

复旦大学计算机科学技术学院

《数据结构》期末考试试卷（参考答案与评分标准）

A 卷 共 8 页

课程代码: COMP130004.01-03 考试形式: ☐开卷 ☒闭卷 2012 年 1 月
(本试卷答卷时间为 120 分钟, 答案必须写在试卷上, 做在草稿纸上无效)

专业_____学号_____姓名_____成绩_____

题号	一	二	三	四	总分
得分					

一、填空题 (25%,1~8 题每题 2 分)

- 1、 设 W 为一个二维数组, 其每个数据元素占用 4 个字节, 行下标 i 从 0 到 7 , 列下标 j 从 0 到 3 , 则二维数组 W 的数据元素共占用_____个字节。若按行顺序存放二维数组 W, 其起始地址为 100(10 进制), 则二维数组元素 W[6, 3]的起始地址为_____ (10 进制表示)。

答: 128, 208。

- 2、 后缀算式 $9\ 2\ 3\ +\ -\ 10\ 2\ /\ -$ 的值为_____。中缀算式 $(3+4X) - 2Y/3$ 对应的后缀算式为_____。

答: -1, $3\ 4\ X\ * +\ 2\ Y\ * 3\ /\ -$ 。

- 3、 由权值分别为 11, 8, 6, 2, 5 的叶子结点生成一棵哈夫曼树, 它的带权路径长度为_____。

答: 71

- 4、 在有序表 (12, 24, 36, 48, 60, 72, 84) 中二分查找关键字 72 时所需进行的关键字比较次数为_____。

答: 2

- 5、 在含 n 个顶点和 e 条边的无向图 (无自环、重边) 的邻接矩阵中, 零元素的个数为_____。

答: n^2-2e

- 6、 已知一棵完全二叉树中共有 768 结点, 则该树中共有_____个叶子结点。

答: 384

- 7、 有向图的边集为{(a, c), (a, e), (e, b), (e, d), (b, d), (d, c), (c, f)}, 该图的一个拓扑排序为:_____

答: a e b d c f

- 8、 当输入序列局部有序或元素个数较小时, 在快速排序、选择排序、插入排序、归并排序、堆

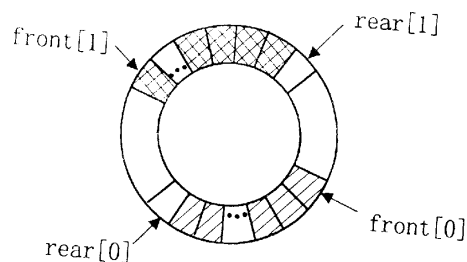
排序中，最佳的排序方法是_____。

答：插入排序

9、假设两个队列共享一个循环向量空间（参见右图），

其类型 Queue2 定义如下：

```
typedef struct{
    DataType data[MaxSize];
    int front[2],rear[2];
}Queue2;
```



对于 $i=0$ 或 1 ， $front[i]$ 和 $rear[i]$ 分别为第 i 个队列的头指针和尾指针。请对以下算法填空，实现第 i 个队列的入队操作。

```
int EnQueue (Queue2 *Q, int i, DataType x)
{//若第 i 个队列不满，则元素 x 入队列，并返回 1；否则返回 0
    if(i<0||i>1)return 0;
    if(Q->rear[i] == Q->front[_____]) return 0;
    Q->data[_____] = x;
    Q->rear[i] = [_____];
    return 1;
}
```

答： $(i+1)\%2$ (或 $1-i$)； $Q->rear[i]$ ； $(Q->rear[i]+1)\%Maxsize$ ；（每空 1 分）

10、以下是一个判断二叉树是否平衡的程序，对于平衡的二叉树，`depth` 将会返回其高度（对于非平衡二叉树不要求返回）。请在空白处填上代码完成程序。（6 分）

```
bool isBalanced( BiTreeNode * pRoot, int& depth)
{
    if(pRoot == NULL) {_____; return _____;}
    int leftDepth, rightDepth;
    if(_____)
    {
        int dif = leftDepth - rightDepth;
        if(_____)
        {
            depth = (1+leftDepth) > (1+ rightDepth) ? 1+leftDepth : 1+ rightDepth;
            return _____;
        }
    }
    return false;
}
```

}

答: depth = 0 (1分) true(1分)

isBalanced(pRoot->left, leftDepth) && isBalanced(pRoot->right, rightDepth) (2分)

dif >= -1 && dif <= 1 (1分)

true (1分)

二、选择题 (10%)

1、对于双向循环链表,每个结点有两个指针域 next 和 prior,分别指向前驱和后继。在 p 指针所指向的结点之后插入 s 指针所指结点的操作应为 ()

A. p->next = s; s->prior = p; p->next->prior = s; s->next = p->next;

B. p->next = s; p->next->prior = s; s->prior = p; s->next = p->next;

C. s->prior = p; s->next = p->next; p->next = s; p->next->prior = s;

D. s->prior = p; s->next = p->next; p->next->prior = s; p->next = s;

答: D。

2、一个栈的输入序列为 1 2 3,则下列序列中不可能是栈的输出序列的是()

A. 2 3 1

B. 3 2 1

C. 3 1 2

D. 1 2 3

答: C。

3、采用开放定址法处理散列表的冲突时,其平均查找长度 ()。

A. 低于链接法处理冲突

B. 高于链接法处理冲突

C. 与链接法处理冲突相同

D. 高于二分查找

答: B。

4、假设一个有 n 个顶点和 e 条弧的有向图用邻接表表示,则删除与某个顶点 v_i 相关的所有弧的时间复杂度是()

A. $O(n)$

B. $O(e)$

C. $O(n+e)$

D. $O(n*e)$

答: C。

5、设有 6 个结点的无向图,该图至少应有()条边才能确保是一个连通图。

A.5

B.6

C.7

D.8

答: A

三、问答题 (40%)

1、设一棵 m 叉树中有 N_1 个度数为 1 的结点, N_2 个度数为 2 的结点, ..., N_m 个度数为 m 的结点,则该树中共有多少个叶子结点? (请写出求解过程) (5 分)

答：度数为 i 的结点的孩子个数为 $i \times N_i$ 。除了根结点，其他结点都是某结点的孩子，且父亲结

点唯一。因此该树共有 $\text{total} = 1 + \sum_{i=1}^m i \times N_i$ 个结点。

度数大于 0 的结点都是非叶子结点，因此内部结点个数 $\text{inner} = \sum_{i=1}^m N_i$ 。

叶子结点个数 $\text{leaf} = \text{total} - \text{inner} = 1 + \sum_{i=1}^m (i-1) \times N_i = 1 + \sum_{i=2}^m (i-1) \times N_i$

评分标准：总分 5 分，分析过程正确或者部分正确为 1-5 分，否则为 0 分。

2、一个线性表为 $B = (12, 23, 45, 57, 20, 03, 78, 31, 15, 36)$ ，设散列表为 $\text{HT}[0..12]$ ，散列函数为 $H(\text{key}) = \text{key} \% 13$ 并用线性探查法解决冲突，请画出散列表，并计算等概率情况下查找成功的平均查找长度。（5 分）

答：

0	1	2	3	4	5	6	7	8	9	10	11	12
78		15	03		57	45	20	31		23	36	12

查找成功的平均查找长度： $\text{ASL}_{\text{succ}} = 14/10 = 1.4$

（散列表 4 分，查找成功的平均查找长度 1 分）

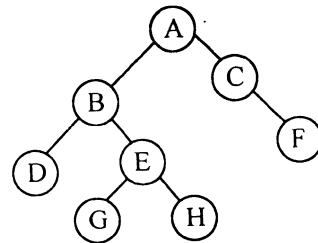
3、已知二叉树的存储结构为二叉链表，阅读下面算法。

```
typedef struct node {
    DataType data;
    Struct node * next;
}ListNode;
typedef ListNode * LinkList;
LinkList Leafhead = NULL;
Void Inorder (BinTree T){
    LinkList s;
    If(T){
        Inorder(T->lchild);
        If ((!T->lchild)&&(!T->rchild)){
```

```

        s=(ListNode*)malloc(sizeof(ListNode));
        s->data=T->data;
        s->next=Leafhead;
        Leafhead=s;
    }
    Inorder(T->rchild);
}

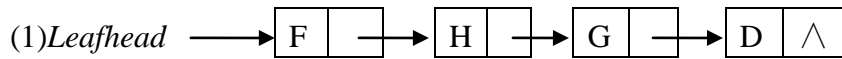
```



对于如下所示的二叉树

- (1) 画出执行上述算法后所建立的结构；(4 分)
- (2) 说明该算法的功能。(2 分)

答:

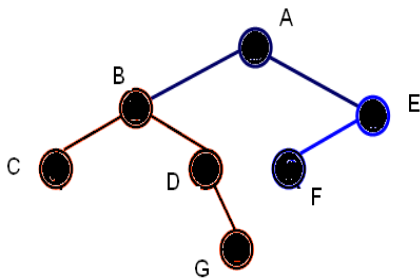


(2) 中序遍历二叉树，按遍历序列中叶子结点数据域的值构建一个以 `Leafhead` 为头指针的逆序单链表（或按二叉树中叶子结点数据自右至左链接成一个链表）。

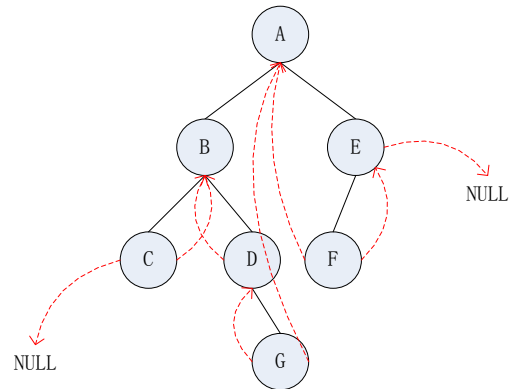
- 4、一棵二叉树的先序序列为 ABCDGEF，中序序列为 CBDGAFE。请画出该二叉树并将其中序线索化，后将二叉树转换为相应的森林。(5 分)

答:

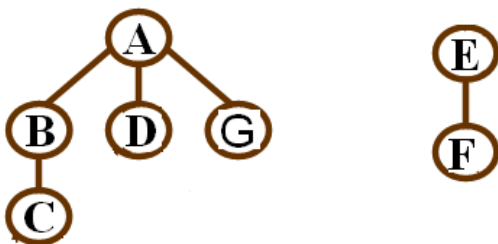
(二叉树: 1 分):



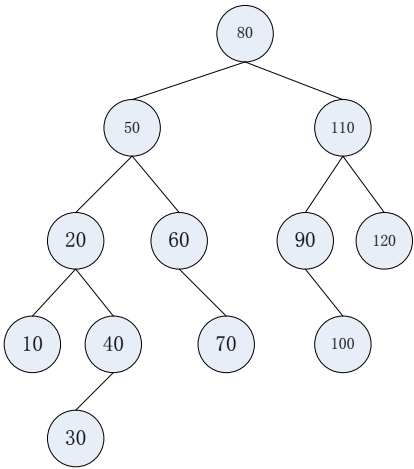
(中序线索: 2 分):



(森林: 2 分):

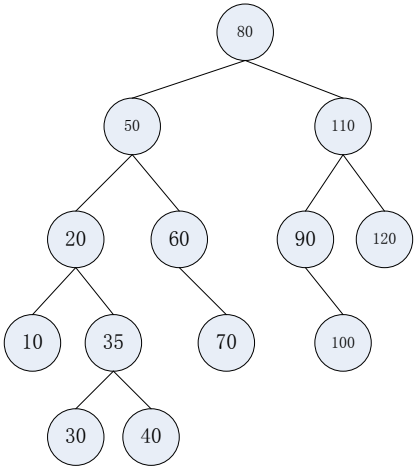


5、对于以下 AVL 树，依次插入关键码 35、28、5。请画出每插入一个关键码之后 AVL 树的形状。
(6 分)

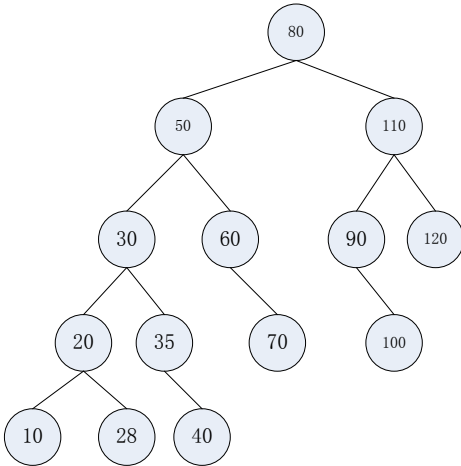


答:

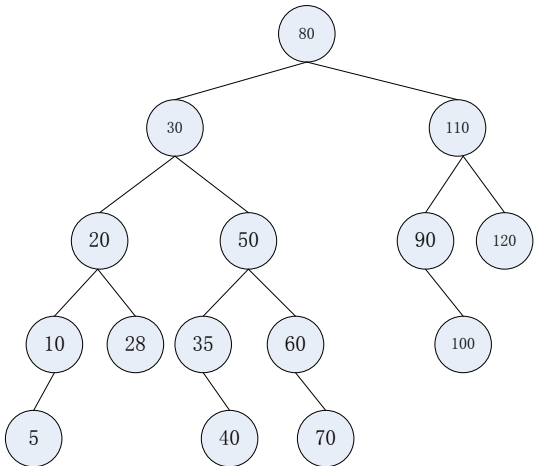
插入 35 (2 分):



插入 28 (2 分):



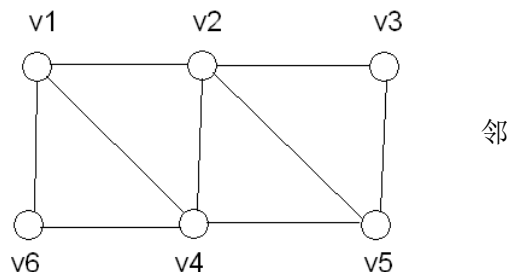
插入 5 (2 分):



评分标准：总分 6 分，每次插入后的形状 2 分。

6、已知一个连通图如下图所示，试给出图的邻接矩阵和邻接表存储示意图，若从顶点 v1 出发对该图进行遍历，分别给出一个按深度优先遍历和广度优先遍历的顶点序列。(8 分)

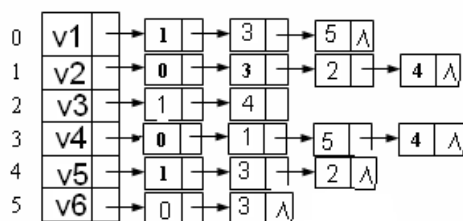
- (1) 图的邻接矩阵 (1 分)
- (2) 邻接表存储示意图 (1 分)
- (3) 从 v1 开始的深度优先遍历的顶点序列 (2 分)
- (4) 分析在深度遍历过程中，分别使用邻接矩阵和接表存储的算法复杂度 (2 分)
- (6) 讨论在图遍历问题中，这两种存储方式的优劣。(2 分)



答：(1)图的邻接矩阵 (1 分)

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

(2) 邻接表存储示意图 (1 分)



(3) 深度优先遍历的顶点序列: v1,v2,v3, v5,v4, v6 (2 分) (有多个答案)

(4) 深度优先邻接矩阵复杂度: $O(n^2)$, n 为顶点数; 邻接表复杂度: $O(n+e)$, n 为顶点数, e 为边的条数。(2 分)

(5)在图的遍历算法中，遍历一个顶点的邻居，对于邻接矩阵，时间复杂度为 $O(n)$ ，而邻接表只需要该顶点的邻居数量复杂度；因此对于整个图的遍历，如深度优先、广度优先等，邻接矩阵需要 $O(n^2)$ ，而邻接表时间复杂度为 $O(n+e)$ ，当图为稀疏图时，邻接表的时间复杂度优于邻接矩阵，而且存储开销也优于邻接矩阵；当图为密集图时，两者性能差不多；总的来说，邻接表的时间复杂度和空间存储开销都优于邻接矩阵。(2 分)

7、分析对比 AVL 树和 Hash 的时空特性，并讨论它们的适合场合。(6 分)

答：

时空特性:

AVL 树是高度平衡的二叉查找树, 查找时间复杂度为 $O(\log n)$, Hash 的查找时间复杂度为 $O(1)$ 。
存储开销 Hash 往往比 AVL 树小。

适合场合:

Hash 必须处理冲突, 而 AVL 树不存在这种问题。对于删除操作, AVL 树的时间开销很稳定, 为 $O(\log n)$, 而 Hash, 如果采用拉链法处理冲突, 则删除操作易于实现, 如果采用开放定址法, 则删除操作很难实现, 因此开放定址法的 Hash 不适合更新频繁的数据存储。另外, AVL 树对数据的存储是有序的, 而 Hash 对数据的存储并不能反映数据之间的大小关系。因此, AVL 树适用于有序的数据存储, 而 Hash 适用于数据量比较大且分布比较均匀, 对数据排序无要求的数据存储。

四、算法题 (25%, 第一题要求写代码; 后两题要求写伪代码或步骤清晰的解题思路)

1、堆是一种很常用的数据结构, 其中的一个重要用途是优先队列。以整数最小堆为例, 写出优先队列 (最小优先) 的两个基本操作的代码: (10 分)

(1) 入队, 即插入一个元素到存放堆的数组的末尾, 然后调整堆。

(2) 出队, 即删除堆顶节点, 然后调整堆。

void insert(int k, int a[], int n) { //假设新的元素 k 先被放在堆末尾位置 a[n]单元中

答: n++;
int j = n-1;
int i = (j-1)/2;
while(j > 0)
{
if(a[i] <= k)
break;
a[j] = a[i];
j = i;
i = (i-1)/2;
}
a[j] = k;

}

int deleteMin(int a[], int n) { //假设在堆顶元素被删除之前, 堆中共有 n 个元素

答: a[0] = a[n-1];
n--;
int i = 0;
int temp = a[i];
while((j = 2*i + 1) < n)
{
if(j < n-1 && a[j+1] < a[j])
j++;
if(a[j] >= temp)
break;
a[i] = a[j];
i = j;


```

    }
    a[i] = temp;
}

```

- 2、堆的另外一个常用场合是求 top k 问题。假设我们已经有一个 n 个元素的最小堆，如果用最直观的方法求最小的 k 个元素，其计算复杂度为 $O(k\log n)$ 。试问：如何在更小的复杂度内求 top k 问题？用伪代码描述你的方法，并给出复杂度证明。(7 分)

答：

假设原最小堆为 H，新建一个最小堆 H'，H' 初始化为空。

将最小堆 H 的堆顶元素 a 插入 H'；然后以下步骤执行 k 次：

将 H' 的堆顶元素 a' 删除并输出，然后将 a' 在 H 中的两个孩子插入 H' 中。

为了快速找到 a' 在 H 中的两个孩子，H' 存储的实际上是元素在 H 中的下标，而大小比较通过下标读取 H 中对应的元素。

由于 k 次对 H' 的删除操作，最多将 2 个新元素插入 H' 中，因此 H' 的大小不超过 2k。每次插入删除操作复杂度都为 $O(\log 2k)$ ，因此算法复杂度为 $O(k\log k)$ 。

评分标准：满分： $O(k\log k)$ 或者更好。若提出了快速排序的 partition 函数 ($O(n)$ 复杂度)，给 4 分。

- 3、给定一个无向图 $G(V, E)$ ，V 为顶点集合，E 为边集合。对于 V 中的两个顶点 u 和 v，如果图中存在一条 u 到 v 的路径，则称 u 和 v 彼此可达。设计一种数据结构，将图 G 进行预处理得到该数据结构，使得对于任意 u 和 v，通过该数据结构可以在常数时间得到 u 和 v 是否彼此可达。要求该数据结构存储开销为 $O(|V|)$ ，预处理复杂度为 $O(|V| + |E|)$ 。(8 分)

注意——图中的结点从 0 开始用整数连续编号。图 Graph 上的操作有：取得结点总数 `int GetVertexNum()`；取得 u 之后的 v 的下一个邻接顶点 `int GetNextNeighbor(int v, int u)`，u 为 -1 则取得 v 的第一个邻接顶点，返回 -1 表示没有下一个邻接顶点了；可以直接使用数组、链表、栈和队列这些基本数据结构及其上的基本操作，但要注释每个所使用操作的含义。

- (1) 描述所设计的数据结构，描述如何得到该数据结构，并写出预处理的代码。
- (2) 描述如何通过该数据结构在常数时间内得到 u 和 v 是否可达，并写出相应代码。

答： 本题考察学生对连通分量的掌握和运用。使用 BFS 算法得到图的 N 个连通分量，为每个连通分量分配一个 id。用一个包含 |V| 个元素的数组 `connectedCom[|V|]`，记录每个顶点属于的连通分量 id，即 `connectedCom[u]` 表示 u 属于的联通分量。对于 u 和 v，若 `connectedCom[u] = connectedCom[v]`，则 u 和 v 彼此可达。

评分标准： 本题答案不唯一，可以是 BFS、DFS，也可以是并查集。

总共 8 分。

思路 2 分——思路清晰正确为 2 分，方法不限于参考答案这一种，思路不完全正

确最多为 1 分，没写为 0 分；算法 1-6 分。