。（5分）

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

試題評析	前半段考算式樹，後段考前序走訪與廣度優先走訪，屬於基本問題，取分不難。
考點命中	《資料結構》，高點出版，王致強編撰，頁6-57～6-58、6-16～6-18、8-24～8-25。

答：

(一) 二元樹如下



(二)前序走訪：＋／＋－a ／+z f y／＊b a c d
(三)廣度優先走訪：＋／d ＋／－／＊c a ＋y b a z f

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

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

(一)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）
(二)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）

試題評析	本題矩陣有些問題，如果如題意為有向圖形，題(一)可以計算，但(二)可能無法計算。按照正常方式計算仍能取得分數。
考點命中	《資料結構》，高點出版，王致強編撰，頁8-71、8-72、8-41、8-49。

答：
(一) 本題是有向圖，須小心計算。

A^0	V1	V2	V3	V4	V5
V1	0	3	1	∞	∞
V2	3	0	1	7	6
V3	1	1	0	5	2
V4	∞	1	5	0	4
V5	∞	6	2	4	0

A^1	V1	V2	V3	V4	V5

V1	0	3	1	∞	∞
V2	3	0	1	7	6
V3	1	1	0	5	2
V4	∞	1	5	0	4
V5	∞	6	2	4	0

A^2	V1	V2	V3	V4	V5
V1	0	3	1	10	9
V2	3	0	1	7	6
V3	1	1	0	5	2
V4	2	1	2	0	4
V5	7	6	2	4	0

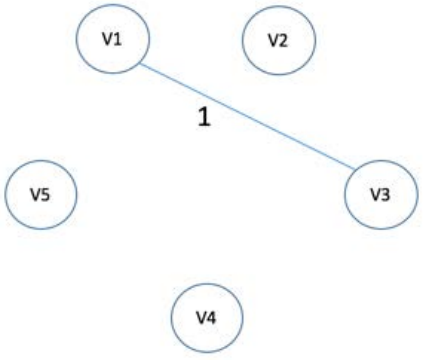
A^3	V1	V2	V3	V4	V5
V1	0	2	1	6	3
V2	2	0	1	6	3
V3	1	1	0	5	2
V4	2	1	2	0	4
V5	3	3	2	4	0

A^4	V1	V2	V3	V4	V5
V1	0	2	1	6	3
V2	2	0	1	6	3
V3	1	1	0	5	2
V4	2	1	2	0	4
V5	3	3	2	4	0

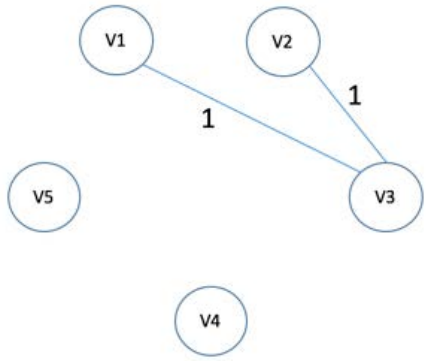
A^5	V1	V2	V3	V4	V5
V1	0	2	1	6	3
V2	2	0	1	6	3
V3	1	1	0	5	2
V4	2	1	2	0	4
V5	3	3	2	4	0

(二)

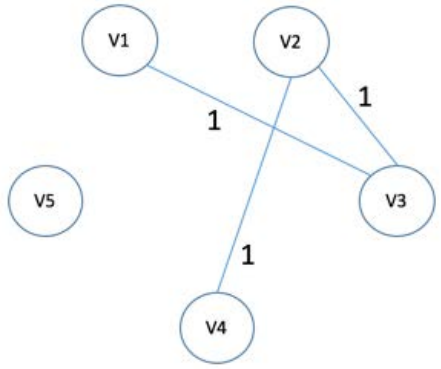
(1)



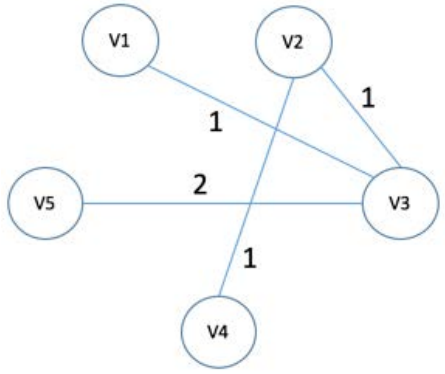
(2)



(3)



(4)



註：本題第二小題似乎題目有問題，有向圖形應該不能進行Kruskal演算法找最低成本擴張樹。