**2014年春季北大《数据结构与算法B》期末考试模拟试卷**

**（本试卷只是给同学们看看考题形式和范围，难度与真实考卷稍有差别）**

**学号\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 姓名\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 教师/教室\_\_\_\_\_\_\_\_\_\_\_\_\_**\_

（注：如未标明，本试卷题中的下标、位置都从0开始计数）

1. 填空题（共32分）
2. 设有字符串变量String A = “This”, B=“is”, C=“just”, D=“a˽test”，请计算下列表达式：

（1）A+B+D=\_“Thisisjusta˽test”\_；

（2）D.IndexOf(“t”) = \_\_2\_\_\_；（求字符在字符串中的第一个位置）

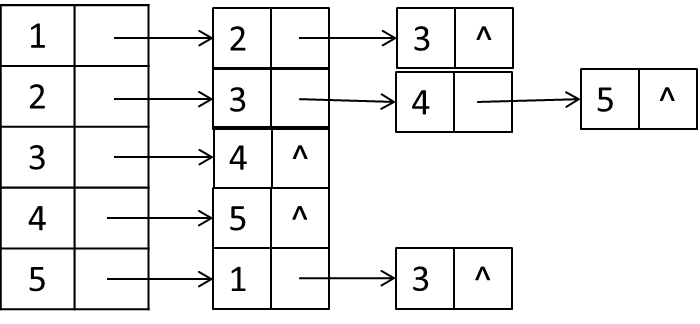
（3）B.Strlength() = \_\_\_2\_\_\_

（4）D.SubStr(1,2) = \_“˽t” \_\_（注：1为起始下标，2为子串长度）【4分】

1. 顺序查找n个元素的顺序表，若查找成功，则比较关键字的次数最多为\_\_\_n\_\_\_次；若查找失败，则比较关键字的次数最多为\_\_\_n\_\_\_，最少为\_\_\_\_n\_\_\_次。【3分】
2. 在散列函数H（key）=key%p中，p值最好取\_\_\_质数（或素数）\_\_\_。【1分】
3. 对于下图邻接表所对应的有向图，试写出：【2分】

(1) 从顶点①出发进行深度优先遍历结果\_\_1, 2, 3, 4, 5\_\_；

(2) 从顶点①出发进行广度优先遍历结果\_\_1, 2, 3, 4, 5 \_；



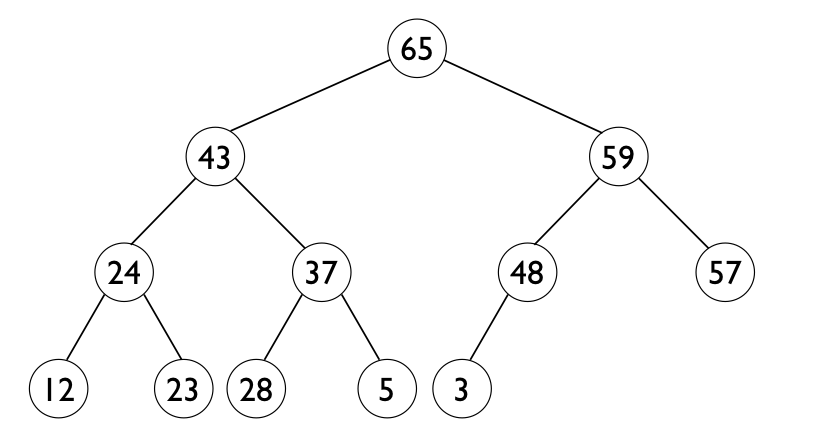
1. 当图中各条边上的权值 \_\_都相等\_\_ 时，宽度优先搜索算法可用来解决单源最短路径问题？【2分】
2. 一棵有n个结点的满二叉树有\_0\_个度为1的结点、有\_(n-1)/2个分支 （非终端）结点；该满二叉树的深度最大为\_\_\_(n+1)/2\_\_, 最小为 int(log2n)或⎣log2n⎦。（独根树深度为0）【4分】
3. 对于给定的n个元素,可以构造出的逻辑结构有 线性结构 ， 树形结构 ， 图形结构 ，\_集合\_\_四种。【2分】
4. 下面程序段的时间复杂度为\_\_O(n)\_。(n>1) [大O表示法] 【2分】

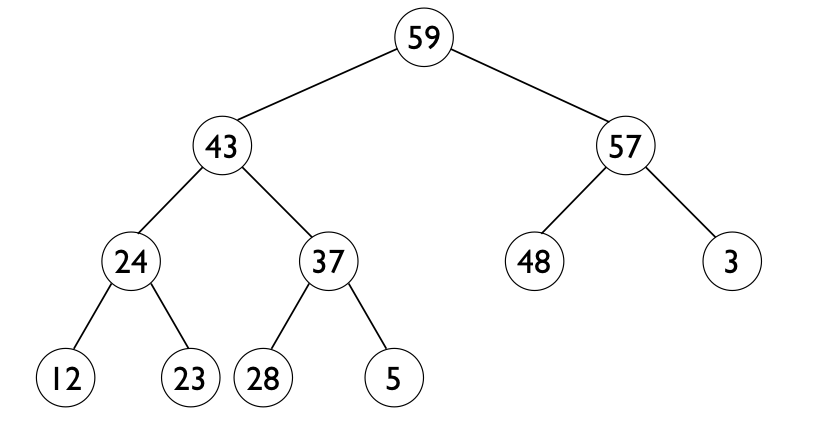
sum=1；

for (i=0;sum<n;i++) sum+=1;

对于最大堆 65 43 59 24 37 48 57 12 23 28 5 3，删除掉最大元素后，堆中元素为

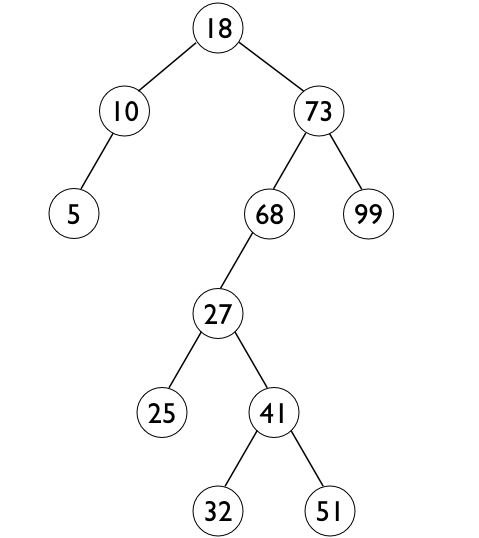
12 23 24 28 5 37 43 48 3 57 59【2分】





1. 从空二叉树开始，严格按照二叉搜索树的插入算法（不进行旋转平衡），逐个插入关键码 { 18,73,10,5,68,99,27,41,51,32,25 } 构造出一棵二叉搜索树，该二叉树转换为森林，则该森林的层次遍历序列为 \_18 73 99 10 68 5 27 41 51 25 32\_\_【4分】

对于的二叉树为



1. 用S表示入栈操作，X表示出栈操作，若元素入栈的顺序为12345，为了得到13542出栈顺序，相应的S和X的操作串为\_SXSSXSSXXX\_\_。【2分】
2. 单选题（18分，每题2分，最后两题每题4分）

1. 对初始状态为递增的表按递增顺序排序，最省时间的是（ C ）算法。

A. 基数排序 B. 桶排 C. 直接插入排序 D. 归并排序

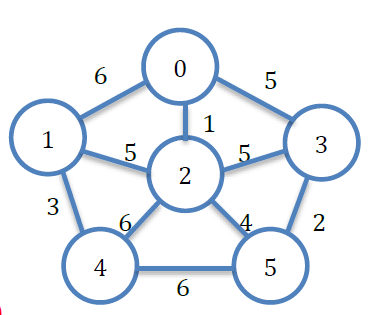
2. 一个n个顶点的连通无向图，其边的个数至少为（A）。

A．n-1 B．n C．n+1 D．nlogn；

3． 线性表（ a1,a2,…,an）以链接方式存储时，访问第i位置元素的时间复杂度为（ C ）

A．O（i） B．O（1） C．O（n） D．O（i-1）

4. 使用Prim算法从结点0出发求下图的最小生成树，依次写出每次被加入到最小生成树中边的编号，下面正确的答案是（B）。



A. (0, 2) (3, 5) (1, 4) (2, 5) (1, 2) B. (0, 2) (2, 5) (3, 5) (1, 2) (1, 4)

C. (0, 2) (3, 5) (1, 4) (1, 2) (2, 5) B. (0, 2) (1, 2) (1, 4) (2, 5) (3, 5)

5．一个有n个结点的图，最多有（ D ）个连通分量。

A．0 B．1 C．n-1 D．n

6. 用二分（对半）查找法检索元素速度比用顺序法( D ) 。【2分】

1. 必然快 B. 必然慢 C. 相等 D. 不能确定

7. 已知一算术表达式的中缀形式为 A+B\*C-D/E，后缀形式为ABC\*+DE/-，其前缀形式为( D )。【4分】

A．-A+B\*C/DE B. -A+B\*CD/E C．-+\*ABC/DE D. -+A\*BC/DE

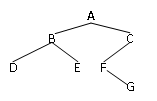
8．对序列{15，9，7，8，20，-1，4}进行排序，进行一趟处理（对应于排序算法中一层循环或一层递归算法）后，数据的排列变为{4，9，-1，8，20，7，15}；则采用的是（ C ）排序。【4分】

A. 选择 B. 快速 C. 希尔（Shell） D. 冒泡

1. 简答题（30分，每题10分）
2. 试问由二叉树的中序序列和后序序列是否能唯一的建立二叉树，请说明理由；若能，对中序序列DBEAFGC和后序序列DEBGFCA构造二叉树。

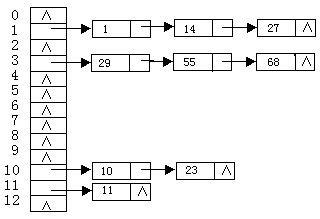
答：（1）给定二叉树结点的后序序列和对称序（中序）序列，可以唯一确定该二叉树。因为后序序列的最后一个元素是根结点，该元素将二叉树中序序列分成两部分，左边（设有*l*个元素）表示左子树，若左边无元素，则说明左子树为空；右边（设有r个元素）是右子树，若为空，则右子树为空。根据后序遍历中“左子树—右子树—根­”的顺序，则后序序列中由从第*1*元素开始的l个结点序列和中序序列根左边的*l*个结点序列构造左子树；由后序序列*l*个元素后面r个元素序列与中序序列根右边的r个元素序列构造右子树。

（2）中序序列DBEAFGC和后序序列DEBGFCA构造的二叉树如下图



2．设一组数据为{1,14,27,29,55,68,10,11,23}，现采用的散列函数是H(key)=key % 13，冲突用链地址法解决，设散列表的大小为13(下标为0~12)，试画出插入上述数据后的散列表。

答：



3. 设T是一棵二叉树，除叶子结点外，其它结点的度数皆为2，若 T中有6个叶结点，试问：

（1）T树的最大深度Kmax=?最小可能深度Kmin=?（假定独根二叉树深度为0）

（2）T树中共有多少非叶结点？

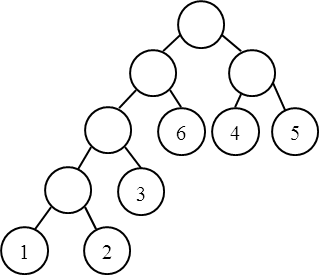
（3）若叶结点的权值分别为1,2,3,4,5,6。请画出一棵哈曼夫树，并计算该哈曼夫树的带权路径长度wpl。

答：1）T树的最大深度Kmax=5（除根外，每层均是两个结点）

T树的最小深度Kmin=4（具有6个叶子的完全二叉树是其中的一种形态）

（2）非叶子结点数是5。（n2=n0-1）

（3）哈夫曼树见下图，其带权路径长度wpl=51



1. 算法填空题（20分，第一题10分，第2题10分）

1. 下面是求二叉树高度（独根树高度是1）的递归算法，试补充完整

二叉树的两指针域为lchild与rchild, 算法中p为二叉树的根，lh和rh分别为以p为根的二叉树的左子树和右子树的高，hi为以p为根的二叉树的高，hi最后返回。

int height(BinTree \*p)

{ if (\_\_\_p \_\_\_\_) // p!=NULL 也对

{ if (p->lchild==null)

lh=\_\_\_0\_\_\_\_;

else lh=\_\_\_height(p->lchild)\_\_\_\_;

if (p->rchild==null)

rh=\_\_0\_\_\_\_\_;

else rh=\_\_ height(p->rchild) \_;

if (lh>rh)

hi=\_\_lh+1\_\_\_；

else hi=\_\_\_rh+1\_\_\_;

}

else hi=\_\_\_0\_\_\_\_;

return hi;

}

2．对单链表（带头结点）中元素按插入方法实现非递增序列的排序的C语言描述算法如下，其中L为链表头结点指针。请填充算法中标出的空白处，完成其功能。

typedef struct node

{ int data;

struct node \*next;

} linknode,\*link;

void Insertsort(link L)

{ link p,q,r,u;

p=L->next; \_\_L->next = NULL\_\_\_\_;

while(\_\_p != NULL \_\_)

{ r=L; q=L->next;

while(q != NULL && q->data <= p->data) {

r=q; q=q->next;

}

u=p->next; \_\_\_p->next=r->next(或p->next = q)\_\_\_; \_\_\_r->next = q\_\_\_; p=u;

}

}

【注：后面两空也可以是r->next = p; p->next = q;此时一定不能写成p->next = r->next】