

中科院计算机技术研究所 1999 年数据结构与程序设计试题和答案

一、选择题 (20 分 每空 2 分)

1. \_\_\_\_ 的遍历仍需要栈的支持。

1.前序线索树 2.中序线索树 3.后序线索树

2.若度为  $m$  的哈夫曼树中,其叶结点个数为  $n$ ,则非叶结点的个数为 \_\_\_\_。

1. $n-1$  2. $\lceil n/m \rceil - 1$  3. $\lceil (n-1)/(m-1) \rceil$  4. $\lceil n/(m-1) \rceil - 1$  5. $\lceil (n+1)/(m+1) \rceil - 1$

3.最优二叉树(哈夫曼树)、最优查找树均为平均查找路径长度  $\sum w_h$  最小的树,其中对最优二叉树, $n$  表示 \_\_\_\_,对最优查找树, $n$  表示 \_\_\_\_; 构造这两种树均 \_\_\_\_。

1.结点数 2.叶结点数 3.非叶结点数 4.度为二的结点数 5.需要一张  $n$  各关键字的有序表 6.需要对  $n$  个关键字进行动态插入 7.需要  $n$  个关键字的查找概率表 8.不许要任何前提

4.对于前序遍历与中序遍历结果相同的二叉树为 \_\_\_\_; 对于前序遍历与后序遍历结果相同的二叉树为 \_\_\_\_。

1.一般二叉树 2.只有根结点的二叉树 3.根结点无左孩子的二叉树 4.根结点无右孩子的二叉树 5.所有结点只有左子树的二叉树 6.所有结点只有右子树的二叉树

5. $m$  路 B+树是一棵 \_\_\_\_, 其结点中关键字最多为 \_\_\_\_ 个, 最少为 \_\_\_\_ 个。

1. $m$  路平衡查找树 2. $m$  路平衡索引树 3. $m$  路 trie 树 4. $m$  路键树 5. $m-1$  6. $m$  7.  $m+1$

8. $\lceil m/2 \rceil - 1$

9. $\lceil m/2 \rceil$  10. $\lceil m/2 \rceil + 1$

二、填空题(10 分,每空一分)

1.对于给定的  $n$  个元素,可以构造出的逻辑结构有 \_\_\_\_, \_\_\_\_, \_\_\_\_, \_\_\_\_ 四种。

2.具有  $n$  个关键字的 B-树的查找路径长度不会大于 \_\_\_\_。

3.克鲁斯卡尔算法的时间复杂度为 \_\_\_\_, 他对 \_\_\_\_ 图较为适合。

4. 深度为  $k$  (设根的层数为 1) 的完全二叉树至少有\_\_\_\_个结点, 至多有\_\_\_\_个结点,  $k$  和结点数  $n$  之间的关系是\_\_\_\_\_。

### 三、问答题(10 分, 每题 5 分)

1. 一棵非空的有向树中恰有一个顶点入度为 0, 其他顶点入度为 1. 但一个恰有一个顶点的入度为 0, 其他顶点入度为一的有向图却不一定是一棵有向树。请举例说明之。

2. 若有  $n$  个元素以构成一个小根堆, 那么如果增加一个元素为  $K(n+1)$ , 请用文字简要说明你如何在  $\log_2(n)$  的时间内将其重新调整为一个堆?

### 四、阅读下述程序, 指出程序的输出。(10 分)

```
void g(int**);  
main(){  
    int line[100], i;  
    int *p=line;  
    for(i=0; i<100; i++){  
        *p=i;  
        g(&p);  
    }  
    for(i=0; i<100; i++) printf("%d\n", line[i]);  
}  
void g(int**p){  
    (**p)++;  
    (*p)++;  
}
```

### 五、编程题。(共 50 分, 要求写出设计思想和程序注解)

1. 设一单向链表的头指针为 **head**, 链表的记录中包含着整数类型的 **key** 域, 试设计算法, 将

此链表的记录按照 **key** 递增的次序进行就地排序. (10 分)

2. 给定一个整数数组  $b[0..N-1]$ ,  $b$  中连续相等元素构成的子序列称为平台. 试设计算法, 求出

$b$  中最长平台的长度. (20 分)

3. 自由树(即无环连通图)  $T=(V,E)$  的直径是树中所有点对间最短路径长度的最大值, 即  $T$  的

直径定义为  $\text{MAX } d(u,v)$  这里  $d(u,v)$  表示顶点  $u$  到顶点  $v$  的最短路径长度(路径长度为路径中

所含的边数). 试写一算法求  $T$  的直径, 并分析算法的时间复杂度(时间复杂度越小得分越高) (20 分)

九、(14 分) 设正在处理器上执行的一个进程的页表如下. 页表的虚页号和物理块号是十进制数, 起始页号(块号)均为 0. 所有的地址均是存储器字节地址, 页的大小为 1024 字节.

A. 详述在设有快表的请求分页存储管理系统中, 一个虚地址转换成物理内存地址的过程.

B. 下列虚地址对应与什么物理地址: (1) 5499; (2) 2221;

虚页号	状态位	访问位	修改位	物理块号
0	1	1	0	4
1	1	1	1	7
2	0	0	0	---
3	1	0	0	2
4	0	0	0	---
5	1	0	1	0

注释: 访问位 --- 当某页被访问时, 其访问位被置为 1.

数据结构与程序设计 参考答案

一. 选择题

一. 3; 二. 3; 三. 2, 1, 7; 四. 6, 2; 五. 2, 6, 9;

二. 填空题

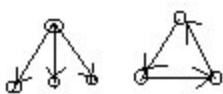
1. 集合, 线性结构, 树型结构, 图结构

2.  $\text{LOG}[m/2]((n+1)/2) + 1$

3.  $O(e \log_2(e))$ , 稀疏

4.  $2^{k-1}$ ,  $2^k - 1$ ,  $k = \lceil \log_2(n) \rceil + 1$  或  $k = \lceil \log_2(n+1) \rceil$

三. 问答题

1. 如: . 即一有向树外加一个有向环所构成的所有有向图都不是一棵有向树.

2. 将  $K_{n+1}$  插入数组的第  $n+1$  个位置(即作为一树叶插入), 然后将其与双亲比较. 若是大于等于

其双亲则终止调整, 否则将  $K_{n+1}$  与其父亲交换. 重复的将  $K_{n+1}$  与新的双亲比较, 算法终止于

$K_{n+1}$  大于等于其双亲或  $K_{n+1}$  本身已上升为根为止.

四. 程序输出 1, 2, 3, ..., 100. 本题主要考察学生对函数中修改指针的理解

五.

1. 略.

2. 思想: 若已知  $b[0..i-1]$  的最长平台的长度为  $p$ , 且  $b[i]$  是下一平台的开始位置, 即

$b[i] \neq b[i-1]$ , 则从

位置  $i$  开始计算下一平台的长度. 若  $p < q$ , 则更新最长平台的长度  $p$ . 当  $i = N$  时,  $p$  即为所求:

```
i:=0; p:=0;//初始化
while i<N do{
    q:=1;//b[i]构成长度为 1 的平台
    i:=i+1;//改查下一元素
    while i<N and b[i-1]=b[i] do{
        q:=q+1;//当前平台长度加 1
        i:=i+1
    }
    if q>p then p:=q;
}
```

3.求树的直径的时间复杂度可为  $O(n)$ ,  $O(n^2)$  和  $O(n^3)$ , 若时间为  $O(n^3)$  适当扣分

解法一:先调用求所有点对间最短路径算法(每边权数为 1)如 FLOYD 算法,然后对指代矩阵求最大的  $COST[i,j]$  作为直径.

解法二:修改 Bfs 算法,使之遍历时,记录当前访问结点的深度(离根的边数),用存在度为 1 的结

点作起点调用 Bfs,求出其他非根结点的深度,在各次调用 Bfs 算法中求最大深度,即为树的直径.时间  $O(n(n+e))=O(n^2+ne)$  这里  $O(ne)$  是一次外部调用 Bfs 的运行时间,最多调用 Bfs 次

数(指外部调用)不超过  $O(n)$ (存储结构为邻接表时)

解法三:用邻接表作为存储结构

依次删去树叶(度为一的结点),将与树叶相连的结点度数减 1.设在第一轮删去原树 T 的所有

树叶后,所得树为 T1;再依次做第二轮删除,即删除所有 T1 的叶子;为此反复,若最后剩下一个

结点,则树直径应为删除的轮数\*2.具体算法如下:

```
m:=0;
for i:=1 to n do
  if (du(v[i])=1)then{
    enqueue(Q,v[i]); //叶子 vi 入队    m:=m+1; //m 记录当前一轮叶子数
  }
r:=0; //表示删除叶子轮数
while m>=2 do { //当前叶子轮数
  j:=0; //j 计算新一轮叶子数目
  for i:=1 to m do{
    dequeue(Q,v); //出队,表示删去叶子 v 将与 v 相邻的结点 w 的度数减 1
    if du(w)=1 then { //w 是新一轮的叶子
      j:=j+1;
      enqueue(Q,w); //w 入队
    }
  }
  r:=r+1; //轮数加 1
  m:=j; //新一轮叶子总数
}
if m=0 then return 2*r-1 //m=0,直径为轮数*2-1
else return 2*r; // m=1,直径为轮数*2
```

NOTE:假定 T 结点数 > 1, 否则在算法首部判定处理之。