上海交通大学 1997 年硕士研究生入学考试试题数据结构及程序设计技术

试题编号：１９

题一（６分）有五个数依次进栈：1,2,3,4,5.在各种出栈的序列中，以 3,4 先出的序列有哪几个。（３在４之前出栈）

题二（４分）试写出进栈操作，出栈操作算法的时间复杂性。

题三（４分）已知ＫＭＰ串匹配算法的模式串是 AABBAAB,试写出改进后的 NEXT 信息帧。

题四（４分）设某通信电文由Ａ、Ｂ、Ｃ、Ｄ、Ｅ、Ｆ六个字符组成，它们在电文中出现的次数分别是 16,5,9,3,20,1。试画出编码用的哈夫曼树。

题五（５分）已知某排序平衡二叉树Ｔ具有下列特点：（１）结点的关键字均在 1 到 9 范围内；（２）在Ｔ中存在一个关键字为 n1 的叶结点，若删去该结点，立即插入一个关键字为 n1 的结点，得到的平衡树与原Ｔ不相同；（３）在Ｔ中存在另一个关键字为 n2 的非叶结点，删去它，并立即插入 n2 结点，得到与原Ｔ相同的平衡树；（４）在Ｔ中插入某 n3 结点后立即删除它，得到的平衡树与原Ｔ不相同。试画出具有上述特点的最简单（结点个数最少）的平衡树Ｔ，并写明 n1,n2,n3 分别等于几？

题六（９分）某整型数组Ａ的１０个元素值依次为 6,2,9,7,3,8,4,5,0,1,用下列各排序方法，将Ａ中元素由小到大排序。

（1） 取第一个元素值 6 作为分割数，（2） 试写出快速排序第一次分隔后Ａ中的结果。

（3） 用堆排序，（4） 试写出将第一个选出的数据放在Ａ的最后位置上，（5） 将Ａ调整成堆后的Ａ中结果。

（6） 用基数为 3 的基数排序法，（7） 试写出第一次分配和收集后Ａ中的结果。

题七（１４分）某赋权有向图及它的单邻接表如下：

（1） 试写出深度优先搜索顺序。

（2） 画出深度优先生成树。

（3） 将该图作为ＡＯＥ网络图，（4） 试写出Ｃ的最早发生时间及活动ＦＣ的最晚开始时间。

（5） 用 Dijkstra 算法计算源点Ａ到各顶点的最短路径，（6） 试写出当计算出ＡＤ及ＡＧ的最短路径时，（7） Ａ到其它各点路径（中间结点）的值。

始点

1　　2　3

3　2

1　2　　1　3

1　　5

终点

请在下列各题的 （N） 处，填写适当的 Pascal 语句（或其它成份），完成各题的程序。

题八（１０分）下列程序输入一个正整数 N(０<n<10)，打印 1,2,….,n 各数字的全排列，例如输入 n=3,打印 123,132,213,231,312,321

```
program   exam8(input,output);
const maxn=9;
var  a:array[1..maxn] of integer;{放全排序一个值}
s:set of 1..maxn;{放１到９各数字的集合}
n:integer
procedure  load(j:integer);{将 S 中数字装入到 a 中}
var  j,k integer;
begin
for   j:=1  to  n  do
if   j  in  s
then   begin
s:=  (1)
a[i]:=j;
if   i<n
then    (2)
else  for  k:=1  to  n   do  writen(a[k]);
(3)
end
end{load};
begin {main}
readln(n);   s:=[1..n];   load(1)
end.
```

题九（１０分）下面的过程对二叉树进行后序遍历（非递归）。假设已有栈的一些操作过程说明。并说明树的结点类型：

```
type pointer=  node;
node=record
data:integer ;
left,right :pointer
end;
procedure  post(p:pointer);
var q:pointer;
begin
```

```
if p<> then
begin create_stack(s);{建立一个 S 栈，并初始化为空栈}
while (P<>nil)  or  not  empty_stack(s){s 栈不空}   do
if  p<>nil
then  begin
push(s,p); {将 P 进栈}
push(s,p);{P 作为标记进栈}
P:=    （ 1 ）
end
else  begin
pop(s,p);{将标记退出 S 栈}{退出到 P 中}
if p<>nil
then  begin
push(s,nil){标记进栈}
（ 2 ）      ;
end
else  begin
（3）       ;
write(q  .data);{访问结点，打印结点数据}
end
end
end{post}
```

题十（ 8 分）设结点的类型定义如下：

```
type
node=record  data:integer; link:integer  end;
```

在数组 a 中存放了 1 0 个 结点：

```
var  a:array[1..10]  of  node;
```

假定在主程序中已经执行了下列语句：

```
for i:=1  to  10  do
begin  a[i].data:=i,  a[i].link:=0   end;
```

最初将 1 0 个结点看成分别属于 1 0 个集合，每个集合有且仅有一个结点，调用下面过程，判断 data=m 和 data=n 的两个结点是否在同一个集合中（最初肯定属于不同集合，除非 m=n），若不在同一集合中，则将这两个结点合并到一个集合中。否则，已在同一个集合中，什么也不做。（此方法用于 Kruskal 求最小生成树的算法中）

```
procedure  merge_set(m,n:integer);
function  find(k:integer):integer;
var   f:integer;
begin
```

```
f:=k;
while a[k].link<>0  do  f:=a[k].link;
(1)          ;
 end;
begin
m:=find(m);    n:=frind(n);
if  m<>n  then      (2)
end{merge_set}
```

题十一（１４分）设有数组变量说明：

```
var  a:array[1..n] of  record   key:integer;  next:0..n  end;
```

假设各元素的 key 已有值，并且假设最大值＜＝９９９９，再假定在主程序已执行下面语句，已将各元素组成一个链：

```
for  i:=Ø  to n-1  do  a[i].next:=i+1;
a[i]:= Ø
```

下面是以 10 为基的基数排序法过程，将数组 a 按 Key 由小到大排序。

```
Procedure  bucket_sorting;
Const  maxkey=9999;
Var bucket,tail :array[0..9] of 0..n;
P,last:0..n;
i:=0..9;
d:integer;
begin
 d:=1;
 repeat
for i:=0 to 9 do tail[i]:= Ø;
{分解}
P：＝a[0].next;{a[0].next 是链的第一个结点}
While P<> Ø  do
Begin   i:=a[p].key div d mod 10;
If          (1)
Then bucket[i]:=p
Else   a[      (2)      ].next:=p;
Tail[i]:=p;
(3)
 end
{收集}
 last:=0
for i:=0  to  9  do
```

```
if          (4)
    then  begin
a[last].next:=      (5)       ;
    last:=      (6)
        end;
    a[last].next:= Ø;
d:=           (7)
    until  d>maxkey
end{bucket_sorting}
```

题十二（12 分）下面是一个判断二叉树是否是排序平衡树的函数说明，若是排序平衡树，则返回值为真，否则返回值假，另外，以参数的形式返回二叉树的高度和最大最小结点值。结点的类型定义与题九的定义相同。

```
Function  isbalance(p:pointer;var   h,min,max:integer)
Var  hleft,hright,lmax,rmin:integer;
    Begin
If  p<>nil
    then   begin
if osbalance(p  .left,hleft,        (1)          )
and isbalance(P  .right,hright,       (2)          )
    then  begin
        if  hleft=0
then  begin  min:=p  .data;  lmax:=p  .data-1  end;
        if  hright=0
then  begin  max:=p  .data;  rmin=p  . data+1  end;
        if hleft>hright
    then  h:=hleft+1  else  h:=hright+1;
    isbalanced:=(abs(hleft-hright)<=1)
            and (       (3)          )
            and (       (3)          )
        end
    else  isbalanced:=false
        end
else  begin  h:=0;  isbalanced:=true   end
end{isbalance}
```