厦门大学《数据结构》期末试题·答案

考试日期：2010•1（B）　　信息学院自律督导部

自律督导部

一、（本题 10 分）

（1）线性表和广义表的主要区别点是什么？已知广义表：C=(a,(b, (a,b)), ((a,b), (a,b))), 则 tail(head(tail(C))) =？

（2）满足什么条件可以实施二分查找？二分查找的时间复杂度是多少？

答：（1）线性表和广义表都是元素 a1,a2,…,an 组成的序列，其主要区别点在于：在线性表中，ai 是单个元素（原子）；在广义表中，ai 可以是单个元素（原子），也可以是广义表。

tail(head(tail(C))) = ((a,b))

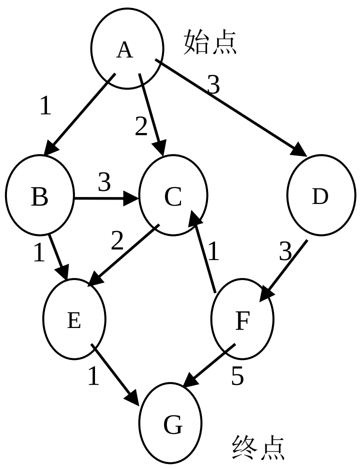
（2）序列 a1,a2,…,an 必须在数组（顺序表）中，且有序；时间复杂度为 O(log n)。

二、（本题 10 分）证明：一棵二叉树的先序序列和中序序列可惟一确定这棵二叉树。

证明：

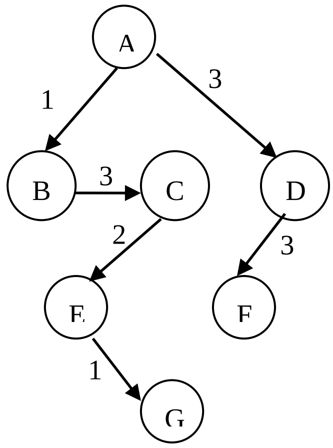
设一棵二叉树的先序序列和中序序列分别存放在一维数组 A[1..n]和 B[1..n]中。因为先序序列的第一个结点 A[1]为二叉树的根结点，在中序序列中找到与 A[1]相同的结点，不妨假设 B[i]=A[1]；又因为二叉树的任何一棵子树的结点是紧挨在一起的，故所构造的二叉树的左子树由先序序列 A[2..i]和中序序列 B[1..i-1]确定的二叉树组成，而所构造的二叉树的右子树由先序序列 A[i+1..n]和中序序列 B[i+1..n]确定的二叉树组成。这是一个递归过程，当先序序列和中序序列分别含有 3 个以下的结点时，可惟一确定对应的二叉树。因此，由一棵二叉树的先序序列和中序序列可以惟一确定这棵树。

三、（本题 15 分）某带权有向图如下：



（1）写出深度优先搜索结点访问序列，并画出深度优先生成树；（当有多种选择时，编号小的结点优先。）

（2）写出该图的拓扑序列（当有多种选择时，编号小的结点优先。）
（3）将该图作为 AOE 网络，写出求关键路径的过程。
解：
（1）深度优先搜索顺序是：A, B, C, E, G, D, F
深度优先生成树如下图所示。



（2）该图的拓扑序列为：A,B,D,F,C,E,G
（3）求解过程如下：

ve(A)=0　ve(D)=3　　ve(F)=ee(D)+3=6　　ve(B)=1;
ve(C)=max{ ve(A)+2 ,ve(B)+3, ve(F)+1}=7
ve(E)=max{ ve(B)+1, ve(C)+2 }=9
ve(G)=max{ ve(E)+1, ve(F)+5 }=11
vl(G)=11　　　　　　vl(E)=vl(G)-1=10　　　　vl(C)=vl(E)-2=8
vl(B)=min{vl(E)-1,　　vl(C)-3}=5
vl(F)=min{ vl(G)-5, vl(C)-1 }=6　　　　　　vl(D)=vl(F)-3=3
vl(A)=min{ vl(B)-1, vl(C)-2, vl(D)-3 } =0

所以

e(AB)=ve(A)=0　e(AC)=ve(A)=0　　e(AD)=ve(A)=0
e(BC)=ve(B)=1　e(BE)=ve(B)=1　　e(CE)=ve(C)=7　　e(DF)=ve(D)=3
e(EG)=ve(E)=9　e(FC)=ve(F)=6　　e(FG)=ve(F)=6
l(AB)=vl(B)-1=4　l(AC)=vl(C)-2=6　　l(AD)=vl(D)-3=0
l(BC)=vl(C)-3=5　l(BE)=vl(E)-1=9　　l(CE)=vl(E)-2=8　　l(DF)=vl(F)-3=3
l(EG)=vl(G)-1=10　l(FC)=vl(C)-1=7　　l(FG)=vl(G)-5=6

所以

e(AD)=l(AD)　　e(DF)=l(DF)　　e(FG)= l(FG)

所以关键路径为：ADFG

四、（本题 10 分）已知待散列存储的关键字序列为（4,15,38,49,33,60,27,71），哈希函数为 H(key)=key MOD 11，哈希表 HT 的长度为 11，采用线性探测再散列法解决冲突。试构造此哈希表，并求出在等概率情况下查找成功的平均查找长度。

解：哈希表为

0	1	2	3	4	5	6	7	8	9	10
33				4	15	38	49	60	27	71

平均查找长度为：(1+2+2+3+1+4+5+6)/8 =3。

五、（本题 15 分）以关键字序列（29，18，25，47，58，12，51，10）为例，执行以下排序算法，写出每一趟结束时的关键字状态：
（1）增量序列为 5，3，1 的希尔排序（2）快速排序（3）堆排序。
解：（1）
第 1 趟： 12，18，10，29，47，58，51，25
第 2 趟： 12，18，10，29，25，58，51，47
第 3 趟： 10，12，18，25，29，47，58，51

（2）
第 1 趟：10，18，25，12，29，58，51，47
第 2 趟：10，18，25，12，29，47，51，58
第 3 趟：10，12，18，25，29，47，51，58
第 4 趟：10，12，18，25，29，47，51，58

（3）
第 1 趟：58，47，51，29，18，12，25，10
第 2 趟：51，47，25，29，18，12，10，（58）
第 3 趟：47，29，25，10，18，12，（51，58）
第 4 趟：29，18，25，10，12，（47，51，58）
第 5 趟：25，18，12，10，（29，47，51，58）
第 6 趟：18，10，12，（25，29，47，51，58）
第 7 趟：12，10，（18，25，29，47，51，58）
第 8 趟：10，（12，18，25，29，47，51，58）

六、（本题 10 分）在两个有序线性表中，寻找是否存在共同元素。如果存在共同元素，返回第一个共同元素在第一个有序表中的位置。请设计数据结构，并在其上设计算法。

答：可以参考有序表的归并算法。
数据结构可以使用一维数组，并且第 0 个元素放空。

```
int SearchCommonItem(int a[n], int b[m])//第 0 位放空，返回值为 0 代表找不到
{
    int i=1,j=1;
    while (i<=n && j<=n)
    {
        if (a[i]==b[j]) return i;
        else (a[i]<b[j]) i++;
        else j++;
    }
    return 0;
```

}

七、（本题 15 分）在带头结点的非空线性链表中，试设计一算法，将链表中数据域值最小的那个结点移到链表的最前面，其余各结点的顺序保持不变。要求：不得额外申请新的链结点。

解：程序如下：

```
typedef struct node {
    int data;
     struct node * next;
}Node,*LinkList;
void MinFirst(LinkList L)
{Node *p,*q,*ptrmin;
     if(L->next == NULL) return; //空表
     ptrmin = L; //ptrmin 指向当前最小结点的前一个结点
    p = L->next;//p 指向当前结点的前一个结点
    while(p->next!=NULL) {
         if( p->next->data < ptrmin->next->data ) ptrmin = p;
        p = p->next;
    }
//q 指向最小结点，并从链表中删除
    q = ptrmin->next; ptrmin->next = q->next;
q->next = L->next; L->next = q; //q 指向的最小结点插入到链表头
```

八、（本题 15 分）请利用两个队列 Q1 和 Q2 来模拟一个栈。已知队列的三个运算定义如下：bool EnQueue(Queue &Q,int e):插入一个元素 e 入队列； bool DeQueue(Queue &Q,int &e):删除一个元素 e 出队列；bool QueueEmpty(Queue Q)：判队列为空。假设数据结构 Queue 已定义，栈 Stack 的数据结构定义如下。请利用队列的运算来实现该栈的三个运算：Push(Stack ST,int x):元素 x 入 ST 栈；Pop(Stack ST, int x)：ST 栈顶元素出栈，赋给变量 x；StackEmpty(Stack ST)：判 ST 栈是否为空。

```
typedef struct {
    Queue Q1;
    Queue Q2;
} Stack;
```

答：队列 Q1 或 Q2 中的某一个保存所有的栈元素。

程序如下：

```
bool StackEmpty(Stack S)
{
    return QueueEmpty(S.Q1) && QueueEmpty(S.Q2);
}

bool Push(Stack &S, int e)
{
    if( QueueEmpty(S.Q2)==false )
        return EnQueue(S.Q2, e);
    return EnQueue(S.Q1, e);
}
```

```
bool Pop( Stack &S, int &e)
{Queue *from,*to;
  int x;
      if( QueueEmpty(S.Q1)==true && QueueEmpty(S.Q2)==true ) return false;
      if( QueueEmpty(S.Q1)==false ) {
            from = &S.Q1; to = &S.Q2;
      }
      else {
            from = &S.Q2; to = &S.Q1;
      }

      DeQueue(*from,x);
      while( QueueEmpty(*from)==false ) {
            EnQueue(*to,x);
            DeQueue(*from,x);
       }
      e = x;
      return true;
}
```