52《数据结构》

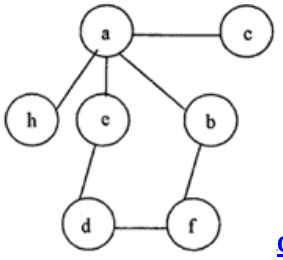
单选(735)--

1、Char*p；p=StrCat(“ABD“,“ABC“)；Printf(“%“,p)；的显示结果为（）。-->ABDABC
2、e5、e1,则栈 S 的容量至少应该是（）。-->3
3、e6 依次通过栈 S,一个元素出栈后即进入队列 Q,若 6 个元素出队的顺序是 e2、e4、e3、e6、e5、e1,则栈 S 的容量至少应该是（）。-->3
4、G 是一个非连通无向图，共 28 条边，则该图至少有（）个顶点。-->D.9
5、m 阶 B-树是一棵（）。-->m 叉平衡排序树
6、n(n≥2)个权值均不相同的字符构成哈夫曼树,关于该树的叙述中,错误的是（）。-->该树一定是一棵完全二叉树
7、n 个顶点的连通图用邻接距阵表示时,该距阵至少有（）个非零元素。-->2（n-1）。
8、n 个顶点的强连通图的形状是（）。-->环状
9、n 个顶点的强连通图至少有（）条边。-->n
10、n 个顶点的强连通图中至少含有（）。-->B.n 条有向边
11、n 个结点的二叉树中，用二叉链表做存储，非空指针数目为（）。-->C.n-1
12、（）遍历二叉排序树可得到一个有序序列-->主关键字
13、（）不属于线性表的基本操作。-->B.求子表
14、（）查找是一种最简单的查找方法-->顺序
15、（）查找又称为二分查找。使用该查找算法的前提条件是,查找表中记录相应的关键字值必须按升序或降序排列-->折半
16、（）查找只适用于顺序存储结构的有序表-->折半
17、（）的一个重要应用是解决主机和打印机之间速度不匹配的问题。-->D.队列
18、（）的一个重要应用是在程序设计中实现递归调用。-->C.栈
19、（）的一个重要应用是在程序设计中实现递归调用。-->C.栈
20、（）是按关键字的非递减或非递增顺序对一组记录重新进行排列的操作-->关键字
21、（）是记录某个数据项的值,用它可以识别、确定一个记录-->关键字
22、（）是数据的基本单位,在计算机中通常作为一个整体进行考虑和处理。用于完整地描述一个对象,如一个学生记录,树中棋盘的一个格局(状态)、图中的一个顶点等-->数据元素
23、（）是数据的基本单位。-->数据元素
24、（）是相互之间存在一种或多种特定关系的数据元素的集合-->数据结构
25、（）是性质相同的数据元素的集合,是数据的一个子集。-->数据对象
26、（）是由用户定义的,表示应用问题的数学模型,以及定义在这个模型上的一组操作的总称。具体包括三部分：数据对象、数据对象上关系的集合和对数据对象的基本操作的集合-->抽象数据类型
27、（）是组成数据元素的最小单位。-->数据项
28、（）数据类型独立于具体实现,将数据与操作封装在一起,实现了信息隐藏-->抽象
29、（）有两个指针域，分别指向直接前驱和直接后继，可以实现从前向后和从后向前查找。 -->D.双向循环链表
30、把数据存储到计算机中 并具体体现（）称为物理结构。-->数据元素间的逻辑关系
31、把数据存储到计算机中,并具体体现数据元素间的逻辑结构称为（）。-->物理结构
32、把数据存储到计算机中,并具体体现数据元素间的逻辑结构称为（）。-->存储结构
33、把一棵树转换为二叉树后,这棵二叉树的形态是（）。-->唯一的
34、表达式 3*（x+y）/（2-x）的后缀表达式是（）。-->D.3xy+*2x-/
35、表达式 8/5+4 的后缀表达式是（）。-->D.85/4+
36、表达式 a*(b+c)-d 的后缀表达式是（）。-->B.abc+*d-
37、表达式 a*（b+c）-d 的后缀表达式是（）。-->B.abc+*d-
38、表达式 a*（b+c）-d 的后缀表达式是（）。-->abc+*d-
39、不考虑计算机的软硬件环境因素,影响算法时间代价的最主要的因素是（）。-->问题规模
40、部分以行序为主序存储到一维数组 B 中(数组下标从 1 开始),则矩阵中元素缸．3 在一维数组 B 中的下标是（）。-->9
41、采用邻接表存储的图，其深度优先遍历类似于二叉树的（）。-->B.先序遍历
42、采用顺序查找法对长度为 n 的线性表进行查找(不采用表尾设监视哨的方法),最坏的情况下要进行（）次元素间的比较。-->n
43、采用顺序查找方法查找长度为 n 的线性表时，每个元素的平均查找长度为（）。-->C.（n+1）/2
44、采用线性探测法处理冲突,可能要探测多个位置,在查找成功的情况下,所探测的这些位置上的关键字（）。-->不一定都是同义词
45、采用折半查找方法查找长度为 n 的线性表时,其算法的时间复杂度为（）。-->D.0(1og2n)。
46、**采用折半查找方法查找长度为 n 的线性表时,其算法的时间复杂度为（）。** 0(1og2n)
47、采用折半查找方法查找长度为 n 的线性表时，每个元素的平均查找长度为（）。-->D.log2n
48、常对数组进行的两种基本操作是（）。-->C.查找和修改
49、程序段 chara[]=“Englih“；char*p=a；intn=O；while(*p!=‘/0')(n十+；P++；)结果中,P 指向（）。-->字符串的结束符
50、抽象数据类型相当于在概念层上描述问题,而类相当于在（）层上描述问题-->实现
51、串“ababaaababaa”的 next 数组为（）。-->011234223456
52、串”ababaabab”的 nextval 为（）。-->010104101
53、串的长度是指（）。-->D.串中所含非空格字符的个数
54、串的长度是指（）。-->B.串中所含字符的个数
55、串函数 index(a,b)的功能是进行（）。-->C.模式匹配
56、串函数 strcat(a,b)的功能是进行串（）-->D.连接
57、串函数 StrCmp(“ABCd“,“ABCD”的值为（）。-->1
58、串函数 StrCmp(“ahA“,“aha 勺的值为（）。-->-1
59、串函数 StrCmp(“b“,“cd”)的值为（）。-->一 1
60、串函数 StrCmp（“ABCd”，“ABCD”）的值为（）。-->C.1
61、**串函数 Stremp（a，b）的功能是进行串（）**A.比较
62、串是（）。-->D.有限个字符的序列
63、串是什么？（）。-->C.有限个字符的序列
64、串是一种特殊的线性表,其特殊性体现在（）。-->数据元素是一个字符
65、串是一种特殊的线性表,其特殊性体现在（）。-->数据元素是单个字符
66、串与普通的线性表相比较,它的特殊性体现在（）。-->数据元素是一个字符
67、创建一个包括 n 个结点的有序单链表的时间复杂度是（）。-->O（n2）。
68、从 n 个数中选取最大元素（）。-->算法的时间复杂度是 O（n）。
69、从未排序序列中挑选元素,并将其放人已排序序列的一端,此方法称为（）。-->C.选择排序
70、从未排序序列中挑选元素,并将其依次放入已排序序列(初始时为空)的一端的方法,称为（）。-->选择排序
71、**从未排序序列中挑选元素，并将其放人已排序序列的一端，此方法称为（）。**C.选择排序
72、从未排序序列中挑选元素，并将其放入已排序序列的一端，此方法称为（）排序。-->C.选择排序
73、从未排序序列中依次取出元素与已经排好序的序列中的元素作比较。将其放人已排序序列的正确的位置上,此方法称为（）。-->A.插入排序
74、从未排序序列中依次取出元素与已经排好序的序列中的元素作比较。将其放入已排序序列的正确的位置上，此方法称为（）。-->A.插入排序
75、从一个顺序存储的循环队列中删除一个元素时，首先需要（）。-->A.队头指针加一
76、从一个栈顶指针为 top 的链栈中删除一个结点时，用变量 x 保存被删结点的值，则执行（）。-->A.x=top-data；top=top-next；

77、存储结构是数据对象在计算机中的存储表示,也称为（）。-->物理结构
78、存储结构由顺序存储结构和（）存储结构两种基本的存储方法-->链式
79、带头结点的单向链表 L 为空的判定条件是（）。
-->B.L-next==NULL
80、带头结点的单向链表为空的判断条件是()(设头指针为 head)。
-->D.head-next==NULL
81、带头结点的链表为空的判断条件是（）(设头指针为 head)。
-->head-next==NULL
82、带头结点的双向循环链表 L 为空表的条件是()。-->L-next==L
83、带头结点的双向循环链表 L 为空表的条件是()。-->L-next==L
84、单链表的存储密度（）。-->小于 1
85、单向链表所具备的特点之一是（）．-->插入元素和删除元素的操作不需要移动元素
86、单向线性链表的结点包含 data 域和（）域。-->A.next
87、单循环链表的主要优点是（）。-->从表中任一结点出发都能扫描到整个链表
88、**当利用大小为 100 的数组顺序存储一个队列时，队列的最大长度为（）。** B.99
89、当利用大小为 n 的数组顺序存储一个队列时，该队列的最大长度为（）。-->B.n-1
90、当两个元素出现逆序的时候就交换位置，这种排序方法称为（）。-->B.交换排序
91、堆的形状是一棵（）。-->完全二叉树
92、堆排序的空间复杂度为 O（）-->1
93、堆是一种（）排序。-->选择
94、队列的插人操作在（）进行。-->队尾
95、队列的出队操作在（）进行。-->A.队头
96、队列是一种操作受限的线性表，其限制是（）。-->D.仅允许在表的一端进行插入，而在另一端进行删除操作
97、对 22 个记录的有序表作折半查找,当查找失败时,至少需要比较（）次关键字。-->4
98、对 n 个不同的关键字由小到大进行冒泡排序,在下列（）情况下比较的次数最多。-->从大到小排列好的
99、对 n 个不同的排序码进行冒泡排序,在元素无序的情况下比较的次数最多为（）。-->n（n-1）/2
100、对 n 个关键字作快速排序,在最坏情况下,算法的时间复杂度是（）。-->O（n2）。
101、对 n 个元素的表做顺序查找时,若查找每个元素的概率相同,则平均查找长度为（）。-->（n+1）/2
102、对 n 个元素进行冒泡排序,通常要进行 n-l 趟冒泡,在第 j 趟冒泡中共要进行（）次元素间的比较。-->j-l
103、对 n 个元素进行冒泡排序若某趟冒泡中只进行了（）次元素间的交换,则表明序列 yi 经排好序。-->0
104、对（）进行中序遍历,可以使遍历所得到的序列是有序序列。-->二叉排序树
105、对不带头结点的单向链表,判断是否为空的条件是（）(设头指针为 head)。-->head==NULL
106、对二叉排序树进行（）遍历，可以使遍历所得到的序列是有序序列。-->C.中序
107、对二叉树的结点从 1 开始进行连续编号,要求每个结点的编号大于其左、右孩子的编号,同一结点的左右孩子中,其左孩子的编号小于其右孩子的编号,可采用（）遍历实现编号。-->后序
108、对记录序列排序是指按记录的某个关键字排序,记录序列按()关键字排序结果是唯一的-->主
109、对具有 n 个元素的任意序列采用插入排序法进行排序，排序趟数为（）。-->A.n-1
110、对链表,以下叙述中正确的是（）。-->不能随机访问任一结点
111、对数据元素序列{49,72,68,13,38,50,97,27}进行排序,前三趟排序结果时的结果依次为第一趟：49,72,68,13,38,50,97,27；第二趟：49,68,72,13,38,50,97,27；第三趟：13,49,68,72,38,50,97,27。该排序采用的方法是（）。-->插入排序法
112、对顺序表,以下叙述中正确的是（）。-->用一组地址连续的存储单元依次存放线性表的数据元素
113、对特殊矩阵采用压缩存储的目的主要是为了（）。-->减少不必要的存储空间
114、对稀疏矩阵进行压缩存储,可采用三元组表,一个 10 行 8 列的稀疏矩阵 A．其相应的三元组表共有 6 个元素,矩阵 A 共有（）个零元素。-->74
115、对稀疏矩阵进行压缩存储,可采用三元组表,一个 10 行 8 列的稀疏矩阵 A 共有 73 个零元素,A 的右下角元素为 6,其相应的三元组表中的第 7 个元素是（）。-->（10，8，6）
116、对稀疏矩阵进行压缩存储,可采用三元组表,一个 10 行 8 列的稀疏矩阵 A 共有 73 个零元素,其相应的三元组表共有()个元素。-->7
117、对稀疏矩阵进行压缩存储,可采用三元组表,一个有 10 行的稀疏矩阵 A 共有 97 个零元素,其相应的三元组表共有 3 个元素。该矩阵 A 有（）列。-->10
118、对线性表进行二分查找时，要求线性表必须（）。-->C.以顺序存储方式，且数据元素有序
119、对序列（49，38，65，97，76，13，47，50）采用直接插入排序法进行排序，要把第七个元素 47 插入到已排序中，为寻找插入的合适位置需要进行（）次元素间的比较。-->C.5
120、对一个钱项指针为 top 的链找进行人找操作,通过指针变量 p 生成人找结点,则执行：p=(structnode*)malloc(sizeof(structnode)；p->data=a；和（）。-->p-nex=top；top=p；
121、对一个战项指针为 top 的链钱进行进校操作,设 P 为待进榜的结点,则执行（）。-->p 一 next=top；top=p；
122、对一个栈项指针为 top 的链栈进行出栈操作,用变量 e 保存栈顶元素的值,则执行（）。-->e=top-data；top=top-next；
123、对一个栈项指针为 top 的链栈进行人栈操作,通过指针变量 p 生成人栈结点,则可执行操作：
p-(structnode*)mallac(sizef(structnode)：p->data=a 和（）．-->p-next=head
124、对一个栈项指针为 top 的链栈进行入栈操作,通过指针变量 p 生成入栈结点,并给该结点赋值 a,则执行-->p-next=top；top=p；
125、对一个栈项指针为 top 的链栈进行入栈操作,通过指针变量生成入栈结点,则可执行操作：
p=(structnode*)malloc(sizeof(structnode)；p—>data=a；和（）。-->p-next=top；top=p
126、对一棵二叉树顺序编号，若一个结点是双亲结点的左孩子，双亲结点的编号为 i，则这个结点的右孩子结点的编号为（）。-->D.4i+1
127、对有 18 个元素的有序表作二分查找，则查找 A[3]的比较序列的下标可能为（）。-->D.9、4、2、3
128、对于具有 n 个顶点的图,若采用邻接矩阵表示,则该矩阵的大小为（）。-->B.n2
129、对于具有 n 个顶点的图，若采用邻接矩阵表示，则该矩阵的大小为（）。-->B.n2
130、对于顺序存储的有序表(5,12,20,26,37,42,46,50,64),若采用折半查找,则查找元素 26 的比较次数是（）。-->4
131、对于一个具有 4 个顶点和 5 条边的无向图,若采用邻接表表示,则所有顶点邻接表中的结点总数为（）。-->D.10
132、对于一个具有 n 个顶点和 e 条边的无向图,若采用邻接表表示,则所有顶点邻接表中的结点总数为（）。-->D.2e
133、对于一个具有 n 个顶点和 e 条边的无向图，若采用邻接表表示，则所有顶点邻接表中的结点总数为（）。-->D.2e
134、对于一个具有 n 个元素的线性表，建立其单向链表的时间复杂度为（）。-->D.O（n）。
135、对于一个链串 s，查找第一个字符值为 x 的算法的时间复杂度为（）-->B.O（n）。
136、对于一个满二叉树，m 个树叶，n 个结点，深度为 h，则（）。-->D.n=2h-1
137、对于一个线性表,若要求既能进行较快地插入和删除,又要求存储结构能够反映数据元素之间的逻辑关系,则应该（）。-->B.以链接存储方式
138、对于一棵具有 n 个结点的树，该树中所有结点的度数之和为（）。-->C.n-1
139、二叉排序树的查找效率与二叉树的（）有关。-->C.树形
140、二叉树的按层遍历算法需要使用（）-->A.队列
141、二叉树的先序遍历和中序遍历如下：先序遍历：EFHI 对 K 中序遍历：HFI 对 KG 该二叉树根的右子树的根是（）。-->C.G
142、二叉树第 k 层上最多有（）个结点。-->B.2k-l
143、二叉树第 k 层上最多有（）个结点。-->B.2k-1
144、二维数组 A 的每个元素是由 10 个字符组成的串,其行下标 i=0,1,…,8,列下标 j=1,2,…,10。若 A 按行先存储,元素 A[8,5]的起始地址与当 A 按列先存储时的元素（）的起始地址相同。设每个字符占一个字节。-->A[3,10]
145、二维数组 A 的每个元素是由 6 个字符组成的串,行下标的范围从 0~8,列下标的范围是从 0~9,则存放 A 至少需要（）个字节。-->540
146、非空的单向循环链表的尾结点满足（）（设头指针为 head，指针 p 指向尾结点）。-->A.p-next==head
147、分别以下列序列构造二叉排序树,与用其它三个序列所构造的结果不同的是（）。-->（100,60,80,90,120,110,130）。
148、根据排序过程中所用的存储器不同,可以将排序方法分为（）排序和外部排序-->内部
149、关键路径是事件结点网络中（）。-->A.从源点到汇点的最长路径
150、关于队列的队头和队尾操作说法正确的是（）。-->插入在队尾进行,删除在队头进行

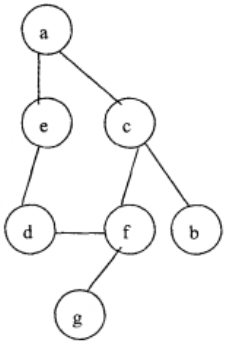
151、关于栈的说法正确的是（）。-->栈是一种先进后出的线性表
152、关于栈和队列的说法中，错误的是（）。-->D.栈是先进先出，队列是后进先出
153、关于栈和队列的说法中，错误的是（）。-->栈是先进先出，队列是后进先出
154、广度优先遍历类似于二叉树的（）。-->层次遍历
155、广义表((a,b,c,d))的表头是（）。-->（a,b,c,d）。
156、广义表(a,b,(c,(d)))的表尾是（）。-->（b,（c,（d））），。
157、广义表(f,h,(a,b,d,c),d,e,((i,j),k))的长度是（）。-->6
158、广义表(f,h,(a,b,d,c),d,e,《i,j),k))的长度是（）。-->A.6
159、广义表 A=(a,b,(c,d),(e,(f,g))),则 Head(Tail(Head(Tail(Tail(A))))) 的值为（）。-->d
160、广义表（a,（d,a,b）,h,（e,《i,j）,k））深度是（）。-->D.4
161、广义表（a，a，b，d，e，《i，j），k））的表头是（）。-->B.a
162、广义表（a，d，e，（i，j），k）的表尾是（）。-->B.（d，e，（i，j），k）。
163、广义表（f,h,（a,b,d,c）,d,e,《i,j）,k））的长度是（）。-->A.6
164、广义表的(a,(d,a,b),h,(e,《i,j),k)))深度是（）。-->D.4
165、广义表的（a,（d,a,b）,h,（e,《i,j）,k））深度是（）。-->D.4
166、哈夫曼树是（）。-->D.带权路径长度最短的二叉树
167、哈希表的平均查找长度（）。-->A.与处理冲突的方法有关，与表的长度无关
168、哈希函数有一个共同的性质，即函数值应当以（）取其值域的每个值。-->D.同等概率
169、基数排序的空间复杂度为 O（）-->n+rd
170、假定一棵二叉树中，双分支结点数为 15，单分支结点数为 30，则叶子结点数为（）。-->B.16
171、假定一组记录的排序码为(46,79,56,38,40,80),对其进行归并排序的过程中,第二趟归并后的结果为（）。-->C.38,46,56,79,40,80
172、假设存放循环队列的数组长度为 MaxSize,循环队列能装入的元素最大个数（）-->B.MaxSize-1
173、假设存放循环队列的数组长度为 MaxSize，循环队列能装入的元素最大个数为（）。-->B.MaxSize-1
174、假设以行序为主序存储二维数组 A=array[1…100,1…100],设每个数据元素占 2 个存储单元,基地址为 10,则 LOC[5,5]=()。-->818
175、将含有 150 个结点的完全二叉树从根这一层开始,每一层从左到右依次对结点进行编号,根结点的编号为 1,则编号为 69 的结点的双亲结点的编号为（）。-->34
176、将含有 86 个结点的完全二叉树从根结点开始编号,根为 1 号,后面按从上到下、从左到右的顺序对结点编号,那么编号为 3 的右孩子编号为（）-->7
177、将含有 86 个结点的完全二叉树从根结点开始编号,根为 1 号,后面按从上到下、从左到右的顺序对结点编号,那么编号为 42 的左孩子编号为（）-->84
178、将两个各有 n 个元素的有序表归并成一个有序表,其最少的比较次数是（）。-->n
179、将树中结点赋上一个有着某种意义的实数,称此实数为该结点的（）-->权
180、将数组称为随机存取结构是因为（）。-->对数组任一元素的存取时间是相等的
181、结构中的元素之间存在一对多的关系是（）。-->树形结构
182、静态查找与动态查找的根本区别在于（）。-->施加在其上的操作不同
183、就排序算法所用的辅助空间而言，堆排序、快速排序、归并排序的关系是（）。-->堆排序快速排序归并排序
184、具有 127 个结点的完全二叉树其深度为（）。-->B.7
185、具有 n 个顶点的无向图最多有（）条边。-->n（n-1）/2
186、可以直接对数组元素进行的操作有（）。-->D.读取
187、空串与空格串（）。-->B.不相同
188、空串与空格串（）。-->B.不相同
189、快速排序在下列（）情况下最易发挥其长处。-->被排序的数据完全无序
190、利用 12 这四个值作为叶子结点的权,生成一棵哈夫曼树,该树中所有叶子的最长带权路径长度为（）。-->18
191、利用 2、4、5、10 这四个值作为叶子结点的权，生成一棵哈夫曼树，该树中所有叶子的最长带权路径长度为（）。-->C.38
192、利用 3、6、8、12 这四个值作为叶子结点的权,生成一棵哈夫曼树,该树中所有叶子结点中的最长带权路径长度为（）。-->18
193、利用 n 个值作为叶结点的权生成的哈夫曼树中共包含有（）个结点。-->D.2*n-1
194、链表不具有的特点是（）。-->可随机访问任一元素
195、链表所具备的特点是（）。-->C.插入删除元素的操作不需要移动元素结点
196、链表所具备的特点是（）。-->插入删除元素的操作不需要移动元素结点
197、链表所具备的特点之一是（）。-->C.插入元素的操作不需要移动元素
198、链接存储的存储结构所占存储空间（）-->分两部分,一部分存放结点值,另一部分存放表示结点间关系的指针
199、链接存储结构中的数据元素之间的逻辑关系是由()表示的。-->指针
200、链式存储的存储结构所占存储空间（）。-->分为两部分,一部分存放结点值,另一部分存放表示结点间关系的指针
201、链式栈结点为(data,link),top 指向栈顶,若想摘掉栈顶结点,并将删除结点的值保存到 x 中,则应执行操作（）。-->x=top-data；top=top-link；
202、链栈和顺序栈相比，有一个比较明显的优点，即（）。-->B.通常不会出现栈满的情况
203、两个字符串相等的条件是（）。-->D.两个串的长度相等且对应位置的字符相同
204、邻接表是图的一种（）。-->B.链式存储结构
205、冒泡排序的空间复杂度为 O（）-->1
206、冒泡排序是一种比较简单的（）排序方法-->交换
207、冒泡排序在最好情况下的时间复杂度为 O（）-->n
208、每次把待排序的区间划分为左、右两个子区间，其中左区间中记录的关键字均小于等于基准记录的关键字，右区间中记录的关键字均大于等于基准记录的关键字，这种排序称为（）。-->B.快速排序
209、每个存储结点不仅含有一个数据元素,还包含一组指针,该存储方式是（）存储方式。-->B.链接
210、每个存储结点只存储一个数据元素，各结点存储在连续的存储空间，该存储方式是（）存储方式。-->A.顺序
211、每个结点只包含一个指针域的线性表叫（）-->单链表
212、描述网络中的结点,用（）结构比较合适。-->图
213、某串的长度小于一个常数，则采用()存储方式最节省空间。-->B.顺序
214、某串的长度小于一个常数，则采用()存储方式最节省空间。-->顺序
215、某二叉树的先序遍历序列和后序遍历序列正好相反，则该二叉树一定（）。-->D.深度等于其结点数
216、排序方法中,从尚未排序序列中挑选元素,并将其依次放入 yi 排序序列(初始为空)的一端的方法,称为（）排序。-->选择
217、排序过程中,每一趟从无序子表中将一个待排序的记录按其关键字的大小放置到 yi 经排好序的子序列的适当位置,直到全部排好序为止,该排序算法是（）。-->直接插入排序
218、排序算法理想的空间复杂度为（）-->1
219、排序算法中,从未排序序列中依次取出元素与 yi 排序序列(初始为空)中的元素进行比较(要求比较次数尽量少),然后将其放人已排序序列的正确位置的方法是（）。-->折半插入
220、判断顺序栈 s 满(元素个数最多 n 个)的条件是（）。-->C.s-top==n-l
221、判断向上增长型的顺序栈空的条件是（）。-->D.top=-1
222、判断向上增长型的顺序栈空的条件是（）。-->top=-1
223、判断一个顺序队列(最多元素为 m)为空的条件是（）。-->front==rear
224、判断一个顺序队列 sq(最多元素为 m)为空的条件是（）-->C.sq-front==sq-rear
225、判断一个顺序队列 sq（最多元素为 m）为空的条件是（）。-->C.sq-front==sq-rear
226、判断一个循环队列 Q(最多元素为 m)为满的条件是（）。-->C.Q-front==(Q-rear+l)%m
227、判断一个循环队列 Q（最多元素为 m）为满的条件是（）。-->C.Q-front==（Q-rear+1）%m
228、判断一个循环队列为满的条件是（）。-->（rear+1）%MaxSize==front
229、判断栈 s 满（元素个数最多 n 个）的条件是()。-->C.s-top==n-1
230、权值为{1，2，6，8}的四个结点构成的哈夫曼树的带权路径长度是（）。-->D.29
231、任何一棵二叉树的叶子结点在前序、中序、后序遍历序列中的相对次序（）。-->肯定不发生改变
232、如果从无向图的任一顶点出发进行一次深度优先搜索即可访问所有顶点，则该图一定是（）。-->B.连通图
233、如果待排序的记录数目很大,无法一次性调入内存,整个排序过程就必须借助外存分批调入（）才能完成-->内存
234、如果对线性表进行删除第一个元素，删除最后一个元素，在第一个元素前面插入新元素，在最后一个元素的后面插入新元素这四种运算，则最好使用（）。-->C.只有头结点指针没有尾结点指针的双向循环链表
235、如果将给定的一组数据作为叶子数值，所构造出的二叉树的带权路径长度最小，则该树称为（）。-->A.哈夫曼树
236、如果结点 A 有 3 个兄弟,B 是 A 的双亲,则结点 B 的度是()。-->4
237、如果进行串的比较，下列哪个串最大？（）-->A.“BEIJING”

238、如果要描述家族关系,用（）比较合适。-->树
239、如果要求一个线性表既能较快的查找,又能适应动态变化的要求,最好采用（）查找法。-->分块查找
240、如果以链表作为栈的存储结构，则退栈操作时（）。-->C.必须判断栈是否空
241、**如图所示,若从顶点 a 出发,按图的广度优先搜索法进行道历,则可能得到的一种顶点序列为（）**



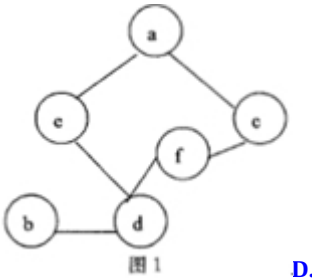
C.ahebcdf

242、**如图所示的一个图,若从顶点 a 出发,按深度优先搜索法进行遍历,则可能得到的一种顶点序列为（）。**



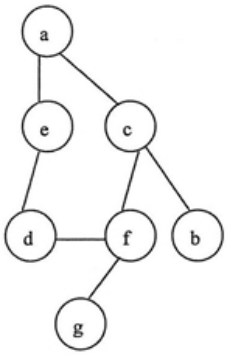
D.aedfcbg

243、**如图所示的一个图,若从顶点 a 出发,按深度优先搜索法进行遍历,则可能得到的一种顶点序列为（）。**



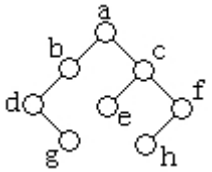
D.aedbfc

244、**如图所示的一个图，若从顶点 a 出发，按深度优先搜索法进行遍历，则可能得到的一种顶点序列为（）。**



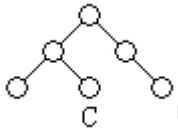
D.aedfcbg

245、**如图所示二叉树的中序遍历序列是（）。**



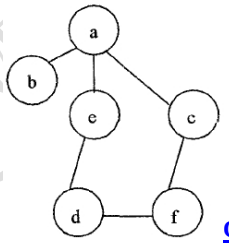
B.dgbaechf

246、**如图所示一棵二叉树中，（）不是完全二叉树。**



C

247、**如下图所示,若从顶点 a 出发,按图的广度优先搜索法进行遍历,则可能得到的一种顶点序列为（）。**



C.aecbdf

248、若 Head 为一个不带表头结点的循环单链表的表头指针，若有 Head->next==Head 条件存在，则该循环单链表是（）。-->B.只有 1 个元素

249、若 Head 为一个带表头结点的单链表的表头指针，则该表为空表的条件是（）。-->B.Head-next==NULL

250、**若 HL 为一个不带表头结点的循环单链表的表头指针，若有 HL->next==HL 条件存在，则该循环单链表是（）。**

B.只有 1 个元素；

251、若 X 是二叉中序线索树中一个有左孩子的结点,且 X 不为根,则 X 的前驱为（）。-->X 的左子树中最右结点

252、若串 S=“English”，其子串的个数是（）。-->D.28

253、若从无向图的任意一个顶点出发进行一次深度优先搜索可以访问图中所有的顶点,则该图一定是（）图。-->连通

254、若对 n 阶对称矩阵 A 以行序为主序方式将其下三角形的元素(包括主对角线上所有元素)依次存放于一维数组 B[1. . (n(n+1))/2]中,则在 B 中确定 aij(i<J)的位置 K 的关系为(td)<)。(难度系数：易>-->j*（j-1）/2+i

255、若二叉树采用二叉链表存储结构,要交换其所有分支结点左、右子树的位置,利用（）遍历方法最合适。-->后序

256、若某表最常用的操作是在最后一个结点之后插入一个结点，则采用（）最节省运算时间。-->D.带头结点的双向循环链表

257、若某线性表中最常用的操作是取第 i 个元素和找第 i 个元素的前趋,则采用（）存储方法最节省时间。-->顺序表

258、若让元素 1,2,3,4,5 依次进栈,则出栈次序不可能出现在（）种情况。-->4,3,1,2,5

259、若让元素 1,2,3 依次进栈,则出栈顺序不可能为（）。-->C.3,1,2

260、若让元素 1，2，3 依次进栈，则出栈顺序不可能为（）。-->C.3，1，2

261、若一个栈的输入序列是 1,2,3,…,n,输出序列的第一个元素是 n,则第 i 个输出元素是（）。-->n-i+1

262、若一个栈以向量 V[1. . n]存储,初始栈顶指针 top 设为 n+1,则元素 x 进栈的正确操作是（）。-->top--；V[top]=x；

263、若一组记录的排序码为(46,79,56,38,40,84),则利用堆排序的方法建立的初始堆为（）。-->84,79,56,38,40,46

264、若一组记录的排序码为(46,79,56,38,40,84),则利用快速排序的方法,以第一个记录为基准得到的一次划分结果为（）。-->40,38,46,56,79,84

265、若已知一个栈的入栈序列是 1,2,3,…,n,其输出序列为 p1,p2,p3,…,pn,若 p1=n,则 pi 为（）。-->n-i+1

266、散列查找的原理是（）。-->在待查记录的关键字值与该记录的存储位置之间建立确定的对应关系

267、散列技术中的冲突指的是（）。-->不同键值的元素对应于相同的存储地址

268、设 F 是一个森林,B 是由 F 变换得的二叉树。若 F 中有 n 个非终端结点,则 B 中右指针域为空的结点有（）个。-->n+1

269、设 G1=（V1,E1）和 G2=（V2,E2）为两个图，如果 V1 [illegible] V2,E1) E2>-->A.G1 是 G2 的子图

270、设 head 为非空的单向循环链表头指针,P 指向一链表的尾结点,则满足逻辑表达式（）的值为真。-->P 一 next==head

271、设 n、m 为一棵二叉树上的两个结点，中序遍历时 n 在 m 前的条件是（）。-->C.n 在 m 左方

272、设 S=“I_am_a_teacher”,其长度为（）。-->14

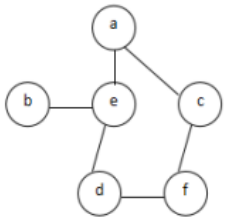
273、设 top 是一个链栈的栈顶指针,栈中每个结点由一个数据域 data 和指针域 next 组成,设用 x 接收栈顶元素,则出栈操作为（）。-->x=top 一 data；top=top 一 next；

274、设 top 是一个链栈的栈顶指针,栈中每个结点由一个数据域 data 和指针域 next 组成,设用 x 接收栈顶元素,则取栈顶元素的操作为（）。-->x=top 一 data；

275、设单向链表中,指针 p 指向结点-->p 一 next=p-next 一 next;
276、设二叉树有 n 个结点,则其深度为（）。-->不能确定
277、设二叉树中有 n2 个度为 2 的结点，n1 个度为 1 的结点，n0 个叶子结点，则此二叉树中空指针域个数为（）。-->D.2n0+n1
278、设二维数组 A[1．．m,1．．n](即 m 行 n 列)按行存储在数组 B[1..m*n]中,则二维数组元素 A[i,j]在一维数组 B 中的下标为()。-->（i-1）*n+j
279、设二维数组 A[5][6]按行优先顺序存储在内存中，已知 A[0][0]起始地址为 1000，每个数组元素占用 5 个存储单元，则元素 A[4][4]的地址为（）。-->A.1140
280、设根结点所在层为第一层,一棵具有 5 层的完全二叉树,最后一层有 6 个结点,则该总共有（）个结点。-->21
281、设广义表 L=((a,b,c)),则 L 的长度和深度分别为（）。-->1 和 2
282、设广义表 L=((a,b,c)),则 L 的长度是（）。-->1
283、设广义表 L=((a,b,c)),则 L 的深度是（）。-->2
284、设哈夫曼树的叶结点数为 n，则它的结点总数为()。-->A.2n-1
285、设哈夫曼树中有 199 个结点,则该哈夫曼树中有（）个叶子结点。-->100
286、设哈希表长 m=14，哈希函数 H（key）=keymod11。表中已有 4 个结点：addr（15）=4；addr（38）=5；addr（61）=6；addr（84）=7。如用线性探测处理冲突，关键字为 49 的结点的地址是（）。-->A.8
287、设哈希表长为 14,哈希函数是 H(key)=key%11,表中已有数据的关键字为 15,38,61,84 共四个,现要将关键字为 49 的元素加到表中,用二次探测法解决冲突,则放入的位置是（）。-->9
288、设计一个判别表达式中左,右括号是否配对出现的算法,采用（）数据结构最佳。-->栈
289、设链表中的结点是 NODE 类型的结构体变量,且有 NODE 头 P；为了申请一个新结点,并由 p 指向该结点,可用以下语句（）。-->p=（NODE*）malloc（sizeof（NODE））。
290、设某一二叉树先序遍历为 abdec,中序遍历为 dbeac,则该二叉树后序遍历的顺序是（）。-->debca
291、设顺序存储的线性表长度为 n,对于删除操作,设删除位置是等概率的,则删除一个元素平均移动元素的次数为（）。-->（n+1）/2
292、设顺序存储的线性表长度为 n,要删除第 1 个元素,按课本的算法,当 i=（）时,移动元素的次数为 3。-->n-3
293、设顺序存储的线性表长度为 n,要在第 i 个元素之前插入一个新元素,按课本的算法当 i=（）时,移动元素次数为 20-->n-l
294、设头指针为 head 的不带头结点的非空的单向循环链表,指针指向尾结点,要删除第一个结点,使它仍为单向循环链表,则可利用操作：head=head->next；和（）。-->p-nex
295、设头指针为 head 的非空的单向链表，指针 p 指向尾结点,则通过以下操作（）可使其成为单向循环链表。-->D.p-next=head;
296、设一棵采用链式存储的二叉树,除叶结点外每个结点度数都为 2．该树结点中共有 20 个指针域为空,则该树有（）个叶结点。-->10
297、设一棵采用链式存储的二叉树,除叶结点外每个结点度数都为 2．该树结点中共有 20 个指针域为空。则该树共有（）个非叶子结点-->9
298、设一棵二叉树中没有度为 1 的结点，已知叶子结点数为 n，此树的结点数为（）。-->D.2n-1
299、设一棵哈夫曼树共有 14 个非叶结点,则该树总共有（）个结点。-->29
300、设一棵哈夫曼树共有 2n 十 1 个结点,则该树有（）个非叶结点。-->n
301、设一棵哈夫曼树共有 2n 十 1 个叶结点,则该树有（）个叶结点。-->n+l
302、设一棵哈夫曼树共有 31 个结点,则该树共有（）个非叶子结点。-->B.15
303、设一棵哈夫曼树共有 n 个非叶结点,则该树有（）个结点。-->2n+1
304、设一棵哈夫曼树共有 n 个非叶结点,则该树有（）个叶结点。-->n+1
305、设一棵晗夫曼树共有 15 个非叶结点,则该树总共有（）个结点。-->31
306、设一棵有 2n+1 个结点的二叉树,除叶结点外每个结点度数都为 2,则该树共有（）个叶结点。-->n+1
307、设一棵有 n 个叶结点的二叉树,除叶结点外每个结点度数都为 2,则该树共有（）个结点。-->2n-1
308、设一棵有个叶结点的二叉树,共有 2n+2 个结点,则该二叉树中度数为 1 的结点数有（）个。-->5
309、设已有 m 个元素有序，在未排好序的序列中挑选第 m+1 个元素，并且只经过一次元素的交换就使第 m+1 个元素排序到位，该方法是（）。-->D.直接选择排序
310、设有 2000 个无序的元素，希望用最快的速度挑选出其中前 10 个最大的元素，最好选用（）排序法。-->D.堆排序
311、设有两个串 p 和 q,其中 q 是 p 的子串,求 q 在 p 中首次出现的位置的算法称为（）。-->C.模式匹配
312、设有两个串 p 和 q，其中 q 是 p 的子串，q 在 p 中首次出现的位置的算法称为（）。-->C.匹配
313、设有数据结构(D,R),其 D={d1,d2,d3,d4},R={<D1,D4>,<D1,D3>,<D3,D2>}。那么这个数据结构是（）。-->树结构
314、设有数组 A[i,j],数组的每个元素长度为 3 个字节,i 的值为 1~8,j 的值为 1~10,数组从内存首地址 BA 开始顺序存放,当用以列为主序存放时,元素 A[5,8]的存储首地址为（）。-->BA+180
315、设有数组 A[i,j],数组的每个元素长度为 3 字节,i 的值为 1 到 8,j 的值为 1 到 10,数组从内存首地址 BA 开始顺序存放,当用以列为主存放时,元素 A[5,9]的存储首地址为（）。-->BA+183
316、设有头指针为 head 的非空的单向链表,指针 p 指向其尾结点,要使该单向链表成为单向循环链表,则可利用下述语句（）。-->P 一 next=head;
317、设有一个 10 阶的对称矩阵-->b[13]
318、设有一个 10 阶的对称矩阵 A,采用压缩存储方式,以行序为主存储,a11 为第一元素,其存储地址为 1,每个元素占一个地址空间,则 a85 的地址为（）。-->33
319、设有一个 12 阶的对称矩阵 A,采用压缩存储方式将其下三角部分以行序为主序存储到一维数组 b 中(矩阵 A 的第一个元素为,数组 b 的下标从 1 开始),则矩阵 A 中第 4 行的元素在数组 b 中的下标 i 一定有（）。-->7≤i≤10
320、设有一个 17 阶的对称矩阵-->51
321、设有一个 17 阶的对称矩阵 AC 第一个元素为 a1,1 采用压缩存储的方式,将其下三角部分以行序为主序存储到一维数组 B 中(数组下标从 1 开始),则矩阵中元素 a5,2 在一维数组中的下标是()-->11
322、设有一个 18 阶的对称矩阵 A,采用压缩存储的方式,将其下三角部分以行序为主序存储到一维数组 B 中(数组下标从 1 开始),则矩阵中 a10,8 元素对应于数组中第（）号元素。(矩阵中的第 1 个元素是 a1,1)。-->53
323、设有一个 18 阶的对称矩阵 A,采用压缩存储的方式,将其下三角部分以行序为主序存储到一维数组 B 中(数组下标从 1 开始).则数组中第 33 号元素对应于矩阵中的元素是（）。(矩阵中的第 1 个元素是 al．l)。-->a8,5
324、设有一个 18 阶的对称矩阵 A,采用压缩存储的方式,将其下三角部分以行序为主序存到一维数组 B 中(数组下标从 1 开始),则矩阵元素队：对应于数组 B 中第（）号元索。阵中的第 1 个元素是 a1,)。-->38
325、设有一个 20 阶的对称矩阵 A(第一个元素为 a1,1),采用压缩存储的方式,将其下三角部分以行序为主序存储到一维数组 B 中(数组下标从 1 开始),则矩阵元素 a6,2 在一维数组 B 中的下标是()。-->17
326、设有一个 20 阶的对称矩阵 A,采用压缩存储的方式,将其下三角部分以行序为主序存储到一维数组 B 中(数组下标从 1 开始),则数组中第 38 号元素对应于矩阵中的元素是（）。(矩阵中的第 1 个元素是 a1,1)。-->a9,2
327、设有一个 20 阶的对称矩阵 A，采用压缩存储的方式，将其下三角部分以行序为主序存储到一维数组 B 中（数组下标从 1 开始），则矩阵中元素 a9,2 在一维数组 B 中的下标是（）。-->D.38
328、设有一个 25 阶的对称矩阵 A(第一个元素为 ai,,,采用压缩存储的方式,将其下三角部分以行序为主序存储到一维数组 B 中(数组下标从 1 开始),则矩阵中元素 a4,s 在一维数组 B 中的下标是()。-->B.9
329、设有一个 25 阶的对称矩阵 A,采用压缩存储的方式,将其下三角部分以行序为主序存储到一维数组 B 中(数组下标从 1 开始),则矩阵中元素．a7．6 在一维数组 B 中的下标是（）。(矩阵中的第 1 个元素是 a1．1)。-->27
330、设有一个 28 阶的对称矩阵 A(矩阵的第一个元素为 a1,1),采用压缩存储的方式,将其下三角部分以行序为主序存储到一维数组 B 中(数组下标从 1 开始),则数组中第 40 号元素对应于矩阵中的元素是（）。-->a9,4
331、设有一个 30 阶的对称矩阵 A(第一个元素为 al．l),采用压缩存储的方式,将其下三角部分以行序为主序存储到一维数组 B 中(数组下标从 1 开始),则矩阵中元素 a9,2 在一维数组 B 中的下标是（）。-->38
332、设有一个长度为 10 的顺序表,要在第 3 个元素之后插入一个元素,则需移动元素的个数为（）。-->C.7
333、设有一个长度为 18 的顺序表,要在第 4 个元素之前插入 1 个元素(也就是插入元素作为新表的第 4 个元素),则移动元素个数为（）。-->15
334、设有一个长度为 22 的顺序表,要删除第 8 个元素需移动元素的个（）。-->14

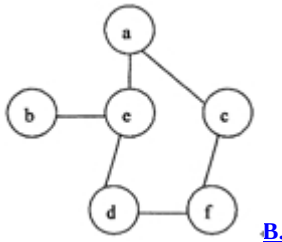
335、设有一个长度为 25 的顺序表,要删除第 10 个元素(下标从 1 开始),需移动元素的个数为（）。-->15
336、设有一个长度为 28 的顺序表,要在第 12 个元素之前插入一个元素(也就是插入元素作为新表的第 12 个元素),则移动元素个数为（）。-->17
337、设有一个长度为 32 的顺序表,要删除第 8 个元素需移动元素的个数为（）。-->24
338、设有一个长度为 32 的顺序表,要在第 5 个元素之前插入 1 个元素(也就是插入元素作为新表的第 5 个元素),需移动元素个数为（）。-->28
339、设有一个长度为 33 的顺序表,要删除第 10 个元素(下标从 1 开始)需移动元素的个数为（）。-->23
340、设有一个长度为 35 的顺序表,要在第 5 个元素之前插入 1 个元素(也就是插入元素作为新表的第 5 个元素),则移动元素个数为（）。-->31
341、设有一个长度为 40 的顺序表,要删除第 10 个元素(下标从 1 开始)需移动元素的个数为（）。-->30
342、设有一个长度为 n 的顺序表,要删除第 i 个元素移动元素的个数为（）。-->n-i
343、设有一个长度为 n 的顺序表,要在第 i 个元素之前(也就是插入元素作为新表的第 i 个元素),插入一个元素,则移动元素个数为（）。-->C.n-i+l
344、设有一个长度为 n 的顺序表，要删除第 i 个元素,则需移动元素的个数为（）。-->C.n-i
345、设有一个长度为 n 的顺序衰,要删除第 i 个元素需移动元素的个数为（）。-->n-i
346、设有一个带头结点的链队列,队列中每个结点由一个数据域 data 和指针域 next 组成,front 和 rear 分别为链队列的头指针和尾指针,要执行出队操作,用 x 保存出队元素的值,p 为指向结点类型的指针,可执行如下操作：p=front->next；x=p->data；然后执行（）。-->frant-next=p-next；
347、设有一个递归算法如下：intfact(intn)-->n-1
348、设有一个对称矩阵 A,采用压缩存储的方式,将其下三角部分以行序为主序存储到一维数组 B 中(数组下标从 1 开始),B 数组共有 55 个元素,则该矩阵是（）阶的对称矩阵。-->10
349、设有一个广义表 A（a），其表尾为（）。-->C.（）。
350、设有一个巧阶的对称矩阵 A(第一个元素为 a1,：),采用压缩存储的方式,将其下三角部分以行序为主序存储到一维数组 B 中(数组下标从 1 开始),则矩阵中元素 as,：在一维数组 B．中的下标是（）。-->13
351、设栈 S 和队列 Q 的初始状态为空,元素 e1、e2、e3、e4、e5、e6 依次通过栈 S,一个元素出栈后即进入队列 Q,若 6 个元素出队的顺序是 e2、e4、e3、e6、e5、e1,则栈 S 的容量至少应该是（）。-->3
352、设主串为“DBcCDABcdEFdBc”,以下模式串能与主串成功匹配的是（）。-->A.dBc
353、设主串为“FABcCDABcdEFaBc”，以下模式串能与主串成功匹配的是（）。-->A.EFaBc
354、深度为 5 的二叉树至多有（）-->C.31
355、深度为 5 的二叉树至多有（）个结点。-->C.31
356、深度为 5 的满二叉树至多有（）个结点(根结点为第一层)。-->31
357、深度为 5 的完全二叉树第 5 层上有 4 个结点,该树一共有（）个结点。-->19
358、深度为 5 的完全二叉树共有 20 个结点,则第 5 层上有（）个结点。(根所在层为第一层)。-->5
359、深度为 k 的完全二叉树至少有（）个结点。-->2k-1
360、深度优先遍历类似于二叉树的（）。-->先序遍历
361、是数据的基本单位,在计算机中通常作为一个整体进行考虑和处理。用于完整地描述一个对象,如一个学生记录,树中棋盘的一个格局(状态)、图中的一个顶点等-->数据元素
362、是相互之间存在一种或多种特定关系的数据元素的集合-->数据结构
363、是性质相同的数据元素的集合,是数据的一个子集-->数据对象
364、是组成数据元素的、有独立含义的、不可分割的最小单位。例如,学生基本信息表中的学号、姓名、性别等-->数据项
365、适用于折半查找的表的存储方式及元素排列要求为（）。-->顺序方式存储,元素有序
366、树的（）没有前驱结点，其他结点有且仅有一个直接前驱结点。-->A.根结点
367、树形结构中数据元素之间的关系是（）-->B.一对多
368、树中的结点数等于所有结点的度数加（）。-->A.1
369、树中所有结点的度等于所有结点数加（）。-->D.-1
370、树中所有结点数等于所有结点的度加（）。-->A.1
371、树最适合用来表示（）。-->C.元素之间具有分支层次关系的数据
372、数据表中有 10000 个元素,如果仅要求求出其中最大的 10 个元素,则采用（）算法最节省时间。-->堆排序
373、数据的（）结构与所使用的计算机无关。-->逻辑
374、数据的存储结构包括数据元素的表示和（）。-->D.数据元素间的关系的表示
375、数据的存储结构包括数据元素的表示和（）。-->数据元素间的关系的表示
376、数据的存储结构包括数据元素的表示和（）。-->D.数据元素间的关系的表示
377、数据的逻辑结构在计算机内存中的表示是（）。-->数据的存储结构
378、数据的物理结构（）。-->包括数据元素的表示和关系的表示
379、数据结构是（）。-->带有结构的数据元素的集合
380、数据结构是一门研究计算机中（）对象及其关系的科学。-->非数值运算
381、数据结构中，与所使用的计算机无关的是数据的（）。-->D.逻辑结构
382、数据元素是数据的基本单位,它（）。-->可以是一个数据项也可以由若干个数据项组成
383、数组 A[0…4,-1…-3,5…7]中含有元素的个数（）。-->45
384、数组 a 经初始化 chara[]=“English”；a[1]中存放的是（）。-->字符 n
385、数组 a 经初始化 chara[]=“English”；a[7]中存放的是（）。-->字符串的结束符
386、数组 Q［n］用来表示一个循环队列, f 为当前队列头元素的前一位置, r 为队尾元素的位置,假定队列中元素的个数小于 n ,计算队列中元素个数的公式为（）。-->（n+r-f）%n
387、双向线性链表的结点有（）域。-->C.3 个
388、双向循环链表结点的数据类型为：truCnode)intdata；struCnode*next；/,指向直接后继,/struCnode*prior；)；设 p 指向表中某一结点,要显示 p 所指结点的直接前驱结点的数据元素,可用操作（）。-->print《“%d“,p 一 prior 一 data）。
389、顺序表所具备的特点之一是（）。-->A.可以随机访问任一结点
390、顺序表中第一个元素的存储地址是 100,每个元素的长度为 2,则第 5 个元素的地址是（）。-->108
391、顺序查找法与二分查找法对存储结构的要求是（）。-->二分查找适用于顺序表
392、顺序查找法与折半查找法对存储结构的要求是（）。-->D.折半查找适用于顺序表
393、顺序查找方法适合于存储结构为（）的线性表。-->D.顺序存储或链接存储
394、顺序存储结构中的数据元素之间的逻辑关系是由（）表示的。-->存储位置
395、算法的时间复杂度取决于（）。-->A 和 B
396、算法的时间复杂度与（）有关。-->与算法本身
397、算法分析的两个主要方面是（）。-->空间性能和时间性能
398、算法分析的目的是（）。-->分析算法的效率以求改进
399、算法指的是（）。-->解决问题的有限运算序列
400、算法指的是（）。-->对特定问题求解步骤的一种描述,是指令的有限序列。
401、讨论树、森林和二叉树的关系,目的是为了（）。-->将树、森林按二叉树的存储方式进行存储并利用二叉树的算法解决树的有关问题
402、通常的使用顺序栈或者链栈实现递归算法，下面哪个说法正确（）。-->C.顺序栈和链栈性能基本相同
403、通常可以把某城市中各公交站点间的线路图抽象成（）状结构-->图
404、通常要求同一逻辑结构中的所有数据元素具有相同的特性,这意味着（）。-->不仅数据元素所包含的数据项的个数要相同,而且对应数据项的类型要一致
405、同一种逻辑结构（）。-->可以有不同的存储结构
406、头指针为 head 的不带头结点的单向链表为空的判定条件是逻辑表达式（）为真。-->head-==NULL
407、头指针为 head 的带头结点的单向链表为空的判定条件是（）为真。-->head-next==NULL
408、图的 BFS 生成树的树高比 DFS 生成树的树高（）。-->小或相等
409、图的深度优先遍历算法类似于二叉树的（）遍历。-->A.先序
410、图的深度优先遍历算法类似于二叉树的（）遍历。-->先序
411、图状结构中数据元素的位置之间存在（）的关系。-->多对多
412、图状结构中数据元素之间的关系是（）-->D.多对多
413、为解决计算机主机与打印机间速度不匹配问题,通常设一个打印数据缓冲区。主机将要输出的数据依次写入该缓冲区,而打印机

则依次从该缓冲区中取出数据。该缓冲区的逻辑结构应该是（）。-->队列
414、为了实现图的深度优先搜索遍历,其非递归的算法中需要使用的一个辅助数据结构为（）-->栈
415、无向图的邻接矩阵是一个（）。-->A.对称矩阵
416、稀疏矩阵采用压缩存储的目的主要是（）-->D.减少不必要的存储空间的开销
417、稀疏矩阵采用压缩存储的目的主要是（）。-->D.减少不必要的存储空间的开销
418、下列的叙述中，不属于算法特性的是（）。-->C.可读性
419、下列关键字序列中,（）是堆。-->16,23,53,31,94,72
420、下列关于串的叙述中,不正确的是（）。-->空串是由空格构成的串
421、下列广义表中的线性表是（）。-->C.E（a,b）。
422、下列排序算法中,（）不能保证每趟排序至少能将一个元素放到其最终的位置上。-->希尔排序
423、下列说法不正确的是（）。-->串不是线性结构
424、下列说法中,不正确的是（）。-->D.数据项可由若干个数据元素构成‘
425、下列有关图遍历的说法不正确的是（）。-->C.非连通图不能用深度优先搜索法
426、下面（）不是算法所必须具备的特性。-->高效性
427、下面（）不属于特殊矩阵。-->稀疏矩阵
428、下面（）可以判断出一个有向图中是否有环（回路）。-->B.拓扑排序
429、下面（）算法适合构造一个稠密图 G 的最小生成树。-->Prim 算法
430、**下面程序段的时间复杂度是（）。**
for（i=1；i＜=n；i++）
for（j=1；j＜=n；j++）{
c[i][j]=O；
for(k=l；k＜=n；k++)
c[i][j]=c[i][j]+a[i][k]*b[k][j]；
}
选：D.O(n3)
431、下面程序段的时间复杂度为（）。x=90；y=100；
while(y>0)if(x>100){x=x-10；y--；}elsex++；-->O（1）。
432、下面的操作不是栈基本运算的是（）。-->C.排序操作
433、下面的说法中,不正确的是（）。-->广义表是一种非线性结构
434、下面的应用中，不符合栈的后进先出特点的是（）。-->D.算数运算、逻辑运算和关系运算
435、下面关于串的叙述中，不正确的是（）。-->B.空串是由空格构成的串
436、下面关于队列的说法正确的是（）。-->队列是一种先进先出的线性表
437、下面关于哈希查找的说法,正确的是（）。-->不存在特别好与坏的哈希函数,要视情况而定
438、下面关于线性表的叙述中,错误的是（）。-->线性表采用顺序存储,进行插入和删除操作,不需要进行数据元素间的移动
439、下述几种排序方法中,（）是稳定的排序方法。-->归并排序
440、下图二叉树的后序遍历序列是（）。-->FDBGHECA
441、下图二叉树的中序遍历序列是（）。-->BFDAGEHC
442、**下图中，若从顶点 a 出发，按深度优先搜索法进行遍历，则可能得到的一种顶点序列为（ ）。**

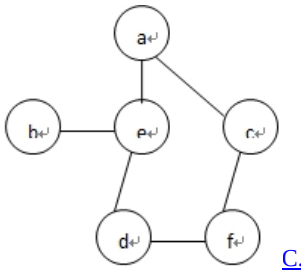


C.aedfcb
443、下图中的二叉树的先序遍历序列是（）。-->ABDFCEGH
444、下图中的哈夫曼树的带权路径长度是（）。-->36
445、线性表 L=(a1,a2,……an),下列说法正确的是（）。-->除第一个和最后一个元素外,其余每个元素都有一个且仅有一个直接前驱和直接后继
446、线性表 L 在（）情况下适用于使用链式结构实现。-->需不断对 L 进行删除、插入
447、线性表采用链接存储时,其地址（）。-->连续与否均可以
448、线性表的顺序存储结构是一种（）的存储结构。-->随机存取
449、线性表的顺序结构中,（）。-->逻辑上相邻的元素在物理位置上也相邻
450、线性表若采用链式存储结构时,要求内存中可用存储单元的地址（）。-->连续或不连续都可以的
451、线性表以（）方式存储,能进行折半查找。-->D.关键字有序的顺序
452、线性表元素的个数等于 0 时称为（）表-->空
453、线性表在存储后,如果相关操作是：要求 YI 知第 i 个结点的位置访问该结点的前驱结点,则采用（）存储方式是不可行的。-->单链表
454、线性表在存储后,如果相关操作中有要求：利用已知的指向某结点的指针或序号,访问该结点的前驱结点,则采用（）的存储方式是不可行的。-->单向链表
455、线性表只要以（）方式存储就能进行折半查找。-->关键字有序的顺序
456、线性表中（）称为线性表的长度。-->C.数据元素个数
457、线性表中（）称为线性表的长度。-->C.数据元素个数
458、线性结构、树形结构、图形结构都是按数据的（）来分类的。-->D.逻辑结构
459、线性结构的基本特点烛除第一个元素无直接（）,最后一个元素无直接后继之外,其他每个元素都有一个前驱和后继-->前驱
460、线性结构中数据元素的位置之间存在（）的关系。-->一对一
461、线性结构中数据元素之间的关系是（）-->A.一对一
462、线性结构中数据元素之间的关系是（）-->A.一对一
463、向顺序栈中压入新元素时,应当（）。-->A.先移动栈顶指针,再存入元素
464、向顺序栈中压入新元素时，应当（）。-->A.先移动栈顶指针，再存入元素
465、向一个有 127 个元素的顺序表中插入一个新元素，并保持原来的顺序不变，平均要移动（）个元素。-->A.63.5
466、校和队列的共同特点是（）。-->只容许在端点处插入和删除元素
467、序列按顺序依次进钱,按该校的可能输出序列依次入队列,该队列的不可能输出序列是（）0(进枝、出校可以交替进行)。-->4,8,12,16
468、循环队列存储在数组 A[0．．m]中,则入队时的操作为（）。-->rear=（rear+1）%（m+1）。
469、一般情况下,将递归算法转换成等价的非递归算法应该设置()。-->A.栈
470、一般情况下，将递归算法转换成等价的非递归算法应该设置（）。-->A.栈
471、一个不带头结点的单循环链表,尾指针为 rear,在链表中插入一个 s 所指向的新结点,并作为新的尾结点,可执行（）。-->s 一 next=rear 一 nextrear 一 next=s；rear=s；
472、一个存储结点存储一个（）。-->数据元素
473、一个单链表中,在 p 所指结点之后插入一个 s 所指的结点时,可执行：s->next=p->next；和（）。-->p-next=s；
474、一个递归算法必须包括（）。-->D.终止条件和迭代部分
475、一个队列的人队序列是 1,2,3,4。则队列的输出序列是（）。-->B.1,2,3,4
476、一个队列的人队序列是 2,4,6,8,按该队列的输出序列使各元素依次入钱,该枝的可能输出序列是（）。-->8,6,4,2
477、一个队列的入队顺序是 a,b,c,d，则离队的顺序是（）。-->B.a,b,c,d
478、一个队列的入队序列是 10,20,30,40。则队列的输出序列是()。-->B.10,20,30940
479、一个队列的入队序列是 1，2，3，4。则队列的输出序列是()。-->B.1，2，3，4
481、一个非空广义表的表头（）。-->D.可以是子表或原子
482、一个非空广义表的表头元素（）。-->D.可以是子表或原子
483、一个高度为 h 的满二叉树共有 n 个结点,其中有 m 个叶子结点,则有（）成立。-->n=2m-1
484、一个具有 1025 个结点的二叉树的高 h 为（）。-->11 至 1025 之间
485、一个具有 n 个顶点的无向完全图包含（）-->C.n(n-1)/2
486、一个具有 n 个顶点的有向完全图包含()条边。-->A.n(n-1)。
487、一个钱的进战序列是 a,b,C,d,e,则拢的不可能输出序列是（）(进拽出钱可以交替进行)。-->dceab
488、一个顺序表第一个元素的存储地址是 90,每个元素的长度为 2,则第 6 个元素的地址是（）。-->100
489、一个校的进技序列是 12345 则钱的不可能输出序列是()(进找出战可以交替进行)。-->43512
490、一个校的进校序列是 1,2,3,4,则校的不可能的出钱序列是（）(进出校操作可以交替进行)。-->1,4,2,3
491、一个校的进战序列是 a,b,C,d,则栈的不可能的出钱序列是()。-->adbc
492、一个栈的进栈序列是 10，20，30，40，50，则栈的不可能输出序列是（）（进栈出栈可以交替进行）。-->B.40,30,50,10,20

493、一个栈的进栈序列是 a,b,c,d,则栈的不可能的出栈序列是(）。-->adb
494、一个栈的进栈序列是 efgh,则栈的不可能的出栈序列是（）(进出栈操作可以交替进行)。-->ehfg
495、一个栈的入栈序列是 1,2,3,4,5,则栈的不可能的输出序列是(）。-->4,3,5,1,2
496、一棵采用链式存储的二叉树中,共有 n 个指针域被有效使用(即指针域为非空)。该二叉树有（）个指针域为空。-->n+2
497、一棵采用链式存储的二叉树中,共有 n 一 l 个指针域被有效使用(即指针域为非空)该二叉树中有（）个指针域为空。-->n+1
498、一棵采用链式存储的二叉树中有 n 个指针域为空,该二叉树共有（）个结点。-->n-1
499、一棵非空的二叉树的先序遍历序列与后序遍历序列正好相反,则该二叉树一定满足（）。-->只有一个叶子结点
500、一棵高度为 4 的二叉树，最多含有（）个结点。-->B.15
501、一棵哈夫曼树有 10 个非叶子结点(非终端结点),该树总共有（）个结点。-->21
502、一棵哈夫曼树有 n 个叶子结点(终端结点儿该树总共有因 l(）个结点。-->2n-l
503、一棵哈夫曼树总共有 23 个结点,该树共有（）个叶结点(终端结点)。-->12
504、一棵哈夫曼树总共有 25 个结点,该树共有（）个非叶结点(非终端结点)。-->12
505、一棵结点数 31<n<40 的完全二叉树,最后一层有 4 个结点,则该树有（）个叶结点。-->18
506、一棵具有 16 个结点的完全二叉树,共有（）层。(设根结点在第一层)。-->5
507、一棵具有 36 个结点的完全二叉树,最后一层有（）个结点。-->5
508、一棵具有 38 个结点的完全二叉树,最后一层有（）个结点。-->7
509、一棵深度为 5 的满二叉树,有（）个分支结点-->15
510、一棵完全二叉树共有 4 层,且第 4 层上有 2 个结点,该树共有（）个非叶子结点(根为第一层)。-->4
511、一棵完全二叉树共有 5 层,且第 5 层上有六个结点,该树共有（）个结点。-->21
512、一棵完全二叉树共有 6 层,且第 6 层上有 6 个结点,该树共有（）个结点。-->37
513、一棵完全二叉树上有 1001 个结点,其中叶子结点的个数是(）。-->501
514、一棵有 23 个结点,采用链式存储的二叉树中,共有（）个指针域为空。-->24
515、一棵有 n 个结点,采用链式存储的二叉树中,共有)个指针域被有效使用(即指针域为非空)。-->n-1
516、一棵有 n 个结点采用链式存储的二叉树中,共有（）个指针域为空。-->n+1
517、一维数组 A 采用顺序存储结构，每个元素占用 6 个字节，第 6 个元素的存储地址为 100，则该数组的首地址是（）。-->C.70
518、一种逻辑结构（）存储结构。-->可以有不同的
519、一种逻辑结构在存储时（）。-->可采用不同的存储结构
520、一组记录的关键字序列为(46,38,56,40,79,84),利用快速排序,以第一个关键字为分割元素,经过一次划分后结果为（）。-->40,38,46,56,79,84
521、一组记录的关键字是{19,14,23,1,68,20,84,27,55,11,10,79}，用链接地址法构造散列表，散列函数为 H（key）=keymod13,散列地址为 1 的链中有（）个记录。-->D.4
522、一组记录的关键字序列为(12,45,22,4,6,50),利用快速排序,以第一个关键字为分割元素,经过一次划分后结果为（）。-->6,4,12,22,45,50
523、一组记录的关键字序列为(25,48,16,35,79,82,23,40,36,72),其中,含有 5 个长度为 2 的有序表,按归并排序的方法对该序列进行一趟归并后的结果为（）。-->16，25，35，48，23，40，79，82，36，72
524、一组记录的关键字序列为(26．59．36．18．20．25),利用堆排序的方法建立的初始小根堆为（）。-->18,20,25,59,26,36
525、一组记录的关键字序列为(40,80,65,100,14,30,55,50),利用堆排序的方法建立的初始小根堆为（）。-->14，40，30，50，80，65，55，100
526、一组记录的关键字序列为(46,20,30,79,56,38,40,84,90,110),利用快速排序,以第一个关键字为分割元素,经过一次划分后结果为（）。-->40,20,30,38,46,56,79,84,90,110
527、一组记录的关键字序列为(46,79,56,38,40,84),利用快速排序,以第一个关键字为分割元素,经过一次划分后结果为（）。-->40,38,46,56,79,84
528、一组记录的关键字序列为(56,30,89,66,48,50,94,87,100),利用快速排序,以第一个关键字为分割元素,经过一次划分后结果为(）。-->50，30，48，56，66，89，94，87，100
529、一组记录的关键字序列为(75,63,95,80,53,45,38,2 的,利用堆排序(堆顶元素是最大元素)的方法建立的初始堆为（）。-->95,80,75,63,53,45,38,20
530、一组记录的关键字序列为（25，48，16，35，79，82，23，40，36，72），其中，含有 5 个长度为 2 的有序表，按归并排序的方法对该序列进行一趟归并后的结果为（）。-->B.16，25，35，48，23，40，79，82，36，72
531、一组记录的关键字序列为（26，59，36，18，20，25），利用堆排序的方法建立的初始小根堆为（）。-->B.18，20，25，59，26，36
532、一组记录的关键字序列为（46，20，30，79，56，38，40，84，90，110），利用快速排序，以第一个关键字为分割元素，经过一次划分后结果为（）。-->A.40，20，30，38，46，56，79，84，90，110
533、一组记录的关键字序列为（46，79，56，38，40，84），利用堆排序的方法建立的初始堆为（）。-->B.84，79，56，38，40，46
534、一组记录的关键字序列为（46，79，56，38，40，84），利用快速排序，以第一个关键字为分割元素，经过一次划分后结果为（）。-->C.40，38，46，56，79，84
535、一组记录的关键字序列为（60，47，80，57，39，41，46，30），利用归并排序的方法,第一趟归并后的结果为（）。-->D.47，60，57，80，39，41，30，46
536、一组记录的关键字序列为（60，47，80，57，39，41，46，30），利用归并排序的方法,对该序列进行（1,1）归并，即第一趟归并后的结果为（）。-->D.47，60，57，80，39，41，30，46
537、一组记录的关键字序列为（80，57，41，39，46，47），利用堆排序（堆顶元素是最小元素）的方法建立的初始堆为（）。-->C.39，46，41，57，80，47
538、依次将每两个相邻的有序表合并成一个有序表的排序方法称为（）。-->D.归并排序
539、依次将每两个相邻的有序表合并成一个有序表的排序方法称为（）。-->D.归并排序
540、已知 10 个数据元素为（54，28，16，34，73，62，95，60，26，43），对该数列从小到大排序，经过一趟冒泡排序后的序列为（）。-->B.28，16，34，54，62，73，60，26，43，95
541、已知某二叉树的后续遍历序列是 dabec，中序遍历是 debac，则它的先序遍历序列是（）。-->D.cedba
542、**已知如图所示的一个图,若从顶点 a 出发,按广度优先搜索法进行遍历,则可能得到的一种顶点序列为（）。**

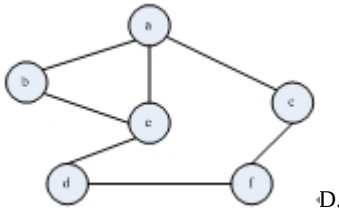


B.aecbdf
543、已知如图所示的一个图,若从顶点 B 出发,按广度优先法进行遍历,则可能得到的一种顶点序列为（）。-->BADECHFG
544、已知如图所示的一个图,若从顶点 V1 出发,按深度优先搜索法进行遍历,则可能得到的一种顶点序列为（）。-->V1V2V4V8V5V3V6V7
545、**已知如图所示的一个图，若从顶点 a 出发，按深度优先搜索法进行遍历，则可能得到的一种顶点序列为（）。**



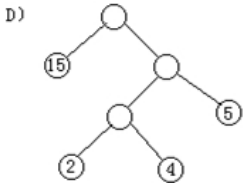
C.aedfcb
546、已知无向图 G 的顶点数为 n，边数为 e，其邻接表表示的空间复杂度为（）。-->C.O（n+e）。
547、已知一个图的边数为 m,则该图的所有顶点的度数之和为(）。-->2m
548、已知一个图的所有顶点的度数之和为 m,且 m 是以下 4 种情况之一,则 m 只可能是（）。-->8

549、已知一个图的所有顶点的度数之和为 m,则 m 一定不可能是（）。-->9
550、已知一个图的所有顶点的度数之和为 m,则该图的边数为（）。-->m/2
551、**已知一个图如下图所示，若从顶点 a 出发按广度优先搜索法进行遍历，则可能得到的一种顶点序列为（）。**



D.abecdf

552、已知一个有向图的邻接矩阵表示，要删除所有从第 i 个结点发出的边，应（）。-->B.将邻接矩阵的第 i 行元素全部置为 0
553、已知一个有序表为(12,18,24,35,47,50,62,83,90,115,134),当折半查找值为 90 的元素时,经过（）次比较后查找成功。-->2
554、已知一个有序表为,则顺序查找元素 55 需要比较（）次。-->5
555、已知一个有序表为{11,22,33,44,55,66,77,88,99},则顺序查找元素 55 需要比较（）次。-->C.5
556、已知一个有序表为{11,22,33,44,55,66,77,88,99}，则顺序查找元素 55 需要比较（）次。-->C.5
557、以下（）不是队列的基本运算。-->C.从队列中删除第 i 个元素
558、以下表中可以随机访问的是（）。-->顺序表
559、以下陈述错误的是（）。-->线性表的链式存储结构优于顺序存储结构
560、以下陈述中正确的是（）。-->A.串是一种特殊的线性表
561、以下程序段的结果是 C 的值为（）。-->7
562、以下数据结构中,（）是非线性结构。-->树
563、以下数据结构中（）是非线性结构？-->图
564、以下说法不正确的是（）。-->枝的删除操作在战底进行,插入操作在战顶进行
565、以下说法不正确的是（）。-->一种逻辑结构只能有唯一的存储结构
566、以下说法正确的是（）。-->拢的删除和插入操作都只能在校顶进行
567、以下说法正确的是（）。-->在顺序表中可以随机访问任一结点
568、以下说法正确的是（）。-->栈的特点是先进后出
569、以下说法正确的是（）。-->一些表面上很不相同的数据可以有相同的逻辑结构
570、以下说法中不正确的是（）。-->YI 知单向链表中任一结点的指针就能访问到链表中每个结点
571、以下四个串中最小的是（）。-->A.”ABADF”
572、引入二叉线索树的目的是（）。-->加快查找结点的前驱或后继的速度
573、用单链表表示的链栈的栈顶在链表的（）位置。-->A.链头
574、用链接方式存储的队列,在进行删除运算时（）。-->头、尾指针可能都要修改
575、用邻接表表示图进行广度优先遍历时,通常借助（）来实现算法。-->队列
576、用邻接表表示图进行深度优先遍历时,通常借助（）来实现算法。-->栈
577、用某种排序的方法对线性表（25，84，21，47，15，27，68，35，20）进行排序时，元素序列的变化情况如下：（1）25，84，21，47，15，27，68，35，20（2）20，15，21，25，47，27，68，35，84（3）15，20，21，25，35，27，47，68，84（4）15，20，21，25，27，35，47，68，84 其所采用的排序方法是（）。-->C.快速排序
578、**用权值分别为 15，2，4，5 的四个结点，构造出的哈夫曼树为（）。**



579、用折半查找法,对长度为 12 的有序的线性表进行查找,最坏情况下要进行（）次元素间的比较。-->4
580、由 1 个结点可以构造出多少种不同的二叉树？（）-->1
581、由 3 个结点可以构造出多少种不同的二叉树？（）-->5
582、由六个叶子结点 a、b、c、d、e、f 构造的哈夫曼树（）。-->B.不唯一
583、由权值为{3,8,6,2,5}的叶子结点生成一棵哈夫曼树,其带权路径长度为（）。-->53
584、有关线性表的正确说法是（）。-->D.除了一个和最后一个元素外，其余元素都有一个且仅有一个直接前驱和一个直接后继
585、有数据{53,30,37,12,45,24,96}，从空二叉树开始逐个插入数据来形成二叉排序树，若希望高度最小，应该选择的序列是（）。-->B.37,24,12,30,53,45,96
586、有序表为(1,2,4,6,10,18,20,32),用课本中折半查找算法查找值 18,经（）次比较后成功查到。-->2
587、有一个长度为 10 的有序表，按折半查找对该表进行查找，在等概率情况下查找成功的平均比较次数为（）。-->A.29/10
588、有一个长度为 12 的有序表,按折半查找对该表进行查找,在等概率情况下查找成功的平均比较次数为（）。-->A.37/12
589、有一个长度为 12 的有序表，按折半查找对该表进行查找，在等概率情况下查找成功的平均比较次数为（）。-->A.37/12
590、有一个长度为 12 的有序表，按折半查找对该表进行查找，在等概率情况下查找成功的平均比较次数为（）。-->Dec-37
591、有一个长度为 5 的线性表,按顺序查找某关键字,在等概率情况下查找成功的平均比较次数为（）-->3
592、有一个长度为 5 的线性表,按顺序查找某关键字,在等概率情况下查找成功的平均比较次数为（）。-->C.3
593、有一个长度为 7 的有序表,按折半查找对该表进行查找,在等概率情况下查找成功的平均比较次数为（）。-->17/7
594、有一个长度为 9 的有序表,按折半查找对该表进行查找,在等概率情况下查找成功的平均比较次数为（）。-->25/9
595、与数据元素本身的形式、内容、相对位置、个数无关的是数据的（）。-->逻辑结构
596、与顺序表相比,链表的优势是（）-->D.插入数据元素较快
597、**元素 a，b，c，d 按顺序依次进栈，则该栈的可能输出序列是（）（进栈出栈可以交替进行）**.C.a，c，b，d
598、元素 1,3,5,7 按顺序依次进钱,按该校的可能输出序列依次人队列,该队列的可能输出序列是（）．(进找出钱可以交替进行)。-->7,5,3．1
599、元素 111,113,115,117 按顺序依次进钱,则该栈的不可能输出序列是（）(进栈出栈可以交替进行)。-->117,115,111,113
600、元素 12,14,16,18 顺序依次进栈,则该栈的不可能输出序列是（）。(进栈出栈可以交替进行)。-->18,16,12,14
601、元素 15,9,11,13 按顺序依次进找,则该校的不可能输出序列是（）(进找出钱可以交替进行)。-->13,11,15,9
602、元素 2,4,6,8 按顺序依次进栈,按该栈的可能输出序列依次人队列,该队列的可能输出序列是（）(进栈出栈可以交替进行)。-->8,6,4,2
603、元素 2,4,6,8 按顺序依次进栈,则该栈的不可能输出序列是（）(进栈出栈可以交替进行)。-->8,6,2,4
604、元素 20,14,160,180 按顺序依次进栈,则该栈的不可能输出序列是（）。(进栈出栈可以交替进行)。-->180,160,20,14
605、元素 212,214,216,218 按顺序依次进栈,则该栈的不可能输出序列是（）(进栈出栈可以交替进行)。-->D.218,216,212,214
606、元素 4,6,8,10 按顺序依次进栈,按该栈的可能输出序列依次入队列,该队列的可能输出序列是（）(进栈出栈可以交替进行)。-->10,8,6,4
607、元素 41,43,45,47 按顺序依次进栈,则该栈的可能输出序列是（）(进栈出栈可以交替进行)。-->43,45,41,47
608、元素 4，6，8，10 按顺序依次进栈，按该栈的可能输出序列依次入队列，该队列的可能输出序列是（）（进栈出栈可以交替进行）。-->D.10，8，6，4
609、元素 a,b,c,d 按顺序依次进擒,则该拢的不可能输出序列是（）(进找出钱可以交替进行)。-->d,c,a,b
610、在 C 语言中,存储字符串“ABCD“需要占用（）字节。-->5
611、在 C 语言中,分别存储“S”和“s”,各需要占用（）字节。-->两个和一个
612、在 C 语言中,顺序存储长度为 3 的字符串,需要占用（）个字节。-->4
613、在 n 个结点的顺序表中,算法的时间复杂度是 O(1)的操作是（）。-->访问第 i 个结点(1≤i≤n)和求第 i 个结点的直接前驱(2≤i≤n)。
614、在待排序元素基本有序的情况下，效率最高的排序方法是（）。-->A.插入排序
615、在单链表中,要将 s 所指结点插入到 p 所指结点之后,其语句应为（）。-->s-next=p-next；p-next=s；
616、在堆排序和快速排序中,若原始记录接近正序和反序,则选用（）排序-->堆
617、在对一组记录(50,40,95,20,15,70,60,45,80)进行直接插入排序时,当把第 7 个记录 60 插入到有序表时,为寻找插入位置需要比较（）次-->3

618、在对一组元素(64,48,106,33,25,82,70,55,93)进行直接插人排序时,当进行到要把第 7 个元素 70 插入到 yi 经排好序的子表时,为找到插人位置,需进行（）次元素 n 的比较(指由小到大排序)。-->3
619、在二叉树的第 4 层最多含有（）个结点。-->A.8
620、在二叉树的第 4 层最多含有（）个结点。-->8
621、在二叉树的链式存储结构中,通常每个结点中设置三个域,它们是值域、（）、右指针。-->左指针
622、在非空双向循环链表的*p 结点之前插入*q 结点的操作是()。-->D.q-next=p；q-prior=p-prior；p-prior-next=q；p-prior=q；
623、在个不带头结点的单循环链表中,p、q 分别指向表中第一个结点和尾结点,现要删除第 4 个结点,且 p、q 仍然分别指向新表中第一个结点和尾结点。可用的语句是 p=p 一>next；和（）。-->q 一 next=P
624、在解决计算机主机与打印机之间速度不匹配问题时通常设置一个打印数据缓冲区，主机将要输出的数据依次写入缓冲区中，而打印机则从缓冲区中取出数据打印，该缓冲区应该是一个（）结构。-->B.队列
625、在排序过程中,可以通过某一趟排序的相关操作所提供的信息,判断序列是否经排好序,从而可以提前结束排序过程的排序算法是（）。-->冒泡
626、在排序过程中,可以有效地减少一趟排序过程中元素间的比较次数的算法是（）。-->折半插入
627、在平衡二叉树中插入一个结点后造成了不平衡,设最低的不平衡结点为 A,并已知 A 的左孩子的平衡因子为 0 右孩子的平衡因子为 1,则应作（）型调整以使其平衡。-->RL
628、在实际应用中，要输入多个字符串，且长度无法预定。则应该采用（）存储比较合适。-->A.链式
629、在数据结构和算法中,与所使用的计算机有关的是（）。-->数据的存储结构
630、在数据结构中,从逻辑上可以把数据结构分为（）。-->D.线性结构和非线性结构
631、在数据结构中，从逻辑上可以把数据结构分为（）。-->D.线性结构和非线性结构
632、在双向链表存储结构中,删除 p 所指的结点时须修改指针()。-->p-next-prior=p-prior；p-prior-next=p-next；
633、在双向循环链表中,在 p 所指的结点之后插入指针 f 所指的新结点,其操作步骤是()。-->f-prior=p；f-next=p-next；p-next-prior=f；p-next=f；
634、在双向循环链表中,在 p 指针所指的结点后插入 q 所指向的新结点,其修改指针的操作是（）。-->q-prior=p；q-next=p-next；p-next-prior=q；p-next=q；
635、在所有的排序方法中,关键字比较的次数与记录初始排列秩序无关的是（）。-->直接选择排序
636、在无向图中定义顶点 vi 与 vj 之间的路径为从 vi 到 vj 的一个（）。-->A.顶点序列
637、在下列存储形式中,（）不是树的存储形式？-->顺序存储表示法
638、在下列几种排序方法中，平均情况下占用内存量最大的是()方法。-->D.归并排序
639、在下列排序方法中，关键字比较的次数与记录初始排列秩序无关的是（）。-->C.选择排序
640、在线性表(a0,a1,a2,a3,…．an)中,a1 的后继是（）。-->a2
641、在线性表(a1,a2,a3,…．an)中,a1 的前驱是（）。-->没有前驱
642、在线性表的顺序结构中，以下说法正确的是（）。-->D.逻辑上相邻的元素在物理位置上也相邻
643、**在循环双链表的 p 所指结点之后插入 s 所指结点的操作是()。**
B.p->right=s；p->right->left=s；s->left=p；s->right=p->right；
644、在一非空二叉树的中序遍历序列中，根结点的右边（）。-->A.只有右子树上的所有结点
645、在一非空二叉树的中序遍历序列中，根结点的右边（）。-->只有右子树上的所有结点
646、在一个不带头结点的单循环链表中,p、q 分别指向表中第一个结点和尾结点,现要删除第一个结点,可用的语句是（）。-->p=p 一 next；q-next=p；
647、在一个不带头结点的链队中，假设 f 和 r 分别为队头和队尾指针，则从该对列中删除一个结点并把结点的值保存在变量 x 中的运算为（）。-->C.x=f-data；f=f-next；
648、在一个查找表中,能够唯一地确定一个记录的关键字称为（）-->主关键字
649、在一个长度为 n 的顺序表中,在第 i 个元素(1<=i<=n+1)之前插入一个新元素时需向后移动（）个元素。-->n-i+1
650、在一个长度为 n 的顺序表中为了删除第 5 个元素，由第 6 个元素开始从后到前依次移动了 15 个元素。则原顺序表的长度为()。-->20
651、在一个长度为 n 的顺序表中为了删除第 5 个元素，由第 6 个元素开始从后到前依次移动了 15 个元素。则原顺序表的长度为()。-->B.20
652、在一个带头结点的单向链表中，若要在指针 q 所指结点后插入 p 指针所指结点，则执行（）。-->A.p-next=q-next；q-next=p；
653、在一个单链表 Head 中，若要向表头插入一个由指针 p 指向的结点，则执行（）。-->A.Head=p；p-next=Head；
654、在一个单链表中,p、q 分别指向表中两个相邻的结点,且 q 所指结点是 p 所指结点的直接后继,现要删除 q 所指结点,可用语句()。-->q-next,q,p-next,q-next,NULL
655、在一个单链表中,已知 q 所指结点是 p 所指结点的直接前驱,若在 q 和 p 之间插入 s 所指结点,则执行（）操作。-->q-next=s；s-next=p；
656、在一个单链表中 p 所指结点之后插入一个 s 所指的结点时,可执行（）。-->D.s-next=p-next,p-next=s
657、在一个单链表中 p 所指结点之后插入一个 s 所指的结点时,可执行（）。-->D.s-next=p-next；p-next=s；
658、在一个单链表中 p 指向结点 a,q 指向结点 a 的直接后继结点 b,要删除结点 b,可执行（）。-->A.p-next=q-next
659、在一个单向链表中,p 和 q 分别是指向结点类型的指针,要删除 p 所指结点的直接后继结点,可执行()。-->q=p-next；p-next=q-next
660、在一个单向链表中所指结点之后插入一个所指的结点时,可执行（）。-->s-next=p-next；p-next=s
661、在一个链队中,假设 f 和 r 分别为队头和队尾指针,p 指向一个新结点,要为结点 p 所指结点赋值 x,并入队的运算为 p->data=x；P 一>next=NULL；-->r-next=P；r=P；
662、在一个链队中,假设 f 和 r 分别为队头和队尾指针,p 指向一个已生成的结点,现要为该结点的数据域赋值 e,并使结点入队的运算为 p->data=e；p->next 一 NULL；和（）。-->B.r-next=p；r=p
663、在一个链队中,假设 f 和 r 分别为队头和队尾指针,yi 生成一个结点 p,要为结点 p 赋值 x,并入队的运算为（）。-->p 一 data=x；p 一 next=NULL；r 一 next=p；r=p；
664、在一个链队中,假设 f 和 r 分别为队头和队尾指针,则插入 s 所指结点的运算为（）。-->r-next=s；r=s；
665、在一个链队中,假设 f 和 r 分别为队头和队尾指针,则删除一个结点的运算为（）。-->f=f-next；
666、在一个链队中，假设 f 和 r 分别为队头和队尾指针，则插入 s 所指结点的运算为（）。-->B.r-next=s；r=s；
667、在一个链队中，假设 f 和 r 分别为队头和队尾指针，则删除一个结点的运算为（）。-->C.f=f-next；
668、在一个钱顶指针为 top 的链找中进行出钱操作,用变量 x 保存钱顶元素的值,则执行（）。-->x=top 一 data；top=top 一 next；
669、**在一个顺序表中，为了删除第 5 个元素，由第 6 个元素开始依次往前移动了 15 个元素。则原顺序表的长度为（）**C.20
670、在一个顺序队列中，队首指针指向队首元素的()位置。-->A.前一个
671、在一个头指针为 head 的带头结点的单向循环链表中,p 指向尾结点,要使该链表成为不带头结点的单向链表,可执行（）。-->D.head-head-next；p-next=NULL
672、在一个头指针为 head 的带头结点的单向循环链表中,p 指向尾结点,要使该链表成为不带头结点的单向链表,可执行（）。-->head=head-next；p-next=NULL
673、在一个头指针为 head 的单向链表中,p 指向尾结点,要使该链表成为单向循环链表可执行（）。-->p-next=head；
674、在一个图 G 中,所有顶点的度数之和等于所有边数之和的()倍。-->C.2
675、在一个图 G 中，所有顶点的度数之和等于所有边数之和的()倍。-->C.2
676、在一个尾指针为 rear 的不带头结点的单循环链表中,插入一个所指的结点,并作为第一个结点,可执行()。-->s→next=rear→next；rear→next=s；
677、在一个无向图 G 中,所有边数之和等于的所有顶点的度数之和（）倍。-->A.1/2
678、在一个无向图中,所有顶点的度数之和等于所有边数的()倍。-->2
679、在一个无向图中，若两顶点之间的路径长度为 k，则该路径上的顶点数为（）。-->B.k+1
680、在一个有 115 个元素的顺序表中插入一个新元素并保持原来顺序不变,平均要移动的元素个数为（）。-->57．5
681、在一个有向图中,所有顶点的入度之和等于所有顶点的出度之和的（）倍。-->1
682、在一个栈顶指针为 top 的链栈中,将一个 p 指针所指的结点入栈,应执行（）。-->C.p-next=top；top=p
683、在一个栈顶指针为 top 的链栈中，将一个 p 指针所指的结点入栈，应执行（）。-->C.p-next=top；top=p；
684、在一个栈顶指针为 top 的链栈中删除一个结点时,用 x 保存被删结点的值,则执（）-->D.x=top-data；top=top-next；

685、在一个栈顶指针为 top 的链栈中删除一个结点时，用 x 保存被删结点的值，则执行（）。-->D.x=top-data；top=top-next；
686、在一棵度具有 5 层的满二叉树中结点总数为（）。-->A.31
687、在一棵度为 3 的树中,度为 3 的结点个数为 2,度为 2 的结点个数为 1,则度为 0 的结点个数为（）。-->6
688、在一棵二叉树的二叉链表中，空指针域数等于非空指针域数加（）。-->C.2
689、在一棵二叉树上,第 5 层的结点数最多为（）。-->C.16
690、**在一棵二叉树上，第 4 层的结点数最多为（）**A.8
691、在一棵二叉树上，第 5 层的结点数最多为（）。-->C.16
692、在一棵二叉树中(其根结点编号为 1),若编号为 8 的结点存在右孩子,则该右孩子的顺序编号为（）。-->D.17
693、在一棵二叉树中,编号为 17 的结点的双亲结点的的顺序编号为（）。-->8
694、在一棵二叉树中,编号为 19 的结点的双亲结点的顺序编号为（）。-->A.9
695、在一棵二叉树中,若编号为 16 的结点是其双亲结点的左孩子,则他的双亲结点的顺序编号为（）。-->8
696、在一棵二叉树中,若编号为 8 的结点存在右孩子,则右孩子的顺序编号为（）。-->D.17
697、在一棵二叉树中,若编号为 9 的结点存在右孩子,则右孩子的顺序编号为（）。-->19
698、在一棵二叉树中,若编号为 i 的结点存在双亲结点,则双亲结点的顺序编号为)。-->i/2 向下取整
699、在一棵二叉树中,若编号为 i 的结点存在右孩子,则（）孩子的顺序编号为 2i+1-->右
700、在一棵二叉树中,若编号为 i 的结点存在右孩子,则右孩子的顺序编号为（）。-->2i+1
701、在一棵二叉树中,若编号为 i 的结点存在左孩子,1 结点的左孩子的顺序编号为（）。-->2*i
702、在一棵二叉树中,若编号为 i 的结点是其双亲结点的左孩子,则双亲结点的顺序编号为（）。-->i/2
703、**在一棵二叉树中（其根结点编号为 1），若编号为 5 的结点存在左孩子，则该左孩子的顺序编号为（）**
B.10
704、在一棵二叉树中，若编号为 8 的结点存在右孩子，则右孩子的顺序编号为（）。-->D.17
705、在一棵树中,（）没有前驱结点。-->树根结点
706、在一棵树中，度为 0 的结点称作（）。-->A.叶子结点
707、在有向图的邻接表中,每个顶点邻接表链接着该顶点所有（）邻接点。-->B.出边
708、在有向图的邻接表中,每个顶点邻接表链接着该顶点所有（）邻接点。-->B.出边
709、在有向图的邻接表中，每个顶点邻接表链接着该顶点所有()邻接点。-->B.出边
710、在有向图的逆邻接表中，每个顶点邻接表链接着该顶点所有（）邻接点。-->A.入边
711、在有序表{1,3,8,13,33,42,46,63,76,78,86,97,100}中,用折半查找值 86 时,经（）次比较后查找成功。-->4
712、在有序表{10,14．34．43．47．64．75．80．90}中,用折半查找法查找值 80 时,经（）次比较后查找成功。-->3
713、在有序表{1，3，8，13，33，42，46，63，76，78，86，97，100}中，用折半查找值 86 时，经（）次比较后查找成功。-->B.4
714、在有序表{2,4,7,14,34,43,47,64,75,80,90,97,120}中,用折半查找法查找值 80 时,经（）次比较后查找成功。-->2
715、在有序表{21,23,2,33,4,45,4,7,7,78,8,9,106}中,用折半查找值 43 时,经（）次比较后查找成功。-->3
716、在最坏情况下，折半查找与二叉排序树查找性能比较，（）。-->B.折半查找性能好
717、战和队列的共同特点是（）。-->都是操作受限的线性结构
718、栈的操作特性决定了它是一种（）的线性表。-->B.先进后出
719、栈的插入和删除操作在（）进行。-->A.栈顶
720、栈的基本运算包括（）-->D.取栈顶元素
721、栈顶指针通常命名为（）-->B.top
722、栈和队列的共同特点是（）。-->C.只容许在端点处插入和删除元素
723、栈和队列的主要区别在于（）。-->插入、删除运算的限定不一样
724、栈在（）中有所应用。-->前三个选项都有
725、折半查找有序表(4,6,10,12,20,30,50,70,88,100)。若查找表中元素 58,则它将依次与表中（）比较大小,查找结果是失败。
-->20,70,30,50
726、折半搜索与二叉排序树的时间性能（）。-->有时不相同
727、这四个值作为叶子结点的权,生成一棵哈夫曼树,该树中所有叶子结点中的最长带权路径长度为（）。-->18
728、**执行下面程序段时，执行 S 语句的次数为（）。**

```
for (int i=1;i<=n;i++)
  for (int j=1;i<=i;j++)
    S;
```

答：B.n2/2
729、直接插入排序在最好情况下的时间复杂度为 O（）-->n
730、子串“acd”在主串“abdcacdefac”中的位置是（）。-->7
731、字符串“DABcdabcd321ABC“的子串是（）。-->“cd32“
732、字符串 a1=“AEIJING“,a2=“AEI“,a3=“AEFANG“,a4=“AEFI“中最大的是（）。-->a1
733、字符串（）是“abcd321ABCD”的子串。-->“21AB”
734、字符串的基本函数不包括（）。-->D.串的分割
735、最小生成树指的是（）。-->连通网中所有生成树中权值之和为最小的生成树

简答(6)--

1、简述广义表和线性表的区别和联系。...
2、简述数据的逻辑结构和存储结构的区别与联系...
3、解释顺序存储结构和链式存储结构的特点,并...
4、设栈 S 和队列 Q 的初始状态为空,元素 e1,e2,e3,...
5、有 5 个元素,其入栈次序为：A、B、C、D、E,在各...
6、栈、队列和线性表的区别是什么？...

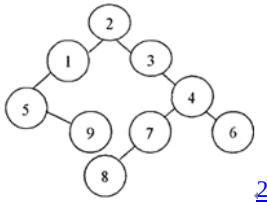
1、简述广义表和线性表的区别和联系。
答：广义表是线性表的的推广，它也是 n（n>0）个元素 a1，a2，…，ai，…，an 的有限序列，其中 ai 或者是原子或者是一个广义表。所以，广义表是一种递归数据结构，而线性表没有这种特性，线性表可以看成广义表的特殊情况，当 ai 都是原子时，广义表退化成线性表。
2、简述数据的逻辑结构和存储结构的区别与联系,它们如何影响算法的设计与实现？
答：若用结点表示某个数据元素，则结点与结点之间的逻辑关系就称为数据的逻辑结构。数据在计算机中的存储表示称为数据的存储结构。可见，数据的逻辑结构是反映数据之间的固有关系，而数据的存储结构是数据在计算机中的存储表示。尽管因采用的存储结构不同，逻辑上相邻的结点，其物理地址未必相同，但可通过结点的内部信息，找到其相邻的结点，从而保留了逻辑结构的特点。采用的存储结构不同，对数据的操作在灵活性，算法复杂度等方面差别较大。
3、解释顺序存储结构和链式存储结构的特点,并比较顺序存储结构和链式存储结构的优缺点。
答：顺序结构存储时，相邻数据元素的存放地址也相邻，即逻辑结构和存储结构是统一的，要求内存中存储单元的地址必须是连续的。
优点：一般情况下，存储密度大，存储空间利用率高。
缺点：（1）在做插入和删除操作时，需移动大量元素；（2）由于难以估计，必须预先分配较大的空间，往往使存储空间不能得到充分利用；（3）表的容量难以扩充。
链式结构存储时，相邻数据元素可随意存放，所占空间分为两部分，一部分存放结点值，另一部分存放表示结点间关系的指针。
优点：插入和删除元素时很方便，使用灵活。
缺点：存储密度小，存储空间利用率低。
4、设栈 S 和队列 Q 的初始状态为空,元素 e1,e2,e3,e4,e5 和 e6 依次通过 S,一个元素出栈后即进队列 Q,若 6 个元素出队的序列是 e2,e4,e3,e6,e5,e1,则栈 S 的容量至少应该是多少？
答：出队序列是 e2,e4,e3,e6,e5,e1 的过程：
（1）e1 入栈（栈底到栈顶元素是 e1）
（2）e2 入栈（栈底到栈顶元素是 e1，e2）
（3）e2 出栈（栈底到栈顶元素是 e1）
（4）e3 入栈（栈底到栈顶元素是 e1，e3）
（5）e4 入栈（栈底到栈顶元素是 e1，e3，e4）
（6）e4 出栈（栈底到栈顶元素是 e1，e3）
（7）e3 出栈（栈底到栈顶元素是 e1）
（8）e5 入栈（栈底到栈顶元素是 e1，e5）
（9）e6 入栈（栈底到栈顶元素是 e1，e5，e6）
（10）e6 出栈（栈底到栈顶元素是 e1，e5）
（11）e5 出栈（栈底到栈顶元素是 e1）
（12）e1 出栈（栈底到栈顶元素是空）
栈中最多时有 3 个元素，所以栈 S 的容量至少是 3。
5、有 5 个元素,其入栈次序为：A、B、C、D、E,在各种可能的出栈次序中,以元素 C、D 最先的次序有哪几个？
答：从题中可知，要使 C 第一个且 D 第二个出栈，应是 A 入栈，B 入栈，C 入栈，C 出栈，D 入栈。之后可以有以下几种情况：
（1）B 出栈，A 出栈，E 入栈，E 出栈，输出序列为：CDBAE。
（2）B 出栈，E 入栈，E 出栈，A 出栈，输出序列为 CDBEA。
（3）E 入栈，E 出栈，B 出栈，A 出栈，输出序列为 CDEBA
所以可能的次序有：CDBAE，CDBEA，CDEBA
6、栈、队列和线性表的区别是什么？
答：栈是一种先进后出的线性表，栈的插入和删除操作都只能在

栈顶进行，而一般的线性表可以在线性表的任何位置进行插入和删除操作。
队列是一种先进先出的线性表，队列的插入只能在队尾进行，队列的删除只能在队头进行，而一般的线性表可以在线性表的任何位置进行插入和删除操作。

判断(438)--

1、“顺序查找法”是指在顺序表上进行查找的方法。-->错
2、18 个元素进行冒泡法排序，通常需要进行 17 趟冒泡,其中第 10 趟冒泡共需要进行 8 次元素间的比较。-->对
3、AOV 网是一个带权的有向图。-->错
4、AOV 网拓扑排序的结果是惟一的。-->错
5、AOV 网拓扑排序的结果是惟一的。-->错
6、n 个顶点的无向连通图至少有 n-1 条边,n 个顶点的有向强连通图至少有 n 条边。-->对
7、n 个元素进行冒泡法排序,通常第 j 趟冒泡要进行 n-j 次元素间的比较。-->对
8、n 个元素进行冒泡法排序，通常第 j 趟冒泡要进行 n-j 次元素间的比较。-->对
9、n 个元素进行冒泡法排序，最多需要进行 n-1 趟冒泡。-->对
10、按照二叉树的递归定义,对二叉树遍历的常用算法有深度优先遍历和深度优先遍两种方法。-->错
11、按照一定规则，在二叉排序树上插入、删除结点，仍能保持二叉排序树的性质。-->对
12、边数很多的稠密图,适宜用邻接矩阵表示。-->对
13、不管堆栈采用何种存储结构,只要堆栈不空,可以任意删除一个元素。-->错
14、采用分块查找时，数据的组织方式是把数据分成若干块，块内数据不必有序，但块间必需有序，每块内最大（或最小）的数据组成索引表。-->对
15、采用分块查找时，数据的组织方式是把数据分成若干块，块内数据不必有序，但块间必需有序，每块内最大（或最小）的数据组成索引表。-->对
16、采用链式存储的线性表称作链表。-->对
17、采用邻接表存储的图的广度优先遍历方法类似于二叉树的按层次遍历方法。-->对
18、采用邻接表存储的图的广度优先遍历算法类似于二叉树的按层次遍历。-->对
19、采用顺序查找法对长度为 n（n 为偶数）的线性表进行查找，采用从前向后的方向查找。在等概率条件下成功查找到前 n/2 个元素的平均查找长度为（n+2）/4。-->对
20、采用顺序查找方法查找长度为 n 的线性表时，每个元素的平均查找长度为 n/2。-->错
21、长度为 0 的线性表称为空表。-->对
22、长度为 0 的线性表称为空表。-->对
23、长度为 0 字符串称为空白串。-->错
24、程序是用计算机语言表述的算法。-->对
25、串的两种最基本的存储方式是顺序和链接。-->对
26、串的两种最基本的存储方式是顺序和链接。-->对
27、串函数 strcmp(“ABCd”,“ABCD”)的值为一 1。-->错
28、串函数 stremp(“ABCd”,“ABCD”)的值为-1。-->错
29、串是一种内容受限的线性表,串的数据元素是字符。-->对
30、串是一种特殊的线性表，其特殊性表现在组成串的数据元素都是字符。-->对
31、串是一种特殊的线性表，其特殊性表现在组成串的数据元素都是字符。-->对
32、串中的元素只可能是字母。-->错
33、串中任意个字符组成的子序列称为该串的子串。-->错
34、从源点到终点的最短路径是唯一的。-->错
35、存储图的邻接矩阵中，邻接矩阵的大小不但与图的顶点个数有关，而且与图的边数也有关。-->错
36、存储图的邻接矩阵中，邻接矩阵的大小不但与图的顶点个数有关，而且与图的边数也有关。-->错
37、存储无向图的邻接矩阵是对称的,故只存储邻接矩阵的下(或上)三角部分即可。-->对
38、待排序的序列为 8,3,4,1,2,5,9，采用直接选择排序算法，当进行了两趟选择后，结果序列为 1,2,8,3,4,5,9。-->错
39、单链表从任何一个结点出发,都能访问到所有结点-->对
40、单链表可以实现随机存取。-->错
41、当从一个最小堆中删除一个元素时,需要把堆尾元素填补到堆顶位置,然后再按条件把它逐层向下调整,直到调整到合适位置为止。-->对
42、当字符集中的各字符使用频率不均匀时,等长编码是最优的前缀码。-->错
43、递归的算法简单、易懂、容易编写，而且执行效率也高。-->错
44、递归调用算法与相同功能的非递归算法相比，主要问题在于重复计算太多，而且调用本身需要分配额外的空间和传递数据和控制，所以时间与空间开销通常都比较大。-->对
45、递归定义的数据结构通常不需要用递归的算法来实现对它的操作。-->错
46、递归定义的数据结构通常用递归算法来实现对它的操作。-->对
47、递归定义的数据结构通常用递归算法来实现对它的操作。-->对
48、递归算法可读性差，但是效率高-->错
49、递归算法可读性差，但是效率高-->错
50、递归算法执行时，每次递归可将原问题的规模缩小。-->对
51、递归算法执行时，每次递归可将原问题的规模缩小。-->对
52、堆栈、队列和数组的逻辑结构都是线性表结构。-->对
53、堆栈在数据中的存储原则是先进先出。-->错
54、队列的特性是先进后出。-->错
55、队列的特性是先进后出。-->错
56、队列的特性是先进后出。-->错
57、队列和栈都是运算受限的线性表。-->对
58、队列允许删除的一端称为队尾，允许插入的一端称为队头。-->错
59、队列允许删除的一端称为队尾，允许插入的一端称为队头。-->错
60、对 16 个元素的序列用冒泡排法进行排序，最多需要进行 15 趟冒泡。-->对
61、对 16 个元素的序列用冒泡排序法进行排序,最多需要进行 15 越冒泡。-->对
65、对 n 个元素进行日泡法排序,最多需要进行 n-1 越目泡。-->对
66、对二叉树中的结点进行按层次顺序(每一层自左至右)的访问操作称为二叉树的层次遍历,遍历所得的结点序列称为二叉树的层次序列。-->对
67、对连通图进行深度优先遍历可以访问到该图中的所有顶点。-->对
71、对链表进行插入和删除操作时,不必移动结点。-->对
72、对任意一个图从它的某个顶点出发进行一次深度优先或广度优先搜索遍历可访问到该图的每个顶点。-->错
73、对稀疏矩阵进行压缩存储,矩阵中每个非零元素对应的三元组包括该元素的行号、列号和元素值三项信息。-->对
74、对稀疏矩阵进行压缩存储,矩阵中每个非零元素对应的三元组包括该元素的行下标、列下标、和非零元素值三项信息。-->对
75、对稀疏矩阵进行压缩存储,可采用三元组表,一个 6 行 7 列的稀疏矩阵 A 相应的三元组表共有 8 个元素,则矩阵 A 共有 34 个零元素。-->对
76、对稀疏矩阵进行压缩存储，矩阵中每个非零元素对应的三元组包括该元素的行号、列号和元素值三项信息。-->对
77、对稀疏矩阵进行压缩存储，矩阵中每个非零元素对应的三元组包括该元素的行号、列号和元素值三项信息。-->对
78、对有向图 G,如果从任一顶点出发进行一次深度优先或广度优先搜索就能访问每个顶点,则该图一定是完全图。-->错
79、对于一个具有 n 个结点的单链表，在*p 结点后插入一个新结点的时间复杂度是 O（n）。-->错
80、对于一棵具有 n 个结点,其高度为 h 的二叉树,进行任一种次序遍历的时间复杂度为 O 否。-->对
81、对于一棵具有 n 个结点的二叉树,其相应的链式存储结构中共有 n-1 个指针域空。-->错
82、对于一棵深度为 4 的满三叉树，其结点数为 40。-->对
83、对于一棵深度为 h，度为 3 的树最多有（3h-1）/2 个结点。-->错
84、多维数组是向量的推广。-->错
85、二叉排序树是用来进行排序的-->错
86、二叉排序树在呈单支二叉树时,查找效率最低。-->对
87、二叉排序树在呈单支二叉树时，查找效率最低。-->对
88、二叉排序树中某一结点的左儿子一定小于树中任一个结点的右儿子。-->错
89、二叉树的遍历就是按照一定次序访问树中所有结点，并且每个结点的值仅被访问一次的过程。-->对
90、二叉树的根结点值大于其左子树结点的值，小于右子树结点的值，则它是一棵二叉排序树。-->错
91、二叉树的前序遍历序列中，任意一个结点均处在其子女结点的前面。-->对
92、二叉树的子树有左右之分，其子树的次序不能颠倒。-->对
93、二叉树为二叉排序树的充分必要条件是，任一个分支结点的值都大于其左孩子的值，小于右孩子的值。-->错
94、二叉树只能采用二叉链表来存储。-->错
95、二叉树只能采用二叉链表来存储。-->错

96、二叉树中每个结点的度最大为 2,因此二叉树是一种特殊的树。-->错
97、二叉树中任一结点的值均大于其左孩子的值，小于其右孩子的值，则它是一棵二叉排序树。-->错
98、二路归并时,被归并的两个子序列中的关键字个数一定要相等。-->错
99、二维数组是其数组元素为线性表的线性表。-->对
100、访问单链表中的结点，必须沿着指针链依次进行。-->对
101、非空二叉排序树的任意一棵子树也是二叉排序树。-->对
102、非空线性表中任意一个数据元素都有且仅有一个直接前驱元素。-->错
103、分块查找分为两个步骤：第一步是要对索引表进行查找；第二步是在块中查找。这两步查找都可以采用折半查找或者顺序查找方法。-->错
104、分块查找是一种介于顺序查找和折半查找之间的查找方法。-->A
105、各种链表只需定义有两个域的结点。-->错
106、各种链表只需定义有两个域的结点。-->错
107、根据搜索方法的不同,图的遍历有．先序；中序；后序三种方法。-->错
108、根据图的存储结构进行某种次序的遍历，得到的顶点序列是唯一的。-->错
109、根据无序序列构造二叉排序树的过程，也是对无序序列排序的过程。-->对
110、关键字是记录某个数据项的值，用它可以识别、确定一个记录。-->对
111、广义表 A《a,b,c），(d,e,f））的表尾为《d,e,f））。-->对
112、广义表的表头总是一个广义表。-->错
113、广义表的深度是指广义表中元素的个数。-->错
114、哈夫曼树是带权值的树,且权值较大的结点离树较近。-->对
115、哈夫曼树是访问叶子结点的外部路径长最长的二叉树。-->错
116、哈夫曼树叶结点数比非叶结点数多 1。-->对
117、哈夫曼树一定是完全二叉树或满二叉树。-->错
118、哈夫曼树只存在着双支结点，不存在单支结点。-->对
119、哈夫曼树只存在着双支结点，不存在单支结点。-->对
120、哈希函数的构造方法中，除留余数法是最好的。-->错
121、衡量排序算法的两个主要性能指标是执行排序算法所需要的时间和执行排序算法所需要的附加空间。-->对
122、假设在有序线性表 A[120]上进行折半查找，则比较五次查找成功的结点数为 5。-->对
123、简单插入排序是不稳定的排序算法。-->错
124、健壮的算法不会因非法的输入数据而出现莫名其妙的状态。-->对
125、将 10 个元素散列到 10000 个单元的哈希表中，仍然可能会产生冲突。-->对
126、将新元素插入到队列任意位置是队列的基本运算之一。-->错
127、将新元素插入到队列任意位置是队列的基本运算之一。-->错
128、结构中的数据元素存在多对多的关系称为图形结构。-->错
129、进栈运算是栈的基本运算之一。-->对
130、具有 100 个结点的完全二叉树有 50 个叶子。-->对
131、具有 12 个结点的完全二叉树的深度为 4。-->对
132、具有 256 个结点的完全二叉树的深度为 9。-->对
133、具有 n 个顶点的无向图采用邻接矩阵表示，图中的边数等于邻接矩阵中非零元素之和的一半。-->对
134、具有 n 个结点的二叉树，采用二叉链表存储，共有 n+1 个空链域。-->对
135、具有 n 个结点的二叉树，采用二叉链表存储，共有 n-1 个空链域。-->错
136、具有三个结点的二叉树有五种。-->对
137、可以通过硬件解决算法的效率问题。-->错
138、空串的长度是 0；空格串的长度是空格字符的个数。-->对
139、空串的长度是 1。-->错
140、空串的长度是 1。-->错
141、空串是任意串的子串-->对
142、空串与空格串是相同的。-->错
143、快速排序是排序算法中最快的一种。-->错
144、快速排序在任何情况下均可得到最块的排序效果。-->错
145、类 C 语言是对 C 语言的简化和扩展，强化了 C 语言的表达能力。-->对
146、理想情况下,哈希表查找等概率查找成功的时间复杂度是 O(1)。-->对
147、理想情况下，哈希表查找等概率查找成功的时间复杂度是 O(1)。-->对
148、链式栈与顺序栈相比,一个明显的优点是通常不会出现栈满的情况。-->对
149、链栈通常不会出现栈满的状态-->对
150、两个串相等的充分必要条件是每一个对应位置的字符相同。-->对
151、两个字符串比较时，较长的串比较短的串大-->B
152、两个字符串比较时，较长的串比较短的串大-->错
153、邻接表法只用于有向图的存储,邻接矩阵对于有向图和无向图的存储都适用。 -->错
154、邻接表只能用于存储有向图,而邻接矩阵则可存储有向图和无向图。-->错
155、逻辑结构与数据元素本身的内容和类型无关。-->对
156、满二叉树中存在度为 1 的结点。-->错
157、满二叉树中没有度为 1 的结点。-->对
158、冒泡排序是一种比较简单的插入排序方法。-->错
159、冒泡排序是一种比较简单的插入排序方法。-->错
160、冒泡排序是一种比较简单的交换排序方法。-->对
161、每种数据结构都应具备三种基本运算：插入、删除和搜索。-->错
162、任何无向网络拓扑排序的结果是唯一的。-->错
163、任何有向网络(AOV-网络)拓扑排序的结果是唯一的。-->错
164、任一个有向图的拓扑序列只有一个。-->错
165、如果不知道单向链表的头指针，就无法访问该链表的任意结点。-->对
166、如果结点 A 有 3 个兄弟,而且 B 是 A 的双亲,则 B 的度是 4。-->对
167、如果结点 A 有 3 个兄弟 3 个孩子，而且 B 是 A 的双亲，则 A 的度是 3。-->错
168、如果结点 A 有 3 个兄弟，而且 B 是 A 的双亲，则 B 的度是 4。-->对
169、如果结点 A 有 3 个兄弟，而且 B 是 A 的双亲，则 B 的度是 4。-->对
170、如果一个叶子结点是某二叉树中序遍历序列的最后一个结点，那么它也是该二叉树的先序遍历序列的最后一个结点。-->对
171、**如图所示的二叉树,其先序遍历序列为**



215934786
172、若采用三元组压缩技术存储稀疏矩阵,只要把每个元素的行下标和列下标互换,就完成了对该矩阵的转置运算。-->对
173、若连通图上各边权值均不相同，则该图的最小生成树是惟一的。-->对
174、若频繁地对线性表进行插入和删除操作,该线性表采用顺序存储结构更合适。-->错
175、若让元素 1,2,3 依次进栈，则出栈次序 1,3,2 是不可能出现的情况。-->错
176、若让元素 1，2，3 依次进栈，则出栈次序 1，3，2 是不可能出现的情况。-->错
177、若让元素 1，2，3 依次进栈，则出栈次序 1，3，2 是不可能出现的情况。-->错
178、若让元素 a,b,c 依次进栈，则出栈次序 c,a,b 是不可能出现的情况。-->对
179、若树的度为 2 时,该树为二叉树。-->错
180、若树的度为 2 时，该树为二叉树。-->错
181、若树中各结点的子树是按照一定的次序从左向右安排的，则称之为有序树。-->对
182、若线性表采用顺序存储结构,每个数据元素占用 4 个存储单元,第 12 个数据元素的存储地址为 144,则第 1 个数据元素的存储地址是 101。-->错
183、若一个广义表的表头为空表,则此广义表亦为空表。-->错
184、若一个有向图的邻接矩阵中对角线以下元素均为零，则该图的拓扑有序序列必定存。-->对
185、森林是 m（m≥0）棵互不相交的树的集合。-->对
186、删除二叉排序树中一个结点,再重新插入上去,一定能得到原来的二叉排序树。-->错
187、设 p,q 是指针,若 p=q,则*p=*q。-->错
188、设广义表 L=(,),则其长度是 0。-->错
189、设广义表 L=(,)，则其长度是 0。-->错
190、设广义表 L=(，)，则其表头是()。-->错
191、设广义表 L=(，)，则其表尾是。-->错
192、设广义表 L=)，)，则其长度是 0。-->错

193、设某棵二叉树的中序遍历序列为 ABCD，前序遍历序列为 CABD，则后序遍历该二叉树得到序列为 BCDA。-->错
194、设一棵完全二叉树,其最高层上最右边的叶结点的编号为奇数,该叶结点的双亲结点的编号为 10. 该完全二叉树一共有 21 个结点。-->对
195、设有 n 阶对称矩阵-->错
196、设有 n 阶对称矩阵 A，用一维数组 s 压缩存储 A 的下三角元素，s 的下标从零开始，元素 s[26]相应于 A 中的元素为 a7,6。-->对
197、设有一个不带头结点的单向循环链表，结点的指针域为 next，指针 p 指向尾结点，现要使 p 指向第一个结点，可用语句 p=p->next;-->对
198、设有一个长度为 40 的顺序表,要删除第 8 个元素需移动元素的个数为 33。-->错
199、设有一个长度为 40 的顺序表，要删除第 8 个元素需移动元素的个数为 3。-->错
200、设有一个单向链表，结点的指针域为 next，头指针为 head，p 指向尾结点，为了使该单向链表改为单向循环链表，可用语句 p->next=head。-->对
201、设有一个单向链表，结点的指针域为 next，头指针为 head，p 指向尾结点，为了使该单向链表改为单向循环链表，可用语句 p->next=head。-->对
202、设有一个单向循环链表,结点的指针域为 next,头指针为 head,指针 p 指向表中某结点,若逻辑表达式 p-&gt；next==head；的结果为真,则 p 所指结点为尾结点。-->对
203、设有一个单向循环链表,头指针为 head,链表中结点的指针域为 next,p 指向尾结点的直接前驱结点,若要删除尾结点,得到一个新的单向循环链表,可执行操作 p->next=head；-->对
204、设有一个单向循环链表，结点的指针域为 next，头指针为 head，指针 p 指向表中某结点，若逻辑表达式 p->next==head；的结果为真，则 p 所指结点为尾结点。-->对
205、设有一个单向循环链表，结点的指针域为 next，头指针为 head，指针 p 指向表中某结点，若逻辑表达式 p->next==head；的结果为真，则 p 所指结点为尾结点。-->对
206、设有一个单向循环链表，头指针为 head，链表中结点的指针域为 next，p 指向尾结点的直接前驱结点，若要删除尾结点，得到一个新的单向循环链表，可执行操作 p->next=head；-->对
207、设有一个非空的链栈,栈顶指针为 hs,要进行出栈操作,用 x 保存出栈结点的值,栈结点的指针域为 next,数据域为 data,则可执行 hs=hs-&gt；next；x=hs-&gt；data；-->错
208、设有一个链栈,栈顶指针为 hs,现有一个 s 所指向的结点要入栈,则可执行操作。hs=s；s-&gt；next=hs；-->错
209、深度为 5 的二叉树最多有 31 个结点。-->对
210、深度为 5 的二叉树最多有 31 个结点。-->对
211、深度为 5 的二叉树最多有 3 层。-->错
212、深度为 h 的非空二叉树的第 i 层最多有 2h-1 个结点。-->错
213、深度为 k 的完全二叉树至少有 2k-1 个结点。-->错
214、深度为 k 的完全二叉树至少有 2k-1 个结点。-->错
215、使用三元组表存储稀疏矩阵的元素,有时并不能节省存储空间。-->对
216、使用三元组表示稀疏矩阵中的非零元素能节省存储空间。-->对
217、使用折半查找算法的前提条件是，查找表中记录相应的关键字值必须按升序或降序排列。-->对
218、树的所有结点有且只有一个前驱结点。-->错
219、树的先根遍历序列等同于该树对应的二叉树中序遍历序列。-->错
220、树是一种线性结构。-->错
221、树是一种重要的非线性数据结构。-->对
222、树中全部结点的度均大于 0。-->错
223、树最适合表示元素之间具有层次关系的数据。-->对
224、树最适合表示元素之间具有层次关系的数据。-->对
225、数据的存储结构不仅有顺序存储结构和链式存储结构,还有索引结构与散列结构。-->错
226、数据的存储结构和逻辑结构无关。-->错
227、数据的逻辑结构是与存储该结构的计算机相关的。-->错
228、数据的逻辑结构是指各数据元素之间的逻辑关系，是用户根据应用需要建立的。-->对
229、数据的逻辑结构是指各数据元素之间的逻辑关系，是用户根据应用需要建立的。-->对
230、数据的逻辑结构与数据元素本身的内容和形式无关。-->对
231、数据的逻辑结构与数据元素本身的内容和形式无关。-->对
232、数据的逻辑结构在计算机中的表示称为逻辑结构。-->错
233、数据结构的抽象操作的定义与具体实现有关。-->错
234、数据结构概念包括数据之间的逻辑结构,数据在计算机中的存储方式和数据的运算三个方面。-->对
235、数据结构中,元素之间存在多对多的关系称为树状结构。-->错
236、数据结构中,元素之间存在多对多的关系称为树状结构。-->错
237、数据结构中,元素之间存在多对多的关系称为图状结构。-->对
238、数据结构中，数据可以由一个或多个数据项组成。-->错
239、数据结构中，元素之间存在多对多的关系称为图状结构。-->对
240、数据项是对数据操作的基本单位。-->错
241、数据项是数据处理的最小单位。-->对
242、数据项是数据的最小单位。-->对
243、数据元素可以由一个或多个数据项组成。-->对
244、数据元素可以有一个或多个数据项组成。-->对
245、数据元素是数据处理的最小单位。-->错
246、数据元素是数据的最小单位。-->错
247、数据元素之间的抽象关系称为物理结构。-->错
248、数组是一种复杂的数据结构,数组元素之间的关系既不是线性的,也不是树形的。-->错
249、数组通常具有的操作是顺序存取。-->错
250、双链表中至多只有一个结点的后继指针为空。-->对
251、双向循环链表构建的队列，可以只设立队首指针，也可以只设队尾指针-->对
252、顺序表的插入和删除操作不需要付出很大的时间代价,因为每次操作平均只有近一半的元素需要移动。-->错
253、顺序查找是一种最简单的查找方法。-->对
254、顺序查找是一种最简单的查找方法。-->对
255、顺序存储的线性表可以随机访问,链式存储的线性表只能顺序访问。-->对
256、顺序队列的入队算法是先检查队列是否为满，若不满则将新元素值赋给队头指针所指向的数据单元，再将队头指针加 1。-->错
257、顺序文件是指文件中的物理记录按其在文件中的逻辑记录顺序依次存入存储介质而建立的。-->错
258、顺序栈永远不会出现栈满的状态-->错
259、算法的时间复杂度比空间复杂度更重要。-->错
260、算法的优劣与算法描述语言无关,但与所用计算机有关。-->错
261、算法和程序都应具有下面一些特征：有输入,有输出,确定性,有穷性,有效性。-->错
262、算法和程序都应具有下面一些特征：有输入，有输出，确定性，有穷性，有效性。-->错
263、算法和程序都应具有下面一些特征：有输入，有输出，确定性，有穷性，有效性。-->错
264、算法和程序原则上没有区别，在讨论数据结构时二者是通用的。-->错
265、算法就是程序。-->错
266、算法可以用不同的语言描述,如果用 C 语言或 PASCAL 语言等高级语言来描述,则算法实际上就是程序了。-->错
267、缩短关键路径上活动的工期一定能够缩短整个工程的工期。-->对
268、题干"顺序查找法"是指在顺序表上进行查找的方法。-->错
269、通常可以把某城市中各公交站点间的线路图抽象成树型结构。-->错
270、通常可以把一本含有不同章节的书的目录结构抽象成线性结构。-->错
271、通常可以把一本含有不同章节的书的目录结构抽象成线性结构。-->错
272、同一逻辑结构采用不同的存储方法,可以得到不同的存储结构。-->对
273、图 G 的某一最小生成树的代价一定小于其他生成树的代价。-->错
274、图的广度优先搜索算法通常采用递归算法求解。-->对
275、图的广度优先搜索序列是惟一的。-->错
276、图的连通分量是无向图的极大连通子图。-->对
277、图的强连通分量是无向图的极大连通子图。-->错
278、图的深度优先搜索是一种典型的回溯搜索的例子,可以通过递归算法求解。-->对
279、图的深度优先搜索序列和广度优先搜索序列不是惟一的。-->对
280、图的深度优先搜索序列和广度优先搜索序列不是惟一的。-->对
281、图的生成树是惟一的。-->错
282、图的生成树是惟一的。-->错
283、图的最小生成树的形状可能不唯一。-->对
284、图的最小生成树只有一棵。-->错

285、外部排序是指在排序的整个过程中,全部数据在计算机的外存储器中完成的排序。-->错
286、完全二叉树就是满二叉树。-->错
287、完全二叉树中没有度为 1 的结点。-->错
288、完全二叉树中没有度为 1 的结点。-->错
289、完全二叉树中每个结点或者没有孩子或者有 2 个孩子。-->错
290、往栈中插入元素的操作方式是：先写入元素，后移动栈顶指针。-->错
291、往栈中插入元素的操作方式是：先写入元素，后移动栈顶指针。-->错
292、无向图的邻接矩阵一定是对称的。-->对
293、无向图的邻接矩阵一定是对称的。-->对
294、稀疏矩阵压缩存储后,必会失去随机存取功能。-->对
295、线性表的链式存储表示优于顺序存储表示。-->错
296、线性表的逻辑顺序和存储顺序总是一致的。-->错
297、线性表的顺序存储和链式存储都必须占用内存中的连续存储单元。-->错
298、线性表的顺序存储结构是通过数据元素的存储地址直接反映数据元素的逻辑关系。-->对
299、线性表的顺序存储结构优于链式存储结构。-->错
300、线性表可以看成是广义表的特例,如果广义表中的每个元素都是单元素,则广义表便成为线性表。-->对
301、线性表是一个有限序列，不可以为空。-->错
302、线性表用关键字的顺序方式存储,可以用二分法查找。-->错
303、线性表用关键字的顺序方式存储，可以用二分法查找。-->错
304、线性表用关键字的顺序方式存储，可以用二分法排序。-->错
305、线性表用顺序方式存储可以随机访问。-->对
306、线性表用顺序方式存储可以随机访问。-->对
307、线性表中的数据元素必须具有相同的特性,即属于同一个数据对象,这种线性表称为同质的线性表。-->对
308、线性的数据结构可以顺序存储,也可以链接存储。非线性的数据结构只能链接存储。-->错
309、线性结构的基本特征是：每个元素有且仅有一个直接前驱和一个直接后继。-->错
310、向二叉排序树插入一个新结点时,新结点一定成为二叉排序树的一个叶子结点。-->对
311、向一个长度为 n 的顺序表中的第 i 个元素（1≤i≤n）之前插入一个元素时，需向后移动 n-i 个元素。-->错
312、向一个栈顶指针为 h 的链栈(结点的指针域为 next)中插入一个 s 所指结点时,先执行 s->next=h,再执行 h=s 操作。-->对
313、向一个栈顶指针为 h 的链栈(结点的指针域为 next)中插入一个 s 所指结点时，先执行 s->next=h，再执行 h=s 操作。-->对
314、向一个栈顶指针为 h 的链栈（结点的指针域为 next）中插入一个 s 所指结点时，先执行 s->next=h，再执行 h=s 操作。-->对
315、需要压缩存储的矩阵可分为特殊矩阵矩阵和稀疏矩阵矩阵两种。-->对
316、序列 15,13,16,14,19,17，采用冒泡排序算法（升序），经一趟冒泡后，结果序列是 13,15,14,16,17,19。-->对
317、序列 15,13,16,14,19,17，采用冒泡排序算法（升序），经一趟冒泡后，结果序列是 13,15,14,16,17,19。-->对
318、序列 3,1,7,18,6,9,13,12 经一趟归并排序的结果为 1,3,7,18,6,9,13,12。-->错
319、选择排序过程中元素之间的比较次数与原始序列的状态无关。-->对
320、循环队列的队头指针为 f,队尾指针为 r,当 r==f 时表明队列已满。-->错
321、循环队列的引入,目的是为了克服假上溢。-->对
322、循环队列的最大存储空间为 MaxSize,队头指针为 f,队尾指针为 r,当(r+1)%MaxSize=f 时表明队列已满。-->对
323、循环队列的最大存储空间为 MaxSize=6．采用少用一个元素空间以有效的判断栈空或栈满,若队头指针 front=4．当队尾指针 rear=3 时队满。-->对
324、循环队列队头指针在队尾指针后一个位置，队列是“满”状态。-->对
325、循环队列队头指针在队尾指针前一个位置，队列是“满”状态。-->对
326、循环队列是将队列想象成一个首尾相接的圆环。-->对
327、循环链表判断表尾结点用的条件是该结点的后继指针是看它是否为空指针。-->错
328、要访问单链表中的第 i 个结点,必须从表头开始依次访问过该结点之前的所有结点后才能够实现,即只能够采用顺序存取,而不能够随机存取任一个结点。-->对
329、要将指针 p 移到它所指的结点的下一个结点是执行语句 p←p->next。-->错
330、要在一个带头结点的单向循环链表中删除头结点,得到一个新的不带头结点的单向循环链表,若结点的指针域为 next,头指针为 head,尾指针为 p,则可执行 head=head->next；p->next=head；-->对
331、要在一个带头结点的单向循环链表中删除头结点，得到一个新的不带头结点的单向循环链表，若结点的指针域为 next，头指针为 head，尾指针为 p，则可执行 head=head->next；p->next=head；-->对
332、要在一个带头结点的单向循环链表中删除头结点，得到一个新的不带头结点的单向循环链表，若结点的指针域为 next，头指针为 head，尾指针为 p，则可执行 head=head->next；p->next=head；-->对
333、要在一个单向链表中 p 所指向的结点之后插入一个 s 所指向的新结点，若链表中结点的指针域为 next，可执行 p->next=s；s->next=p->next；的操作。-->错
334、要在一个单向链表中删除 p 所指向的结点,已知 q 指向 p 所指结点的直接前驱结点,若链表中结点的指针域为 next,则可执行 q->next=p->next。-->对
335、要在一个单向链表中删除 p 所指向的结点，已知 q 指向 p 所指结点的直接前驱结点，若链表中结点的指针域为 next，则可执行 q->next=p->next。-->对
336、要在一个单向链表中删除 p 所指向的结点，已知 q 指向 p 所指结点的直接前驱结点，若链表中结点的指针域为 next，则可执行 q->next=p->next。-->对
337、一个递归算法不必包括递归终止条件。-->错
338、一个广义表((a),((b),c),(((d))))的表尾是((b),c),(((d)))。-->错
339、一个广义表((a),((b),c),(((d))))的长度为 3,深度为 4。-->对
340、一个广义表((A),((B),c),(((D))))的长度为 3,深度为 4。-->对
341、一个广义表((a),((b),c),(((d))))的长度为 3，深度为 4。-->对
342、一个广义表((a),((b),c)，(((d))))的长度为 3,深度为 4。-->对
343、一个广义表的表头总是一个广义表。-->错
344、一个广义表的表头总是一个广义表。-->错
345、一个广义表的表尾总是一个表。-->对
346、一个好的哈希函数，应该使哈希地址均匀地分布在整个哈希表的地址区间中，完全避免冲突的发生。-->错
347、一个空格的串的长度是 0。-->错
348、一个空格的串的长度是 0。-->错
349、一个无向连通图的生成树是含有该连通图的全部顶点的极小连通子图。-->对
350、一个新结点插入链表中只需要修改一个指针域即可，而不需要移动数据元素。-->错
351、一个有向图的邻接表和逆邻接表中的节点个数一定相等。-->对
352、一个栈的输人序列是 1,2,3,4,5,则栈的输出序列有可能式 4,3,5,1,2。-->错
353、一棵二叉树的叶结点(终端结点)数为 5. 单分支结点数为 2. 该树共有 11 个结点。-->对
354、一棵二叉树有 6 个叶结点,则该树总共有 11 个结点。-->错
355、一棵哈夫曼树有 m 个叶子结点,则其结点总数为 2m-1。-->对
356、一棵完全二叉树深度为 5，最少有 16 个结点。-->对
357、一棵有 14 个结点的完全二叉树,则它的最高层上有 7 个结点。-->对
358、一棵有 7 个叶结点的二叉树,其 1 度结点数的个数为 2．则该树共有 15 个结点。-->对
359、一棵有 8 个权重值构造的哈夫曼数,共有 17 个结点。-->错
360、已知一棵二叉树的前序序列和中序序列可以唯一地构造出该二叉树．-->对
361、已知一棵树的先序序列和后序序列，一定能构造出该树。-->错
362、已知一棵树的先序序列和后序序列，一定能构造出该树。-->错
363、用非递归方法实现递归算法时一定要使用递归工作栈。-->错
364、用邻接矩阵存储图的时候,占用空间大小不但与图的结点个数有关还与图的边数有关。-->错
365、用邻接矩阵存储图的时候，占用空间大小不但与图的结点个数有关还与图的边数有关。-->错
366、用邻接矩阵存储图的时候，占用空间大小不但与图的结点个数有关还与图的边数有关。-->错
367、用数组实现顺序栈，栈底可以是数组空间的任何一端-->对
368、用相邻矩阵表示图所用的存储空间大小与图的边数成正比。-->错
369、用一组地址连续的存储单元存放的元素一定构成线性表。-->对
370、用字符数组存储长度为 n 的字符串，数组长度至少为 n+1。-->对

371、用字符数组存储长度为 n 的字符串，数组长度至少为 n+l。-->对
372、由树转化为二叉树,其根结点的右子树总是空的。-->对
373、有 n 个结点的无向图中，若边数大于 n-1，则该图是连通的。-->错
374、有向图的邻接矩阵一定是非对称的。-->错
375、有向图是一种非线性结构。-->对
376、有向图用邻接矩阵表示后，顶点 i 的出度等于第 i 行中非 0 且非无穷的元素个数。-->对
377、有一个链栈,栈顶指针为 h,现有一个 p 所指向的结点要入栈,则可执行操作 p-> next=h; 和 h=p; -->对
378、在长度为 n 的顺序表 L 中查找指定元素值的元素，其时间复杂度为 O（n）。-->对
379、在单链表中,要取得某个元素,只要知道该元素所在结点的地址即可,因此单链表是随机存取结构。-->错
380、在单链表中，要删除某一指定的结点，必须找到该结点的直接前驱结点。-->对
381、在队列的顺序存储结构中,当插入一个新的队列元素时,尾指针后移,当删除一个元素队列时,头指针后移。-->对
382、在队列的顺序存储结构中，当插入一个新的队列元素时，尾指针后移，当删除一个元素队列时，头指针后移。-->对
383、在队列的顺序存储结构中，当插入一个新的队列元素时，尾指针后移，当删除一个元素队列时，头指针后移。-->对
384、在队列中,允许插入的一段称为对头。-->错
385、在对 10 个记录的序列（14,30,10,7,22,13,66,85,47,58）进行直接插入排序时，当把第 6 个记录 13 插入到有序表时，为寻找插入位置，需比较 3 次。-->错
386、在二叉树的链接存储中，每个结点设置三个域：值域、左指针域和右指针域。-->对
387、在二叉树中插入结点则该二叉树便不再是二叉树。 -->错
388、在归并排序中，在第 3 趟归并中，是把长度为 4 的有序表归并为长度为 8 的有序表。-->对
389、在任意一棵二叉树中,叶子结点的个数等于度为 2 结点的个数加 1。-->对
390、在顺序查找、折半查找、哈希表查找 3 种方法中,平均查找长度与结点个数 n 无关的查找方法是折半查找。-->错
391、在顺序查找、折半查找、哈希表查找 3 种方法中，平均查找长度与结点个数 n 无关的查找方法是折半查找。-->错
392、在线性表的顺序存储结构中,逻辑上相邻的两个元素在物理位置上不一定相邻。-->错
393、在线性表的顺序存储中，元素之间的逻辑关系是通过物理存储位置决定的；在线性表的链式存储中，元素之间的逻辑关系是通过链域的指针值决定的。-->对
394、在循环队列中,front 指向队列中第一个元素的前一位置,rear 指向实际的队尾元素,队列为满的条件是 front=rear。-->错
395、在循环队列中,front 指向队头元素的前一个位置,rear 指向队尾元素的位置,则队满的条件是 front=rear。-->错
396、在一个不带头结点的非空链队中,f 和 r 分别为队头和队尾指针,队结点的数据域为 data,指针域为 next,若要进行出队操作,并用变量 x 存放出队元素的数据值,则相关操作为 x=f-> data; f=f-> next; -->对
397、在一个查找表中，能够唯一地确定一个记录的关键字称为主关键字。-->对
398、在一个单链表中的 p 所指结点之后插入*s 结点时，应执行 p->next=s 和 s->next=p->next 的操作。-->错
399、在一个具有 n 个顶点和 e 条边的无向图的邻接表中，边结点的个数为 e。-->错
400、在一个链队中,f 和 r 分别为队头和队尾指针,队结点的指针域为 next,s 指向一个要入队的结点,则入队操作为 r=s; r-> next=s; -->错
401、在一个链队中,f 和 r 分别为队头和队尾指针,队结点的指针域为 next,则插入所指结点的操作为 r-> next=s; r=s; -->对
402、在一个链式队列中，若队头指针与队尾指针的值相同，则表示该队列至多有 1 个结点。-->对
403、在一个顺序存储的循环队列中,队头指针指向队头元素的后一个位置。-->错
404、在一个无向图中,所有顶点的度数之和等于所有边数的 2 倍。-->对
405、在用单链表表示的链式队列 Q 中，队头指针为 Q->front，队尾指针为 Q->rear，则队空条件为 Q->front==Q->rear。-->错
406、在用循环单链表表示的链式队列中，可以不设队头指针，仅在链尾设置队尾指针。-->对
407、在有向图中每个顶点的度等于各顶点的入度与出度之和。-->对
408、在有序表 A[1…18]中，采用二分查找算法查找元素值等于 A[17]的元素，所比较过的元素的下标依次是 9、14、16、17。-->对
409、在有序顺序存储的线性表中查找一个元素,用折半查找速度一定比顺序查找快。-->错
410、在有序顺序存储的线性表中查找一个元素，用折半查找速度一定比顺序查找快。-->错
411、在栈满的情况下不能做进栈操作,否则将产生"上溢"。-->对
412、在只有度为 0 和度为 k 的结点的 k 叉树中,设度为 0 的结点有 n0 个,度为 k 的结点有 nk 个,则有 n0=nk+1。-->错
413、栈的插入删除在栈底进行。-->错
414、栈的特点是先进后出,队列的特点是先进先出。-->对
415、栈和队列的存储方式,既可以顺序存储也可以链式存储。-->对
416、栈和队列都是操作受限制的线性表。-->对
417、栈和队列都是顺序存取的线性表，但它们对存取位置的限制不同。-->对
418、栈和队列都是特殊的线性表,但它们对存取位置的限制不同。-->对
419、栈和队列都是特殊的线性表，但它们对存取位置的限制不同。-->对
420、栈和队列是一种操作受限的线性表。-->对
421、栈可以作为实现程序设计语言过程调用时的一种数据结构。-->对
422、栈是限定在表的两端进行插入和删除操作的线性表，又称为先进先出表。-->错
423、栈是限定在表的一端进行插入和删除操作的线性表,又称为先进后出表。-->对
424、栈是限定在表的一端进行插入和删除操作的线性表，又称为先进后出表。-->对
425、栈是限定在表的一端进行插入和删除操作的线性表，又称为先进后出表。-->对
426、栈允许进行插入和删除的一端称为栈顶-->对
427、折半查找的前提条件是，查找表中记录相应的关键字值必须有序或者部分有序。-->错
428、折半查找的前提条件是，查找表中记录相应的关键字值必须有序或者部分有序。-->错
429、折半查找方法适用于按值有序的线性链表的查找。-->错
430、折半查找方法运用在升序序列比降序序列效率更高，所以降序序列最好先转换为升序序列。-->错
431、折半查找只适用与有序表,包括有序的顺序表和有序的链表。-->错
432、直接选择排序是一种不稳定的排序方法。-->错
433、只有用面向对象的计算机语言才能描述数据结构算法。-->错
434、中序遍历一棵完全二叉树可得到一个有序序列。-->错
435、中序遍历一棵完全二叉树树可得到一个有序序列。-->错
436、字符串 a1=〝heijing〞,a2=〝hen〞,a3=〝heifang〞,a4="heni〞最小的是 a2。-->错
437、字符串属于线性的数据结构-->对
438、最小生成树是指边数最少的生成树。-->错

填空(402)--

1、18 个元素进行冒泡法排序,通常需要进行 17 趟冒泡,其中第 10 趟冒泡共需要进行（）次元素间的比较。-->8
2、20 个元素进行冒泡法排序,通常需要进行 19 趟冒泡,其中第 10 趟冒泡共需要进行（）次元素间的比较。-->10
3、c 言中,字符串"E"存储时占（）个字节。-->2
4、n 个顶点的连通图用邻接矩阵表示时,该矩阵至少有（）范围非 0 元素。-->n-1
5、n 个顶点的连通图至少有（）条边。-->n－1
6、n 个元素进行管泡法排序,通常需要进行（）趟冒泡。-->n-1
7、n 个元素进行冒泡法排序,第 j 趟冒泡要进行（）次元素间的比较。-->n-j
8、n 个元素进行冒泡法排序,通常需要进行（）趟冒泡,第 j 趟冒泡要进行（）次元素间的比较。-->n-1、n-j
9、n 个元素进行冒泡法排序,通常需要进行（）趟冒泡。-->n-1
10、（）遍历二叉排序树可得到一个有序序列。-->中序
11、（）查找是一种最简单的查找方法。-->顺序
12、（）串其长度等于零。-->空
13、（）的引入,目的是为了克服假溢出时大量移动数据元素。-->循环队列
14、（）的最大优点是从表中任意结点出发都可访问到表中每一个元素或从表中任意结点出发都可遍历整个链表。-->循环链表
15、（）二叉排序树可得到一个有序序列。-->中序
16、（）结构中,数据元素的位置之间存在多对多的关系。-->图状
17、（）结构中,数据元素间存在一对多的关系。-->树形
18、（）链表适合从指点结点开始,寻找直接前趋的运算。-->双向
19、（）排序不需要进行记录关键字间的比较。-->基数

20、（）是由一个或多个空格字符组成的串,其长度等于其包含的空格个数。-->空格串
21、按某关键字对记录序列排序,若（）在排序前和排序后仍保持它们的前后关系,则排序算法是稳定的,否则是不稳定的。11371t 寻分评卷入 i-->关键字梧等的记录
22、按某关键字对记录序列排序,若关键字（）的记录在排序前和排序后仍保 1 它们的前后关系,则排序算法是稳定的,否则是不稳定的。-->相等
23、按照二叉树的递归定义,对二叉树遍历的常用算法有（）、（）、（）三种。-->先序、中序、后序
24、把数据存储到计算机中,并具体体现数据之间的逻辑结构称为（）结构。-->物理（存储）。
25、榜的操作特点是后进（）。-->先出
26、本书中介绍的树形结构和（）属非线性结构。-->图状结构
27、边很多的图称为（）。-->稠密图
28、边很少的图称为（）-->稀疏图
29、不存在拓扑序列的（）是图中存在回路。-->有向图
30、长度为 255 的表,采用分块查找法,每块的最佳长度是（）。-->15
31、程序包括两个内容：数据结构和（）。-->算法
32、程序段 Char*s=“aBcD”；n=0；while(*s!=’/0’)(if(*s＞＝’a’&&*s＜＝’z’)n++；s++｝执行后 n=（）。-->2
33、出度是以该顶点为起点的（）数目。-->出边
34、除了第 1 个和最后一个结点外,其余结点有且只有一个前驱结点和后继结点的数据结构为（）,每个结点可有任意多个前驱和后继结点数的结构为（）-->线性结构,非线性结构
35、串的两种最基本的存储方式分别是（）和（）。-->顺序存储、链式存储
36、串函数 StrCat(a,b)的功能是进行串（）。-->连接
37、串其长度等于零。-->空
38、串是一种特殊的线性表,其特殊性表现在（）。-->其数据元素都是字符
39、串是一种特殊的线性表,其特殊性表现在组成串的数据元素都是（）。-->字符
40、从长度为 n 的采用顺序存储结构的线性表中删除第 i(1£i£n+1)个元素,需向前移动（）个元素。-->n-i
41、从根结点到该结点所经分支上的所有结点称为该结点的（）。-->祖先
42、从一个钱顶指针为 top 的链战中取钱顶元素,用 d 保存楼顶元素的值,可执行（）。(结点的数据域为 data)。-->d=top->data;
43、从一个战顶指针为 top 的链找中删除一个结点时,用 d 保存被删结点的值,可执行（）。(结点的指针域为 next,数据域为 data)。-->d=top->data；top=top->next;
44、从一个栈顶指针为 h 的链栈中删除一个结点时,用 x 保存被删结点的值,可执行（）-->x=h-
45、从一个栈顶指针为 top 的链栈中删除一个结点时,用 d 保存被删结点的值,可执行 d=top->data；和（）。(结点的指针域为 next,数据域为 data)。-->top=top->next;
46、从一个栈删除元素时,需要前移一位（）。-->栈顶指针
47、单链表中设置（）的作用是简化操作,减少边界条件的判断。-->头结点
48、单向循环链表是单向链表的一种扩充,当单向链表带有头结点时,把单向链表中尾结点的指针域由空指针改为（）；当单向链表不带头结点时,则把单向链表中尾结点的指针域由空指针改为指向（）。-->头结点的指针、指向第一个结点的指针
49、当从一个小根堆中删除一个元素时,需要把（）元素填补到（）位置,然后再按条件把它逐层（）调整。-->堆尾、堆顶、向下
50、当前队列中的元素个数是（）。-->3
51、度大于 0 的结点称作（）或（）。-->分支结点、非终端结点
52、度等于 0 的结点称作（）或（）。-->叶子结点、终端结点
53、队列的操作特点是后进（）。-->后出
54、队列的操作特点是先进（）。-->先出
55、队列的特点之一是 E 元素进、出队的次序是：先进（）。-->先出
56、对 16 个元素的序列用冒泡排序法进行排序,共需要进行（）趟冒泡。-->15
57、对 19 个元素的序列用冒泡排法进行排序,通常第 7 趟冒泡中,共需要进行（）次元素间的比较。-->12
58、对 n 个记录的表 r[1…n]进行简单选择排序,所需要进行的关键字间的比较次数为（）。-->n（n－1）/2
59、对 n 个元素的序列进行冒泡排序时,最少的比较次数是（）。-->n－1
60、对二叉链表的访问只能从（）指针开始。-->根
61、对记录序列排序是指按记录的某个关键字排序,记录序列按（）排序结果是唯一的。-->关键字
62、对记录序列排序是指按记录的某个关键字排序,记录序列按（）排序结果是唯一的。-->主关键字
63、对矩阵压缩存储是为了（）。-->节省存储空间
64、对稀疏矩阵进行压缩存储,矩阵中每个非零元素对应的三元组包括该元素的（）、（）和（）三项信息。-->行下标、列下标和非零元素值
65、对稀疏矩阵进行压缩存储,矩阵中每个非零元素对应的三元组包括该元素的（）、（）和非零元素值三项信息。-->行下标、列下标
66、对稀疏矩阵进行压缩存储,矩阵中每个非零元素对应的三元组包括该元素的：行下标、列下标和（）三项信息。-->非零元素
67、对稀疏矩阵进行压缩存储,矩阵中每个非零元素对应的三元组包括该元素的三项信息总是（）。-->行下标列下标数组元素
68、对稀疏矩阵进行压缩存储,矩阵中每个非零元素对应的三元组包括该元素的行下标、列下标和（）三项信息。-->数组元素
69、对稀疏矩阵进行压缩存储,可采用三元组表,设 a 是稀疏矩阵 A 相应的三元组表类型(结构体类型)变量,a 中的一个成员项是三元组类型的结构体数组 data,按书中定义,若 A．data[O]．i=2；A．data[0]．j=3；A．data[对．v=16；它提供的 A 数组的相关信息有（）。-->A 的第一个非零元素的下标为 2,3,元素为 16
70、对稀疏矩阵进行压缩存储,可采用三元组表,一个 6 行 7 列的稀疏矩阵 A 相应的三元组表共有 8 个元素,则矩阵 A 共有个零元素。-->34
71、对稀疏矩阵进行压缩存储,可采用三元组表,一个 8 行 7 列的稀疏矩阵 A 共有 51 个零元素,其相应的三元组表共有（）个元素-->5
72、对稀疏矩阵进行压缩存储,可采用三元组表,一个有 10 行的稀疏矩阵 A 共有 97 个零元素,其相应的三元组表共有 3 个元素。该矩阵 A 有（）列。-->10；
73、对一个长度为 n 的线性表,要删除第 i 个元素,则在顺序表示的情况下,计算复杂性为 O 否,在链式表示的情况下,计算复杂性为（）。-->O（1）。
74、对一组记录(l,3,9,2,12,7,5,1,6)进行直接插人排序(由小到大排序),当把第 6 个记录 7 插人有序表,为寻找插入位置需比较-->3
75、对于长度为 n 的线性表,若进行顺序查找,则时间复杂度为（）。-->O（n）。
76、对于二叉排序树的查找,若根结点元素的键值大于被查元素的键值,则应该在二叉树的（）上继续查找。-->左子树
77、对于二维数组或多维数组,分为按以行为主序和按（）两种不同的存储方式存储。-->以列为主序
78、对于关键字序列(12,13,11,18,60,15,7,20,25,100),用筛选法建堆,必须从键值为（）的关键字开始。-->60
79、对于某一类特定的问题,算法给出了解决问题的一系列操作,每一操作都有它的确切的定义,并在内计算出结果。-->有穷时间
80、对于一个图 G,若边集 E(G)为无向边的集合,则该图为（）。-->无向图
81、对于一个图 G,若边集 E(G)为有向边的集合,则该图为（）。-->有向图
82、对于一棵具有 n 个结点的二叉树,其相应的链式存储结构中共有（）个指针域为空。-->n 十 1
83、对于有向图,顶点 V 的度分为入度和（）。-->出度
84、二叉排序树插入操作中,新插入的结点总是以树的（）结点被插入的。-->叶
85、二叉排序树或者是一棵空树,或者是具有下列性质的一棵二叉树：(1)若左子数不空,则左子树所有结点的值（）。(2)若右子数不空,则右子树所有结点的值（）(3)左右子树又分别是（）-->均小于根结点的值、均大于根结点的值、二叉排序树
86、二叉排序树或者是一棵空树,或者是一棵具有下列性质的二叉排 z 若它的左子树非空,则左子树的所有结点的值都小于它的根结点的值$若它的右子树非空,对 t~右子的所有结点的值都大于(若允许结点有相同的值,则大于等于)它的根结点的值。这种说法是（）的。(回答正确或不正确)。-->不正确
87、二叉树为二叉排序的充分必要条件是其任一结点的值均大于其左孩子的值、小于其右孩子的值。这种说法是（）的。(回答正确或不正确)。-->错误
88、二叉树为二叉排序的充分必要条件是其任一结点的值均大于其左孩子的值、小于其右孩子的值。这种说法是（）的。(回答正确或不正确)。-->不正确
89、二分查找的存储结构仅限于（）。-->有序的顺序存储结构
90、二维数组 A[0…9,0…19]采用列序为主方式存储,每个元素占一个存储单元,并且元素 A[0,0]的存储地址是 200,则元素 A[6,12]的地址是（）。-->332
91、二维数组 A[10…20,5…10]采用行序为主方式存储,每个元素占 4 个存储单元,并且元素 A[10,5]的存储地址是 1000,则元素 A[18,9]的地址是（）。-->1208
92、二叉树中有 1 个 1 度结点,8 个 2 度结点,则该二叉树树共有（）个结点。-->1

93、分块查找又称为（）,它是一种介于（）和折半查找之间的查找方法。-->索引顺序查找、顺序查找
94、该树共有（）个结点。-->11
95、该完全二叉树一共有（）个结点。-->21
96、高度为 8 的平衡二叉树至少有（）个结点。-->54
97、给定-组权重值,构造哈夫曼树,哈夫曼树的高度一定是唯一的,这种说法是（）的。(回答正确或不正确)。-->不正确
98、根据二叉树的定义,具有三个结点的二叉树有（）种不同的形态。-->5
99、根据数据元素间关系的不同特性,通常可分为集合、线性、()、（）、四类基本结构。-->树形、图状
100、根据搜索方法的不同,图的遍历有（）、（）两种方法。-->深度优先搜索遍历、广度优先搜索遍历
101、根据搜索方法的不同,图的遍历有（）、（）两种方法。-->深度优先、广度优先
102、关键字是记录某个（）,用它可以识别、确定一个（）-->数据项的值、记录
103、广义表((a,b),d,e,((1)J),k))的长度是（）-->4
104、广义表((b,a,c),c,d,(e,i,j,k))的表尾是（）。-->（c,d,（e,i,j,k））。
105、广义表((b,a,c),C,d,f,e,((i,j),k))的表头是（）。-->（b,a,c）。
106、广义表(b,a,(c,b),f,e,((i,j),k))的长度是-->6
107、广义表(g,(a,b,d,c),d,e,((i,j),k))的长度是（）-->5
108、广义表(h,(b,a),f,e,(i,j),k))的深度是（）-->3
109、广义表 A((a,b,c),(d,e,f))的表尾为（）。-->《d,e,f）。
110、广义表的((a,C),d,(e,i,j,k))表尾是（）。-->（d,（e,i,j,k））。
111、广义表的(a,(a,b),d,e,((i,j),k))深度是（）。-->3
112、广义表的(a,a,b,d,e,((i,j),k))表头是（）。-->a
113、广义表的(b,(a,b),d,(c,((e,f),j)))深度是（）。-->4
114、广义表的(c,a,(a,b),d,e,((i,j),k))深度是（）。-->3
115、哈夫曼树是带权路径长度（）的树。-->最短
116、哈夫曼树通常权值较大的结点离根（）。-->较近
117、哈夫曼树又称为（）。-->最优二叉树,最小的二叉树
118、哈希表是用来存放查找表中记录序列的表,每一个记录的存储位置是以该记录得到关键字为（）,由相应哈希函数计算所得到的（）。-->自变量、函数值
119、晗希函数是记录关键字的值与该记录（）之间所构造的对应关系。-->存储位置
120、后序遍历二叉树的的操作定义为：若二叉树为空,则为空操作,否则进行如下操作,后序遍历二叉树的（）；后序遍历二叉树的（）,访问而叉树的（）。-->左子树、右子树、根结点
121、既无前驱也没有后继的结点在所在线性表长度为 1,结点指针域的值为（）。-->空
122、键值序列是一个堆。-->A,C,D,E,F,E,F
123、将树中结点赋上一个有着某种意义的实数,称此实数为该结点的（）。-->权
124、将下三角矩阵 A[1．．8,1．．8]的下三角部分逐行地存储到起始地址为 1000 的内存单元中,已知每个元素占 4 个单元,则元素 A[7,5]的地址为（）。-->1100
125、结点的度是指结点所拥有的（）。-->子树树木或后继结点数
126、结点最少的二叉树为（）。-->空的二叉树
127、结点最少的树为（）。-->只有一个结点的树
128、结构中的数据元素存在多对多的关系称为（）结构。-->图状（网状）。
129、结构中的数据元素存在一对多的关系称为（）结构。-->树形
130、结构中的数据元素存在一对一的关系称为（）结构。-->线性
131、仅允许在同一端进行插入和删除的线性表称为（）。-->栈
132、静态链表是用（）描述的链表。-->数组
133、具有 m 个叶子结点的哈夫曼树共有（）结点。-->2m-1
134、具有 N(N-1)/2 条边的无向图成为（）。-->无向完全图
135、具有 N(N-1)/2 条边的有向图成为（）。-->有向完全图
136、具有五层结点的二叉树平衡树至少有（）个结点-->15
137、空串的长度是（）；空格串的长度是（）。-->0；空格字符的
138、连接图-->连接图
139、链表适合从指点结点开始,寻找直接前趋的运算。-->双向
140、链表相对于顺序表的优点有插入和（）操作方便。-->删除
141、链式存储结构是把逻辑上相邻的结点存储在物理上（）的存储单元里,节点之间的逻辑关系由附加的指针域来体现。-->任意
142、两个串相等的充分必要条件是（）。-->两个串的长度相等且对应位置的字符相同
143、两个串相等的克分必要条件是（）。-->串长度相等且对应位置的字符相等
144、路径长度是指一条路径上经过的边的（）-->数目
145、冒泡排序是一种比较简单的（）方法。-->交换排序
146、某二叉树的前序遍历结点顺序为 abdgcefg,中序遍历结点顺序为 dgbacchf,则后序遍历的结点顺序为-->gdbehfca
147、某人想要在第 8 个元素前插入 1 个元素 7(也就是插入元素作为新表的第 8 个元素),他的做法是从第 8 号元素开始,直到第 25 号元素依次向后移动 1 个位置,然后把 7 存放在 8 号位置,其结果是新表中第 25 号元素的值为（）。-->8
148、排序不需要进行记录关键字间的比较。-->基数
149、判断一个循环队列 LU(最多元素为 m)为空的条件是（）。-->
150、平均查找长度是指为确定记录在查找表中的位置,需要与给定值进行比较的关键字个数的（）。-->数学期望值
151、求两个 n 阶矩阵的乘积,算法的基本操作为（）,时间复杂度为（）。-->乘法 O（n3）。
152、如果 t2 是由序树 t 转换而来的二叉树,那么 t 中的结点的后序就是（）结点中的中序。-->t2
153、如果某二叉树的前序为 s t u w v ,中序为 uwtvs,那么二叉树的后序为-->wuvts
154、如果线性表的存储空间变化较大,则适用（）表。-->链
155、入度是以该顶点为终点的入边（）。-->数目
156、若二叉树中度为 2 的结点有 15 个,则该二叉树有（）个叶子结点。-->16
157、若二叉树中有 20 个叶子结点,则该二叉树有（）个度为 2 的结点-->19
158、若连通网络上各边的权值均不相同,则该图的最小生成树有()棵。-->1
159、若图 G 中任意两个顶点都连通,则称 G 为-->连接图
160、若以 4,5,6,7,8 作为叶子结点的权值构造哈夫曼树,则其带权路径长度是（）。-->69
161、设 G 为具有 N 个顶点的无向连通图,则 G 至少有（）条边。-->N-1
162、设 x,y 是图 G 中的两顶点,(x,y)与(y,x)是（）的两条弧。-->有向
163、设 x,y 是图 G 中的两顶点,则(x,y)与(y,x)被认为（）,-->无向
164、设顺序队列的类型为 typedef( )struct(ElemTypedata[MaxSie];-->sq->rear+十；
165、设一棵哈夫曼树共有 10 个叶结点,则该树共有（）个结点。-->19
166、设一棵哈夫曼树共有 18 个非叶结点,则该树总共有（）个结点。-->37
167、设一棵完全二叉树,其最高层上最右边的叶结点的编号为奇数,该叶节点的双亲结点的编号为 10,该完全二叉树一共有()个结点。-->21
168、设有 n 阶对称矩阵 A,用数组 s 进行压缩存储,当 i≥j 时,A 的数组元素 aij 相应于数组 s 的数组元素的下标为（）。(数组元素的下标从 1 开始)-->i（i-1）/2+j
169、设有 n 阶对称矩阵 A,用一维数组压缩存储 A 的下三角元素,S 的下标从零开始,最后一个元素的下标为 27,则 n=（）。(矩阵中的第 1 个元素是 a1,1)。-->7
170、设有串 p1=“ABADF“,P2=“ABAFD“,P3=“ABADFA“P4=“ABAF“,四个串中最大的是（）。-->p2
171、设有串 pl=“DEADFG“,PZ=“DEAFDF“,P3=“DEADFAB“,P4=“DEAFE“,四个串中最大的是（）-->P4
172、设有一个长度为 18 的顺序表,第 8 号元素到第 18 号元素依次存放的值为 8,9．18．某人想要删除第 8 号元素,程序中他的做法是用语句 forO=18；i<=9；i 一一)a[i-1]=a[1]；即从第 18 号元素开始,直到第 9 号元素,每个元素依次向前(左)移动 1 个位置,事实上这样做是错误的,其结果新表中第 9 号元素的值为（）。-->18
173、设有一个长度为 18 的顺序表,要在第 4 个元素之前插入 2 个元素(也就是插入元素作为新表的第 5 个和第 4 个元素)．则最少要移动元素的个数为（）。-->15
174、设有一个长度为 20 的顺序表,第 8 号元素到第 20 号元素依次存放的值为 8,-->8
175、设有一个长度为 20 的顺序表,要插人一个元素,并作为第 6 个元素,需移动元素的个数为（）-->15
176、设有一个长度为 20 的顺序表,要插入一个元素,并作为第 8 个元素,需移动元素的个数为（）。-->13
177、设有一个长度为 20 的顺序表,要删除第 5 个元素,则最少要移动元素的个数为（）。-->15
178、设有一个长度为 25 的顺序表,第 8 号元素到第 25 号元素依次存放的值为 8,9,10,11,…,25,某人想要删除第 8 个元素,他的做法是从第 25 号元素开始,直到第 9 号元素依次向前移动 1 个位置,其结果新表中第 9 号元素的值为（）。-->25
179、设有一个长度为 42 的顺序表,要删除第 9 个元素需移动元素的个数为（）-->33
180、设有一个带头结点的,头指针为 head 的单向链表,p 指向表中某一个结点,且有 p→next=为=NU153、,现要删除头结点,并使该单向链表构造成单向循环链表,通过操作 head=head 一>next；-->

181、设有一个单向链表,结点的指针域为 next,头指针为 head,p 指向尾结点,为了使该单向链表改为单向循环链表,可用语句（）。-->p->next=head；
182、设有一个单向循环链表,结点的指针域为 next,头指针为 head,指针 P 指向表中某结点,若逻辑表达式（）的结果为真,则 A 所指结点为尾结点。-->p->next==head
183、设有一个单向循环链表,头指针为 head,链表中结点的指针域为 next,p 指向尾结点的直接前驱结点,若要删除尾结点,得到一个新的单向循环链表,可执行操作（）。-->p->next=head；
184、设有一个非空的链钱,钱顶指针为 h,要进行出钱操作,用 x 保存出钱结点的值,找结点的指针域为 next,则可执行 x=h 一>data;（）。-->hs=hs->next；
185、设有一个空栈,现输入序列为 1,2,3,4,5。经过 push,push,pop,push,pop,push,pop,push 后,输出序列是（）。-->234
186、设有一个链钱,钱顶指针为 h,现有一个所指向的结点要人钱,则可执行操作（）和 hs=s。-->s->next=hs；
187、设有一个顺序栈 S,元素 s1,s2,s3,s4,s5,s6 依次进栈,如果 6 个元素的出栈顺序为 s2,s3,s4,s6,s5,s1,则顺序栈的容量至少为（）。-->3
188、设有一个头指针为 head 的单向链表,p 指向表中某一个结点,且有 p->next==NULL,通过操作（）,就可使该单向链表构形成单向循环链表。-->p->next=head；
189、设有一个头指针为 head 的单向链表,p 指向链表中的某结点,若要使该链表成为单向循环链表,可用语句 while(p 一>next!=NULL)（）；和 p 一>next=head；-->P=p->next
190、设有一个头指针为 head 的单向循环链表,p 指向链表中的结点,若 p->next=（）,则 p 所指结点为尾结点。-->head
191、设有一棵深度为 4 的完全二叉树,第四层上有 5 个结点,该树共有（）个结点。(根所在结点为第 1 层)。-->12
192、设有一棵深度为 5 的完全二叉树,第 5 层上有 4 个结点,该树共有（）个结点。(根所在结点为第 1 层)-->19
193、设有一棵深度为 5 的完全二叉树,该树共有 20 个结点,第五层上有（）个叶结点。(根所在结点为第 1 层)。-->5
194、设有一棵深度为 5 的完全二叉树,该树共有 21 个结点,第 5 层上有（）个结点。(根所在结点为第 1 层)。-->6
195、设有一棵有 38 个结点的完全二叉树,该树共有层。(根所在结点为第 1 层)。-->6
196、深度为 5 的二叉树至多以有（）个结点。-->31
197、实现任意二叉树的后序遍历的非递归算法而不适用栈结构,最佳的二叉树方法是采用（）的存储结构-->三叉列表
198、树的带权路径长度为树中所有叶子结点的（）。-->带权路径长度之和
199、树的度是指（）。-->树中所有结点的度的最大值
200、树的深度或高度是指（）。-->树中结点的最大层数
201、树最适合用来表示元素之间具有（）的数据-->分支层次关系
202、数据的（）在计算机中的表示称为物理结构。-->逻辑结构
203、数据的存储结构可用 4 种基本的存储方法表示,它们分别是顺序存储、链式存储、索引存储和（）。-->散列存储
204、数据的逻辑结构包括线性结构、树形结构和图形结构 3 种类型,树型结构和有向图结构合称为（）。-->非线性结构
205、数据的逻辑结构可以分类为线性结构和（）结构两大类。-->非线性
206、数据的逻辑结构在计算机存储器内的表示,称为数据的（）。-->存储结构
207、数据的逻辑结构在计算机中的表示称为（）或（）。-->存储结构；物理结构
208、数据的逻辑结构在计算机中的表示称为（）结构。-->物理（存储）。
209、数据的逻辑结构在计算机中的表示称为（）结构。-->存储结构
210、数据结构按结点间的关系,可分为 4 种逻辑结构：（）、（）、（）、（）、（）。-->集合；线性结构；树形结构；图状结构
211、数据结构包括数据的逻辑结构、数据的存储结构和（）三方面的内容。-->数据的运算
212、数据结构的形式定义为：数据结构是一个（）元组。-->二
213、数据结构在物理上可分为顺序存储结构和回答链式（）存储结构。-->链式
214、数据结构中,数据元素之间的抽象关系称为（）结构。-->逻辑
215、数据结构中的数据元素存在多对多的关系称为（）结构。-->图状结构
216、数据结构中的数据元素存在多对多的关系称为（）结构。-->图状
217、数据结构中的数据元素存在一对多的关系称为（）结构。-->树形
218、数据结构中的数据元素存在一对多的关系称为（）结构。-->树形结构
219、数据结构中的数据元素存在一对一的关系称为（）结构。-->线性结构
220、数据元素可由若干个（）组成。-->数据项
221、数据元素之间的抽象关系称为（）结构。-->逻辑
222、数组 a 经初始化 chara[]=“English“；a[7]中存放的是（）。-->字符串的结束符
223、数组 a 经初始化 chara[]=“fhglisp“；a[6]中存放的是（）。-->字符 p
224、顺序表相对于链表的优点有随机访问和（）。-->空间利用率高
225、顺序表中逻辑上相邻的元素物理位置（）紧邻,单链表中逻辑上相邻的元素物理位置{=不一定}紧邻。-->一定
226、顺序查找法的平均查找长度为（）。-->（n＋1）/2
227、顺序存储结构是把逻辑上相邻的结点存储在物理上（）的存储单元里,结点之间的逻辑关系由存储单元位置的邻接关系来体现。-->连续
228、顺序存储字符串“ABCD',需要占用（）个字节。-->5
229、四类基本结构分别为（）结构。-->集合、线性、树形、图状
230、算法分析的两个主要方面是回答时间（）复杂度和空间复杂度。-->时间
231、算法具有如下特点：-->有穷性
232、通常可以把某城市中各公交站点间的线路图抽象成（）结构。-->图状
233、通常可以把一本含有不同章节的书的目录结构抽象成（）结构。-->树形
234、通常数据的逻辑结构包括（）、（）、（）、（）四种类型。-->集合、线性、树形、图状
235、通常数据的逻辑结构包括集合、线性、（）、（）四种类型。-->树形、图状
236、图常用的两种存储结构是（）和（）。-->邻接矩阵、邻接表
237、图的遍历是从图的某一顶点出发,按照一定的搜索方法对图中（）各做（）访问的过程。-->所有顶点；一次
238、图的广度优先搜索类似于树的（）遍历。-->按层次
239、图的深度优先搜索遍历类似于树的（）遍历。-->先序
240、图的深度优先搜索和广度优先搜索序列不一定是唯一的。此断言是（）的。(回答正确或不正确)。-->正确
241、我们把每种数据结构均视为抽象类型,它不但定义了数据的表示方式,还给出了处理数据的（）。-->实现方法
242、无向图 G 中极大连通子图称为 G 的（）。-->连通分量
243、稀疏矩阵存储时,采用一个由（）、（）、非零元 3 部分信息组成的三元组唯一确定矩阵中的一个非零元素。-->行号、列号
244、先序遍历二叉树的的操作定义为；若二叉树为空,则为空操作,否则进行如下操作,访问二叉树的（）；先序遍历二叉树的（）,先序遍历二叉树的（）。-->根结点、左子树、右子树
245、现要删除头结点,并使该单向链表构造成单向循环链表,通过操作 head=head 一>next；-->p->next=head；
246、线性结构反映结点间的逻辑关系是（）的,非线性结构反映结点间的逻辑关系是一对多或多对多。-->一对一
247、线性链表的逻辑关系是通过每个结点指针域中的指针来表示的。其逻辑顺序和物理存储顺序不再一致,而是一种（）存储结构,又称为（）。-->链式、链表
248、向→个钱顶指针为 h 的链战中插入一个所指结点时,可执行()和 h=s；操作。(结点的指针域为 next)。-->s->next=h；
249、向-个钱顶指针为 h 的链战中插入一个 s 所指结点时,可执行 s->next=h；和（）。-->h=s；
250、向一个链式栈插入一个新结点时,首先把栈顶指针的值赋给(),然后把新结点的存储位置赋给栈顶指针。-->新结点的指针域
251、向一个钱顶指针为 top 的链战中插入一个 p 所指结点时,可执行（）操作。(填两条语句,结点的指针域为 next)。-->p->next=top；top=p；
252、向一个顺序栈插入一个元素时,首先使（）后移一个位置,然后把待插入元素写入到这个位置上。-->栈顶指针
253、向一个栈顶指针为 h 的链栈中插入一个 s 所指结点时,可执行（）操作。(结点的指针域为 next)-->s->next=h；和 h=s；
254、序列 12,10,13,11,16,14,采用冒泡排序算法,经一趟冒泡后,序列的结果是（）(按升序排序)。-->10,12,11,13,14,
255、序列 34,32,35,33,38,36,采用冒泡排序算法(升序),经一趟冒泡后,序列的结果是（）-->32,34,33,35,36,38
256、序列 4,2,15,13,18,16,采用冒泡排序算法,经一趟冒泡后,序列的结果是（）-->2,4,13,15,16,18
257、序列 4,2,53,8,6,7,9,采用归并排序算法(升序),经一趟归并后,序列的结果：（）。-->2,4,3,5,6,8,7,9
258、序列 5,3,8,4,7,6,采用冒泡排序算法,经一趟冒泡后,序列的结果是（）。(按升序排序)。-->3,5,4,7,6,8

259、序列 7,1,4,2,5,3,8,6 用归并法排序(升序),经一次归并后的结果序列是（）-->1,7,2,4,3,5,6,8
260、循环队列的队头指针为 f,队尾指针为 r,当（）时表明队列-->r=f
261、循环队列的引入,目的是为了克服（）。-->假上溢
262、循环队列的最大存储空间为 MaxSize=-->
263、循环队列的最大存储空间为 MaxSize=6,采用少用一个元素空间以有效地判断栈空或栈满,若队头指针 front=4,当队尾指针 rear=（）时队满,队列中共有（）个元素。-->3,5
264、循环队列队头指针在队尾指针（）位置,队列是“满”状态。-->下一个
265、循环队列用 a[0],…,a[7]的一维数组存放队列元素,(采用少用一个元素的模式),设 front 和 rear 分别为队头和队尾指针,且 front 和 rear 的值分别为 2 和 7,当前队列中的元素个数是（）。-->5
266、循环队列在规定少用一个存储空间的情况下,队空的判定条件为（）。-->front==rear
267、循环链队列中,设 from 和 rear 分别为队头和队尾指针,(最多元索为 MexSize,采用少用一个元素的模式),判断循环链队列为满的条件为（）。-->front==（rear+1）%MaxSize
268、循环链队列中,设 front 和 rear 分别为队头和队尾指针,(最多元素为 MaxSize,),判断循环链队列为空的条件是（）为真。
-->front==rear
269、要求在 n 个数据元素中找其中值最大的元素,设基本操作为元素间的比较。则比较的次数和算法的时间复杂度分别为（）和（）。
-->n-1；O（n）。
270、要求在 n 个数据元素中找值最大的元素,其基本操作为（）。算法的时间复杂度为（）。-->元素间的比较 O（n）。
271、要在一个单向链表中 p 所指向的结点之后插入一个 S 所指向的新结点,若链表中结点的指针域为 next,可执行（）和 p 一>next==s,的操作。-->s->next=p->next；
272、要在一个单向链表中删除 P 所指向的结点,已知 q 指向 P 所指结点的直接前驱结点,若链表中结点的指针域为 next,则可执行（）。-->q->next=p->next；
273、一个钱和一个队列的输入序列都为 abcdefg,它们可能有相同的输出序列吗？（）。(若没有则回答没有,若有则写出序列,进找出钱可以交替进行)。-->abcdefg
274、一个算法的时间复杂度是用该算法回答所消耗的时间（）的多少来度量的,一个算法的空间复杂度是用该算法在运行过程中所占用的存储空间的大小来度量的。-->所消耗的时间
275、一个有序表(3,4,10,14,34,43,46,64,75,78,90,96,130)用折半查找法查找值为 90 的结点,经（）次比较后查找成功。-->4
276、一棵 3 度的树,其中 3 度结 1 个,2 度结 2 个,1 度结 2 个,则该树共有（）个叶结点。-->5
277、一棵二叉树,有 1 个 2 度结点,,2 个 1 度结点,则该树共有（）-个结点。-->5
278、一棵二叉树没有单分支结点,有 6 个叶结点,则该树总共有（）个结点。-->11
279、一棵二叉树顺序编号为 6 的结点(树中各结点的编号与等深度的完全二叉中对应位置上结点的编号相同),若它存在右孩子,则右孩子的编号为（）。-->13
280、一棵二叉树叶结点(终端结点)数为 5,单分支结点数为 2,该树共有（）个结点。-->11
281、一棵二叉树中顺序编号为 5 的结点(树中各结点的编号与等深度的完全二叉中对应位置上结点的编号相同),若它存在左孩子,则左孩子的编号为（）。-->10
282、一棵二叉树中顺序编号为 i 的结点,若它存在左、右孩子,则左右孩子的编号分别为（）、（）。-->2i、2i+1
283、一棵二叉树中有 n 个非叶结点,每一个非叶结点的度数都为 2,则该树共有（）个叶结点。-->n+1
284、一棵二叉树没有单分支结点,有 6 个叶结点,则该树总共有（）个结点。-->11
285、一棵有 12 个结点的二叉树,采用链式存储,其中共有（）个指针域为空。-->13
286、一棵有 15 个结点的哈夫曼树,采用链式结构存储,该树结构中有（）个叶结点。-->8
287、一棵有 16 个叶结点的哈夫曼树,则该树共有（）个非叶结点。
-->15
288、一棵有 18 个结点的二叉树,其 2 度结点数的个数为 8,则该树共有（）个 1 度结点。-->1
289、一棵有 18 个叶结点的哈夫曼树,则该树共有（）个非叶结点。
-->17
290、一棵有 19 个结点的二叉树,采用链式结构存储,该树结构中有（）个指针域为空。-->20
291、一棵有 2n-l 个结点的二叉树,其每一个非叶结点的度数都为 2,则该树共有（）个叶结点。-->n
292、一棵有 7 个权重值构造的哈夫曼树,共有（）个结点。-->18
293、一棵有 8 个权垂值构造的哈夫曼树,共有（）个结点。-->15
294、一棵有 N 个顶点的生成树有且仅有（）条边。-->N-1
295、一棵有 n 个叶结点的二叉树,其每一个非叶结点的度数都为,2 则该树共有（）个结点。-->2n 一 1
296、一棵有 n 个叶结点的哈夫曼树,则该树共有（）个结点。-->2n-1
297、一颗二叉树的叶结点在前序、中序,后序遍历中的相对次序不发生改变-->任何
298、一裸有 7 个权重值构造的哈夫曼树,共有（）个结点。-->13
299、一维数组的逻辑结构是（）。-->线性结构
300、已知某二叉树的后序遍历为 dabcc,中序遍历为 debac,则它的前序遍历为（）。-->cedba
301、已知一个图的邻接矩阵表示,删除所有从 i 个结点出发的边的方法是将矩阵的第 i 行全部置为（）。-->0
302、用 S 表示入栈操作,X 表示出栈操作,若元素入栈顺序为 1234,为了得到 1342 出栈顺序,相应的 S、X 操作串为（）。-->SXSSXSXX
303、有 12 个结点的平衡二叉树的最大深度是（）。-->5
304、有向图 G 中极大强连通子图称为 G 的（）。-->强连通分量
305、有向图顶点 V 的度等于其（）和出度之和。-->入度
306、有一个 10 阶对称矩阵 A,采用压缩存储方式,以行序列为主存储,且 A[0][0]=1,则 A[8][5]的地址是（）。-->42
307、在-个链队中,f 和 r 分别为队头和队尾指针,队结点的指针域为 next,则插入一个所指结点的操作为（）；r=s。-->r->next=s
308、在 C 语言中,分别存储“S”,’和“s“,,各需要占用（）字节．-->两个和 1 个
309、在 n 个结点的顺序表中插入一个结点需平均移动（）个结点。
-->n/2
310、在 n 个整数中求最大数的算法中,其基本操作是（）。-->元素间的比较
311、在（）遍历二叉树的序列中,任何结点的子树上的所有结点,都是直接跟在该结点之后。-->先序
312、在插入排序、希尔排序、选择排序、快速排序、堆排序、归并排序和基数排序中,平均比较次数最少的排序是（）。-->快速排序
313、在插入排序、希尔排序、选择排序、快速排序、堆排序、归并排序和基数排序中,需要内存容量最多的是（）。-->基数排序
314、在插入排序和选择排序中,若初始数据基本反序,则选用（）。
-->选择排序
315、在插入排序和选择排序中,若初始数据基本正序,则选用（）；
-->插入排序
316、在查找表中,通过记录的某关键字能唯一地确定一个记录,该关键字称为（）。-->主关键字
317、在长度为 n 的顺序表中插入一个元素的时间复杂度为（）。
-->O（n）。
318、在带表头结点的单链表中,当删除某一指定结点时,必须找到该结点的（）结点。-->前驱
319、在单链表中,任何两个元素的存储位置之间都有固定的联系,因为可以从回答头结点（）进行查找任何一个元素。-->头结点
320、在单链表中除首结点外,任意结点的存储位置都由（）结点中的指针指示。-->直接前驱
321、在单向链表中,q 指向 p 所指结点的直接后继结点,要删除 q 所指结点,可以用操作（）=q->next；-->p->next；
322、在堆排序和快速排序中,若原始记录接近正序和反序,则选用（）,若原始记录无序,则最好选用（）。-->堆排序、快速排序
323、在堆排序和快速排序中,若原始记录接近正序或反序,则选用（）。-->堆排序
324、在堆排序和快速排序中,若原始记录无序,则最好选用（）。
-->快速排序
325、在队列的顺序存储结构中,当插入一个新的队列元素时,（）指针的值增 1,当删除一个元素队列时,（）指针的值增 1。-->尾、头
326、在对 10 个记录的序列(14,30,10,7,22,13,66,85,47,58)进行直接插入排序时,当把第 6 个记录 13 插入到有序表时,为寻找插入位置,元素间需比较（）次。(由小到大排列)-->4
327、在对 10 个记录的序列(8,36,19,78,4,10,53,45,27,68)进行直接插人排序时,当把第 6 个记录 10 插人到有序表时,为寻找插人位置,元素间需比较（）次。-->4
328、在对 11 个记录的序列(12,35,9,7,2,11,56,95,37,58,60)进行直接插入排序时,当把第 6 个记录 11 插入到有序表时,为寻找插入位置,元素间需比较（）次。(由小到大排列)。-->3
329、在对一组记录(44,30,87,11,8,64,58,37,78)进行直接插人排序时,当把第 8 个记录 37 插入到有序表时,为寻找插入位置需比较（）次
-->5
330、在对一组记录(50,34,92,19,11,68,56,41,79)进行直接插入排序(由小到大排序),当把第 8 个记录 41 插入到有序表时,为寻找插入位置需比较（）次。-->5

331、在对一组记录(50,40,95,20,15,70,60,45,80)进行直接插入排序时,当把第 7 个记录 60 插入到有序表时,为寻找插入位置需要比较（）次。-->3
332、在对一组记录(54,38,96,23,15,72,60,45,83)进行直接插入排序时,当把第 8 个记录 45 插入到有序表时,为寻找插入位置需比较（）次。-->5
333、在对一组记录(55,39,97,22,16,73,65,47,88)进行直接插人排序时,当把第 7 个记录 65 插人到有序表时,为寻找插人位置需比较（）次。(由小到大排序)。n 个元素进行冒泡法排序,第 J 趟冒泡要进行（）次元素间的比较。-->3
334、在对一组记录(55,39,97,22,16,73,65,47,88)进行直接插人排序时,当把第 7 个记录 65 插入到有序表时,为寻找插入位置需比较（）次。-->3
335、在对一组记录(55,39,97,22,16,73,65,47,88)进行直接插入排序时,当把第 7 个记录 65 插入到有序表时,为寻插入位置需比较（）次。(由小到大排序)。-->3
336、在对一组记录（）进行直接插入排序(由小到大排序),当把第 7 个记录 46 插入到有序表时,为寻找插入位置需比较（）次。-->3
337、在对一组序列(35,19,77,2,6,53,55,27,26,98)进行直接插人排序时,当把第 9 个记录 26 插人到有序表时,为寻找擂人位置需进行（）次元素间的比较．-->6
338、在二叉树的链式存储结构中,通常每个结点中设置三个壤,它们是值域、（）、（）。-->左指针、右指针
339、在二叉树的链式存储结构中,通常每个结点中设置三个域,它们是（）、（）、右指针。-->值域、左指针、右指针
340、在各种查找方法中,平均查找长度与结点个数 n 无关的查找方法是（）。-->哈希表查找法
341、在具有 6 个结点的无向简单图中,当边数最少为（）条时,才能确保该图一定的连通图。-->5
342、在求表达式值的算符优先算法中使用的主要数据结构是（）。-->栈
343、在散列函数 H(key)=key%p 中,p 应取（）。-->素数
344、在树形结构中,数据元素之间存在（）的关系。-->一对多
345、在双链表中,每个结点有两个指针域,一个指向前驱结点,另一个指向（）。-->后续结点
346、在双向链表中,要删除 p 所指的结点,可以先用语句(p->next)->prior=(p->prior)；然后再用语句(p->prior)->next=（）。-->p->next;
347、在双向链表中,要删除 p 所指的结点,可以先用语句句一>prior)一>next=p 一>next;然再用语句()。-->(p->next)->prior=p->prior;
348、在双向链表中,要删除 p 所指的结点,其中所用的一条语句(P一>prior)一>next=p 一>next;的功能是:使 P 所指结点的直接前驱的右指针指向（）。-->P 所指结点的直接后继
349、在双向链表中,要在 p 所指的结后插入 q 所指的结点(设 q 所指的结点 C. 贼值),可以先用语句 q 一>next=p 一>next;(p 一>next)一>prior=q;然后再用语句 q 一>prior=p;和语句()。-->p->next=q;
350、在双向链表中,要在 p 所指结点后插入 q 所指的结点(设 q 所指的结点已赋值),其中所用的一条语句句一>next)一>prior=q;功能是使 P 所指结点的（）指向 q。-->直接前驱的左指针
351、在顺序表中访问任意一个结点的时间复杂度均为 O(1),因此,顺序表也称为（）的数据结构。-->随机访问
352、在图状结构中,每个结点的前驱结点数和后继结点数可以()。-->有多个
353、在无权图 G 的邻接矩阵 A 中,若(vi,vj)或(vi,vj)属于图 G 的边集,则对应元素 A[i][j]等于（）。-->1
354、在无向图中,如果从顶点 v 到顶点 v′有路径,则称 v 和 v′是()。-->连通
355、在循环链表中,可根据任一结点的地址遍历整个链表,而单链表中需知道（）才能遍历整个链表。-->头指针
356、在一非空二叉树的中,根结点的右边只有（）上的所有结点-->右子树
357、在一个不带头结点的非空链队中,f 和 r 分别为队头和队尾指针,队结点的数据域为 data,指针域为 next,若要进行出队操作,并用变量 x 存放出队元素的数据值,则相关操作为 x=f 一>data；（）。-->f=f->next;
358、在一个不带头结点的非空链队中,f 和 r 分别为队头和队尾指针,队结点的数据域为 data,指针域为 next,若要进行出队操作,并用变量错存放出队元素的数据值,则相关操作为（）；（）。-->x=f->data,f=f->next;
359、在一个查找表中,能够唯一地确定一个记录的关键字称为()。-->主关键字
360、在一个长度为 n 的顺序存储结构的线性表中,向第 i(1£i£n+1)个元素之前插入新元素时,需向后移动（）个数据元素。-->n-i+1
361、在一个带头结点的链队中,设 front 和 rear 分别为队头和队尾指针,则删除一个结点的操作为 p=front 一>next;（）=p 一>next;(结点的指针域为 next,p 为辅助用指针)。-->front->next
362、在一个单链表中 p 所指结点之后插入一个 s 所指结点时,应执行（）和 p->next=s;的操作。-->s->next=p->next;
363、在一个单向链表中,要删除 p 所指结点,已知 q 指向 p 所指结点的前驱结点。则可以用操作（）。-->q->next=p->next;
364、在一个单向链表中,要删除 p 所指结点的直接后继结点．则可以用操作（）。(用一条语句)。-->p->next=p->next->next;
365、在一个单向链表中,已知 q 指向 p 所指结点的直接前驱结点,现要删除 p 所指结点,则可以用操作 q 一>next=（）-->P->next
366、在一个单向链表中,已知 q 指向 p 所指结点的直接前驱结点,现要删除 p 所指结点,则可以用操作（）。-->q->next=p->next
367、在一个单向链表中 p 所指结点之后插人一个所指向的结点时,应执行一>next=p->next;和（）的操作。-->p->next=s;
368、在一个单向链表中 p 所指结点之后插入一个 s 所指向的结点时,应执行（）和 p->next=s;的操作。-->s->next=p->next;
369、在一个链队中,f 和 r 分别为队头和队尾指针,队结点的指针域为 next,指向一个要人队的结点,则人队操作为（）；（）。-->r->next=s,r=s;
370、在一个链队中,设 front 和 rear 分别为队头和队尾指针,则 s 所指结点(数据域已赋值；的入队操作为 s 一>next=NULL；．（）和 rear=s;-->rear->next=s;
371、在一个链队中,设 f 和 r 分别为队头和队尾指针,则插入 s 所指结点的操作为（）和(结点的指针域为 next)-->r=s;r->next=s;
372、在一个链队中,设 f 和 r 分别为队头和队尾指针,则插入 s 所指结点的操作为（）和 r=s;(结点的指针域为 next)。-->r->next=s;
373、在一个链队中,设 f 和 r 分别为队头和队尾指针,则删除一个结点的操作为（）。(结点的指针域为 next)-->f=f->next;
374、在一个链式队列中,若队头指针与队尾指针的值相同,则表示该队列为（）或该队列只含有一个结点。-->空
375、在一个链式栈中,若栈顶指针等于 NULL,则为（）；-->空栈
376、在一个顺序栈中,若栈顶指针等于（）,则为空栈；-->一1
377、在一个图中每条边可以表上具有某种含义的数值,该数值称为（）。-->权
378、在一个无向图中,若存在一条边,则称 Vi 和 Vj 为该边的两个端点,并称他们互为（）。-->邻接点
379、在一个有向图中,所有顶点入度之和等于所有顶点出度之和的（）倍。-->1
380、在一棵二叉树中,若编号为 1 的结点存在右孩子,则右孩子的顺序编号为（）。-->2i+1
381、在一棵二叉树中,若编号为 1 的结点是其双亲结点的左孩子,则 i 结点的双亲结点的顺序编号为（）。-->i/2
382、在一棵二叉树中,若编号为 i 的结点存在右孩子,则右孩子的顺序编号为（）。-->2i+1
383、在一棵树中,每个结点的（）或者说每个结点的（）称为该结点的（）,简称为孩子。-->子树的根、后继结点、孩子结点
384、在有向图 G 中,若任意两个顶点 Vi 和 Vj 都连通,从 VI 到 Vj 和从 Vj 到 Vi 都存在路径,则称该图为（）。-->强连通图
385、在有向图的邻接矩阵上,由第 i 行可得到第 i 个结点的出度,而由第 j 列可得到第回答个结点的入度。-->j
386、在栈的 ADT 定义中,除初始化操作外,其他基本操作的初始条件都要求（）。-->栈存在
387、在作出栈运算时应先判别栈是否（）。-->空
388、在作入栈运算时应先判别栈是否（）。-->满
389、则当队尾指针 rear=（）时,队列为空,当 rear=（）时,队列有 6 个元素。-->4、2
390、则该树共有（）个结点-->15
391、栈和队列的操作特点分别是（）和（）。-->后进先出、先进先出
392、栈又称为后进先出表,队列又称为（）表。-->先进先出
393、栈元素的进、出栈次序是：后进（）。-->先出
394、折半查找又称为（）。使用该查找算法的前提条件是,查找表中记录相应的关键字值必须按（）。-->二分查找、升序或降序排列
395、折半查找只适用于（）的有序表。-->顺序存储结构
396、中序遍历（）树可得到一个有序序列。-->二叉排序树
397、中序遍历二叉排序树可得到一个（）的序列。-->有序
398、中序遍历二叉树的的操作定义为；若二叉树为空,则为空操作,否则进行如下操作,中序遍历二叉树的（）；访问而叉树的（）,中序遍历二叉树的（）。-->左子树、根结点、右子树
399、中序遍历一棵（）树可得到一个有序序列。-->二叉排序树
400、字符串 a1=“heijing“．a2=“hef“．a3=“heifang“,a4=“hefi“中最小的是（）。-->a2
401、字符串 a1=“teijing“,a2=“tef“,a3=“teifang“,a4=“tefi“最小的是（）。-->a2
402、组成串的数据元素只能是（）。-->字符

综合题(102)-

1、按折半查找对该表进行查找,求在等概率情况...
2、对给定的数列 b={6,15,3,7,19,8,5,17,4}...
3、对给定权值 2,1,3,3,4,5,
4、对给定权值 3,1,4,4,5,6,构造深度为 5 的哈夫...
5、对给定权值 4,2,6,6,7,8,构造高度为 4 层的哈...
6、对关键字序列（36，69，46，28，30，74）采用快速排序，以第...
7、对关键字序列（56，51，71，54，46，106），利用快速排序，以...
8、对于一个无向图，假定采用邻接矩阵表示，试分别...
9、给定数列(8,17,5,9,21,10,7,19,6),依次取序...
10、画出对长度为 10 的有序表进行折半查找的判定...
11、假设通信用的报文由 9 个字母 A、B、C、D、E、...
12、简述拓扑排序的步骤。
13、利用筛选法,把序列{37,77,62,97,11,27,62,4...
14、利用筛选过程把序列{42,82,67,102,16,32,57...
15、利用筛选过程把序列{42,82,67,102,16,32,57...
16、请根据以下带权有向图 G
17、如图所示,若从顶点 a 出发,首先经过 C 按图的深...
18、如图所示,若从顶点 a 出发,首先经过顶点 C.按...
19、如下的一棵二叉棵树
20、如下的一棵二叉棵树,
21、如下的一棵树,给出先序遍历序列...
22、如下是一棵二叉排序树,A1,A2,……A9 代表 1,2...
23、如下为一个长度为 10 的有序表,给出按折半查...
24、设 headI 和 PI 分别是不带头结点的单向链表 A 的...
25、设查找表为(16,15,20,53,64,7)，用冒泡法对该...
26、设查找表为(20,19,24,57,68,11)
27、设查找表为{1,10,11,14,23,27,29,55,68},元...
28、设查找表为{1,2,3,4,5,6,7,8,9,10,11}
29、设查找表为{1,2,3,4,5,6,7,8,9,10,11}
30、设查找表为{10,18,24,28,45,58,68,80,88,89...
31、设查找表为{16,15,20,53,64,7},
32、设查找表为{5,6,7,8,9,10,11,12,13,14}...
33、设查找表为{50,60,75,85,96,98,105,110,120...
34、设查找表为：
35、设查找表为：
36、设查找表为：用折半查找在该查找表成功查找到...
37、设关键字序列为：(36,69,46,28,30,74)...
38、设关键字序列为：(36,69,46,28,30,74),将此序...
39、设关键字序列为：（36，69，46，28，30，74），将此序列用快速...
40、设关键字序列为：（36，69，46，28，30，74），用冒泡法对上述...
41、设某二叉树先序遍历为 abdec,中序遍历为 dbea...
42、设某二叉树先序遍历为 abdec，中序遍历为 dbeac...
43、设某二叉树先序遍历为 abdec，中序遍历为 dbeac...
44、设数据集合 a={6,17,10,13,8,15,12,18,14}...
45、设数据集合 a={7,4,9,8,6,5,3},依次取 a 中各...
46、设数据序列为：{53,30,37,12,45,24,96)...
47、设数据序列为：{53,30,37,12,45,24,96}，从空二...
48、设一组记录的关键字序列为（49，83，59，41，43，47），采用...
49、设有查找表{17,26,14,16,15,30,18,19,28},...
50、设有查找表{5,14,2,6,18,7,4,16,3},依次取...
51、设有查找表为(5,14,2,6,18,7,4,16,3),依次...
52、设有数据集合{40,29,7,73,101,4,55,2,81,92...
53、设有数据集合{50,39,17,83,111,14,65,13,91...
54、设有数据集合{50,39,17,83,111,14,65,13,91...
55、设有数据集合{50,39,17,83,111,14,65,13,91...
56、设有数据集合{50,39,17,83,91,14,65},依次...
57、设有数据集合{50，39，17，83，91，14，65}，此二叉排序树...
58、设有数据集合：{50,39,17,83,91,14,65)...
59、设有向图如图所示下,写出首先删除顶点 1 的 1...
60、设有序表为{2,5,11,12,30,48,58},元素的序...
61、设有序表为{21,22,23,24,25;26,27,28,29,30...
62、设有序表为{5,8,14,15,33,51,61,73,81,82,9...
63、设有一个整数序列{40,28,6,72,100,3,54}依...
64、设有一个整数序列{50,38,16,82,110,13,64},...
65、顺序查找算法如下，完成程序中空格部分。...
66、说明什么是顶点活动网(AOV 网)和拓扑序列...
67、写出如下图所示的二叉树的先序、中序和后序...
68、一组记录的关键字序列为(6,9,7,4,5,8),利用...
69、一组记录的关键字序列为{26,59,36,18,20,2}...
70、一组记录的关键字序列为{37,67,43,25,27,32...
71、一组记录的关键字序列为{37,70,47,29,31,85...
72、一组记录的关键字序列为{45,40,65,43,35,95...
73、一组记录的关键字序列为{45,40,65,43,35,95...
74、一组记录的关键字序列为{46,79,56,38,40,84...
75、一组记录的关键字序列为{47,80,57,39,41,46...
76、一组记录的关键字序列为{6,9,7,4,5,8},利用...
77、一组记录的关键字序列为{6,9,7,4,5,8}，利用...
78、一组记录的关键字序列为（42，37，62，40，32，92），利用快...
79、一组记录的关键字序列为（45，40，65，43，35，95），利用快...
80、一组记录的关键字序列为（47，80，57，39，41，46），利用堆...
81、已知某带权图的邻接矩阵如下所示：...
82、已知某二叉树的后序遍历序列是 febch.给出该...
83、已知某二叉树的先序遍历结果是：A,B,D,G,C,E,...
84、已知某二叉树的先序遍历序列是 aecdb,中序遍...
85、已知图 G 的邻接矩阵如下所示：从顶点 1 出发的广...
86、已知无向图 G 描述如下：
87、已知序列(10,18,4,3,6,12,1,9,15,8),请写出...
88、已知序列(17,18,60,40,7,32,73,65,85)请给...
89、已知序列(70,83,100,105,10,32,7,9),请写出...
90、已知一个无向图的邻接矩如下所示,写出从顶...
91、以 1,2,3 ,6,7,8 作为叶结点的权,构造一棵哈...
92、以 1,2,3,6,7,8 作为叶结点的权,构造一棵哈夫...
93、以 2,3,4,7,8,9 作为叶结点的权,构造一棵哈夫...
94、以 2,3,4,7,8,9 作为叶结点的权,构造一棵哈夫...
95、以 2，3，4，7，8，9 作为叶结点的权，构造一棵哈夫曼树。...
96、以 3,4,5,8,9,10 作为叶结点的权,构造一棵哈...
97、以 3,4,5,8,9,作为叶结点的权,构造一棵哈夫...
98、以 3，4，5，8，9，作为叶结点的权，构造一棵哈夫曼树。...
99、以 4,5,6,13,11,12 作为叶结点的权,构造一棵...
100、以给定权重值 1,2,12,13,20,25 为叶结点,建立...
101、由如图所示的二叉树，回答以下问题：...
102、有一棵树如图所示，回答下面问题：...

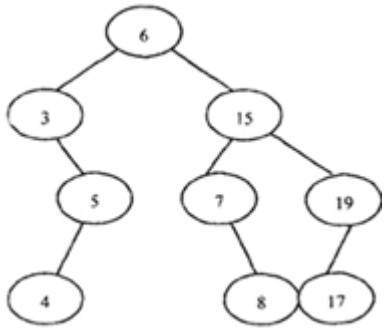
1、按折半查找对该表进行查找,求在等概率情况下查找成功的平均比较次数。

序号	1	2	3	4	5	6	7	8	9	10
序列	28	35	60	75	79	80	86	90	95	99

(l+2 * 2+3*4+4*3)/10=29/10

2、对给定的数列 b={6,15,3,7,19,8,5,17,4}

(1)依次取 b 中各数据，构造一棵二叉排序树。



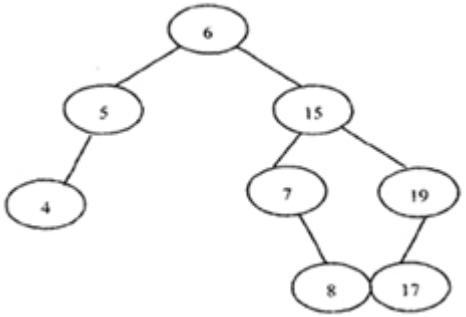
(2)给出按中序遍历该二叉排序树的序列。

3, 4, 5, 6, 7, 8, I5, 17, 19

(3)给出按后序遍历二叉排序树的序列。

4, 5, 3, 8, 7, 17, 19, 15, 6

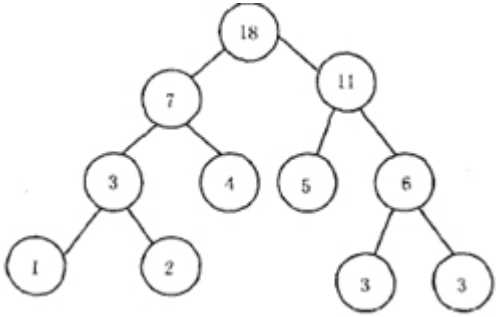
(4)画出在二叉树中删除结点 3 后的树结构。



3、对给定权值 2,1,3,3,4,5,

(1)对给定权值 2,1,3,3,4,5，构造哈夫曼树(要求每个结点的左子树

根结点的权小于等于右子树根结点的权)。



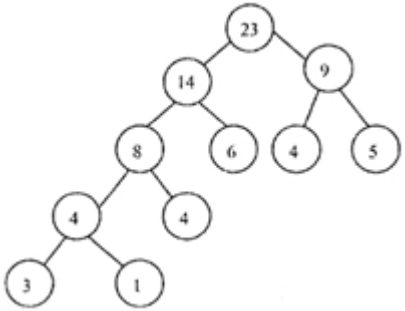
(2)给出各权值的哈夫曼编码。

2: 001
1: 000
3: 110
3: 111
4: 01

5: 10

4、**对给定权值 3,1,4,4,5,6,构造深度为 5 的哈夫曼树。**

(1)对给定权值 3,1,4,4,5,6，构造深度为 5 的哈夫曼树。(设根为第 1 层)



(2)求树的带权路径长度。

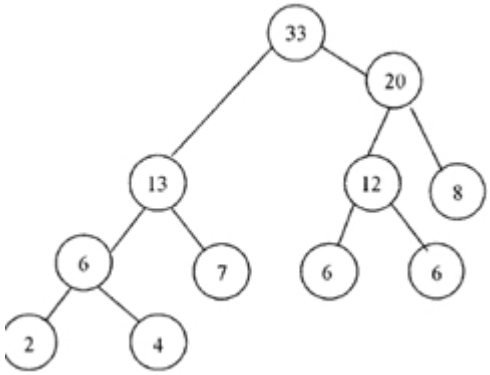
WPL=3*4+1*4+4 * 3+6*2+4*2+5*2=58

(3)链接存储上述哈夫曼树，结点中共有多少个指针域为空，说明理由。

共 11 个结点，22 个指针域，除根结点外，每个结点对应一个指针域，共 10 个指针域非空，故有 22-10=12 个空指针域，

5、**对给定权值 4,2,6,6,7,8,构造高度为 4 层的哈夫曼树。**

(1)对给定权值 4,2,6,6,7,8,构造高度为 4 层的哈夫曼树。(设根为第 1 层)提示：构造中当出现有两个以上值相等的可选结点时，可适当选择结点组合，以控制树的高度



(2)求树的带权路径长度。

WPL=(4+2+6+6) *3+(7+8) *2=84

6、**对关键字序列（36，69，46，28，30，74）采用快速排序，以第一个关键字为分割元素，经过一次划分后的结果序列为（）。**

（1）对关键字序列（36，69，46，28，30，74）采用快速排序，以第一个关键字为分割元素，经过一次划分后的结果序列为(D. 30，28，36，46，69，74)。

（2）用冒泡法对上述序列排序，经过两趟冒泡的结果序列为（A. 36，28，30，46，69，74）。

7、**对关键字序列（56，51，71，54，46，106），利用快速排序，以第一个关键字为分割元素，经过一次划分后结果为（）。**

（1）对关键字序列（56，51，71，54，46，106），利用快速排序，以第一个关键字为分割元素，经过一次划分后结果为（C. 46，51，54，56，71，106）。

（2）一组记录的关键字序列为（60，47，80，57，39，41，46，30），利用归并排序的方法，经过(2,2)归并的结果序列为（D. 47，57，60，80，30，39，41，46）。

8、**对于一个无向图，假定采用邻接矩阵表示，试分别写出从顶点 0 出发按深度优先搜索遍历得到的顶点序列。**

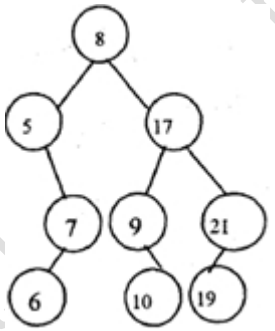
注：每一种序列都是唯一的，因为都是在存储结构上得到的。

	0	1	2	3	4	5	6
0	0	0	1	1	0	0	0
1	0	0	0	0	1	0	1
2	1	0	0	1	0	1	0
3	1	0	1	0	0	1	1
4	0	1	0	0	0	0	1
5	0	0	1	1	0	0	0
6	0	1	0	1	1	0	0

C. 0，2，3，5，6，1，4

9、**给定数列(8,17,5,9,21,10,7,19,6),依次取序列中的数构造丫棵二叉排序树**

(1)给定数列(8,17,5,9，21,10,7,19,6)，依次取序列中的数构造丫棵二叉排序树。

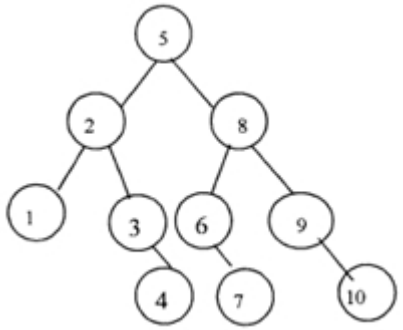


(2)对上述二叉树给出中序遍历得到的序列。

5,6,7,8,9,10,17,18,19,21

10、**画出对长度为 10 的有序表进行折半查找的判定树(以序号 1,2,……10 表示树结点)。**

(1)画出对长度为 10 的有序表进行折半查找的判定树(以序号 1，2，……10 表示树结点)。



(2)对上述序列进行折半查找，求等概率条件下，成功查找的平均查找长度。

ASL=(1x1+2x2+3x4+4x3)/10=29/10

11、**假设通信用的报文由 9 个字母 A、B、C、D、E、F、G、H 和 I 组成,它们出现的频率分别是：10、20、5、15、8、2、3、7 和 30。请请用这 9 个字母出现的频率作为权值求：**

（1）设计一棵哈夫曼树；

（2）计算其带权路径长度 WPL；

（3）写出每个字符的哈夫曼编码。

1）哈夫曼树如图 B-4 所示。

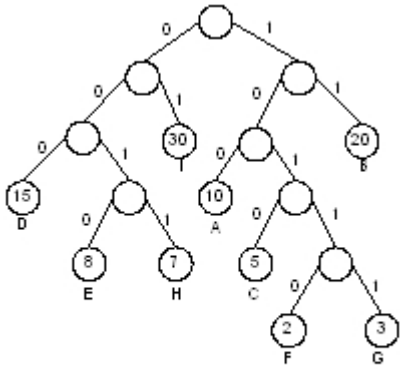
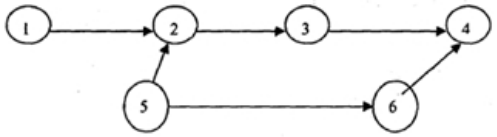


图 B-4

（2）其带权路径长度 WPL 值为 270。

（3）每个字符的哈夫曼编码为：A:100, B:11, C:1010, D:000, E:0010, F:10110, G:10111, H:0011, I:01

12、**简述拓扑排序的步骤。**



(1)简述拓扑排序的步骤。

循环执行以下两步：（1）选择一个度为 0 的顶点并翰出；（2）从网中删除此结点及所有出边

(2)说明有向图的拓扑序列不一定是唯一的原因。

因为选择一个度为。的顶点时不一定是唯一的

(3)如何利用拓扑排序算法判定图是否存在回路。

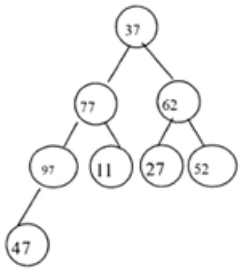
由顶点活动网构造拓扑序列的过程中，输出结点后，余下的结点均有前驱

(4)设有向图 G 如下，写出首先删除顶点 1 的 3 种拓扑序列。

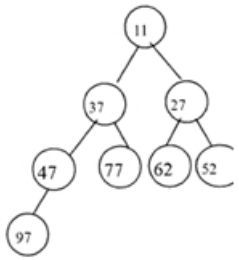
152364，152634，156234

13、**利用筛选法,把序列{37,77,62,97,11,27,62,47}建成堆(小根堆),画出相应的完全二叉树。**

(1)利用筛选法，把序列{37,77,62,97,11,27,62,47}建成堆(小根堆),画出相应的完全二叉树。



初始树



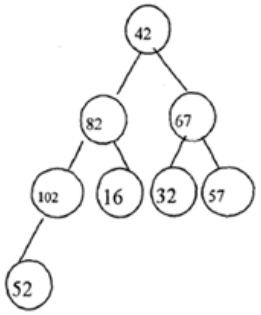
堆

(2)写出对上述堆所对应的二叉树进行前序遍历得到的序列 Q

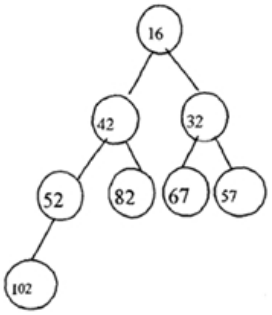
11,37,47,97,77,27,62,52

14、**利用筛选过程把序列{42,82,67,102,16,32,57,52}建成堆(小根堆),**

(1)利用筛选过程把序列{42,82,67,102,16,32,57,52}建成堆(小根堆),画出相应的完全二叉树(不要求中间过程)。



初始树



堆

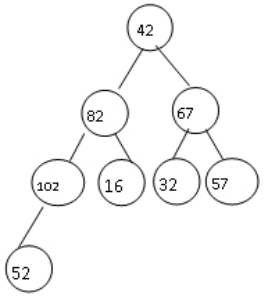
(2)写出对上述堆对应的完全二叉树进行中序遍历得到的序列。

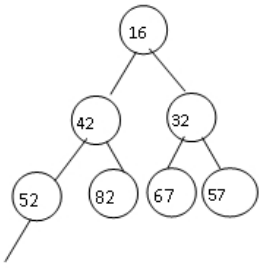
102,52,42,82,16,67,32,57

15、**利用筛选过程把序列{42,82,67,102,16,32,57,52}建成堆(小根堆),画出相应的完全二叉树(不要求中间过程)。**

（2）写出对上述堆对应的完全二叉树进行中序遍历得到的序列。

答：（1）如下图所示





初始树　　堆

（2）102，52，42，82，16，67，32，57

16、**请根据以下带权有向图 G**

（1）给出从结点 v1 出发分别按深度优先搜索遍历 G 和广度优先搜索遍历 G 所得的结点序列；

（2）给出 G 的一个拓扑序列；

（3）给出从结点 v1 到结点 v8 的最短路径。

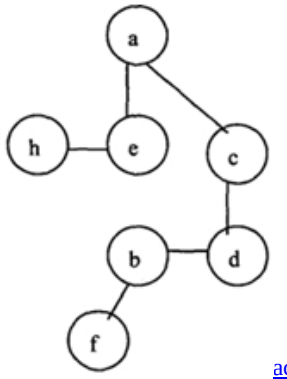
答：（1）深度优先遍历：v1，v2，v3，v8，v5，v7，v4，v6

广度优先遍历：v1，v2，v4，v6，v3，v5，v7，v8

（2）G 的拓扑序列为：v1，v2，v4，v6，v5，v5，v3，v5，v7，v8

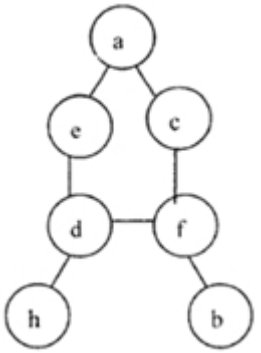
（3）最短路径为：v1，v2，v5，v7，v8

17、**如图所示,若从顶点 a 出发,首先经过 C 按图的深度优先搜索法进行遍历,给出可能得到的一种顶点序列。**



acdbfeh

18、**如图所示,若从顶点 a 出发,首先经过顶点 C.按广度优先搜索法进行遍历,给出可能得到的一种顶点序列。**



(1)如图所示，若从顶点 a 出发，首先经过顶点 C.按广度优先搜索法进行遍历，给出可能得到的一种顶点序列。acefdbh

(2)如图所示，给出从顶点 h 出发，首先经过顶点 d 和 e 按深度优先搜索法进行遍历，给出可能得到的一种顶点序列。hdeacfb

(3)一组记录的关键字序列为(80，57，41，39，46，47) ,利用堆排序的方法的方法建立小根堆(堆顶元素是最小元素)，给出按筛选法建立的的初始堆。39 46 41 57 80 47

19、**如下的一棵二叉棵树**

(1)请给出前序遍历序列？请给出中序遍历序列？

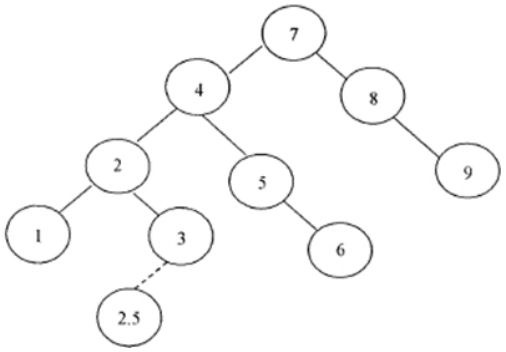
前序 Al A2 A4 A7 A8 A5 A9 A3 A6

中序 A7 A4 A8 A2 A5 A9 A1 A3 A6

(2)把 1,2,3,4,5,6,7,8,9 填人，使它成为一棵二叉排序树。提示：设图中的树是二叉排序树，则中序遍历序列是有序的，从而找出中序遍历序列与 1，2,...9 的对应关系。A7 A4 A8 A2 A5 A9 A1 A3 A6

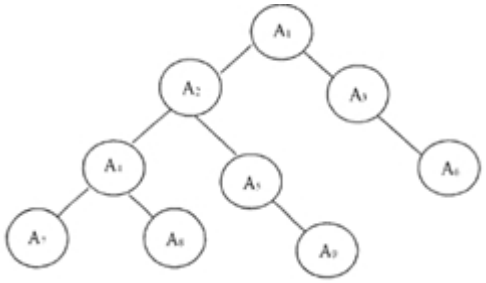
1 2 3 4 5 6 7 8 9

(3)在图中给出在二叉排序树中插人结点 2.5 的结果



20、**如下的一棵二叉棵树,**

(1) 请给出前序遍历序列？请给出中序遍历序列？



前序 A1 A2 A4 A7 AS A5 A9 A3 A6

中序 A7 A4 A8 A2 A5 A9 A1 A3 A6

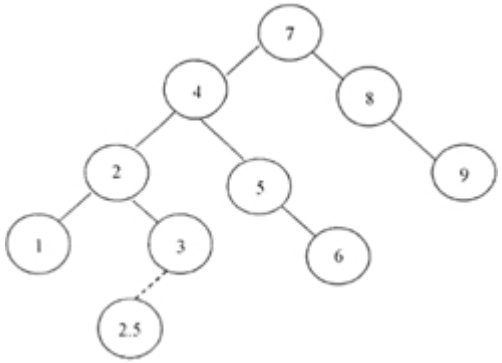
(2) 把 1,2,3,4,5,6,7,8,9 填入，使它成为一棵二叉排序树。

提示：设图中的树是二叉排序树，则中序遍历序列是有序的，从而找出中序遍历序列与 1，2， … 9 的对应关系。

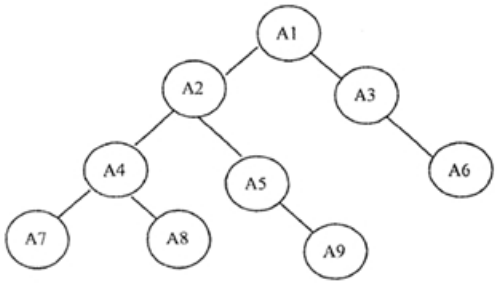
A7 A4 A8 A2 A5 A9 A1 A3 A6

1 2 3 4 5 6 7 8 9

(3) 在图中给出在二叉排序树中插入结点 2.5 的结果。



21、**如下的一棵树,给出先序遍历序列**

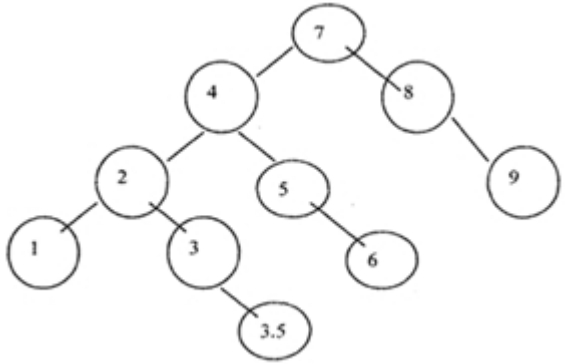


(1)如下的一棵树，给出先序遍历序列

A1 A2 A4 A7 A8 A5 A9 A3 A6

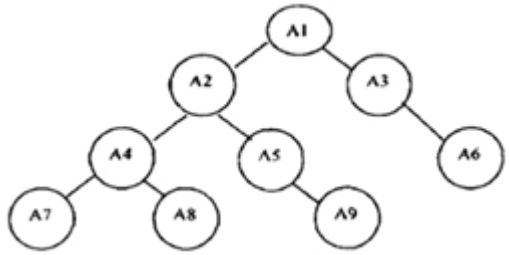
(2)把 1，2，3，4，5，6，7，8，9 填入，使它成为一棵二叉排序树（提示：设图中的树是二叉排序树，找出中序遍历序列与 1,2,……9 的对应关系）

(3)请在该树中再插人一个结点 3.5 作为叶结点，并使它仍然是一棵二叉排序树。



22、**如下是一棵二叉排序树,A1,A2,……A9 代表 1,2,3,……9 中各个不同数字,**

(1)给出对该树中序遍历的结果

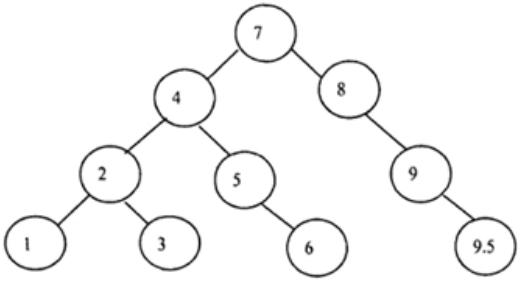


A7 A4 A8 A2 A5 A9 A1 A3 A6

1 2 3 4 5 6 7 8 9

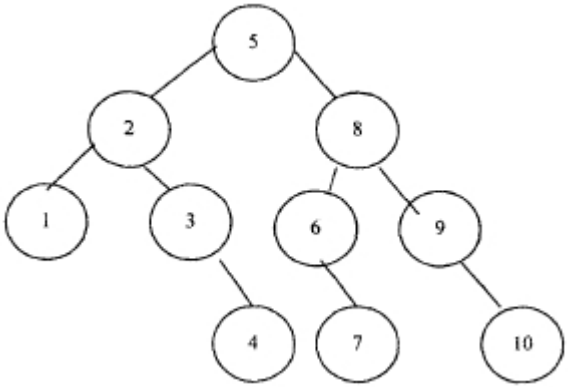
(2)A3,A5,A7 的值各为多少?851

(3)请在该树中再插入一个结点 9.5 作为叶结点，并使它仍然是一棵二叉排序树



23、**如下为一个长度为 10 的有序表,给出按折半查找对该表进行查找的判定树。**

(1)如下为一个长度为 10 的有序表，给出按折半查找对该表进行查找的判定树。

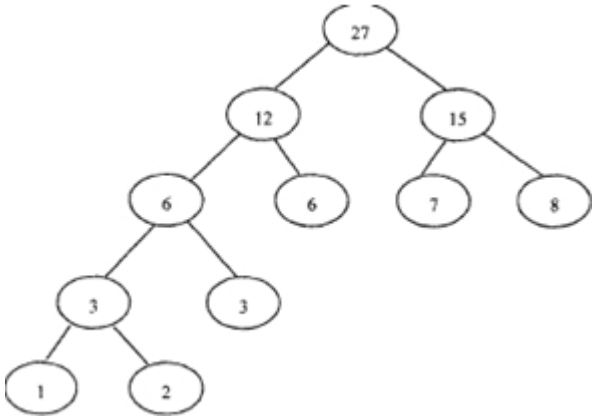


(2)按折半查找对该表进行查找，求在等概率情况下查找成功的平均比较次数。

序号	1	2	3	4	5	6	7	8	9	10
序列	28	35	60	75	79	80	86	90	95	99

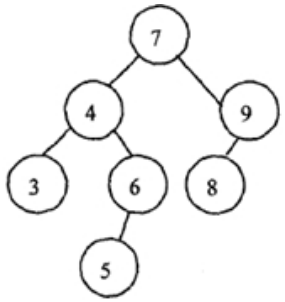
(1+2* 2+3* 4+4 3)/10=29/10

(3)以 1,2,3,6,7,8 作为叶结点的权，构造一棵哈夫曼树。



24、设 headI 和 PI 分别是不带头结点的单向链表 A 的头指针和尾指针，

(1)设 headI 和 PI 分别是不带头结点的单向链表 A 的头指针和尾指针，head2 和 P2 分别是不带头结点的单向链表 B 的头指针和尾指针，若要把 B 链表接到 A 链表之后，得到一个以 headI 为头指针的单向循环链表，写出其中两个关键的赋值语句(不用完整程序，结点的链域为 next)



p->next== head ip:->next= head;

(2)单向链表的链域为 next，设指针 p 指向单向链表中的某个结点，指针 S 指向一个要插入链表的新结点，现要把所指结点插入 p 所指结点之后，某学生采用以下语句：

p 一>next==;->next==p->next;

这样做正确吗？若正确则回答正确，若不正确则说明应如何改 o 写

不对，s->nextm p->next；p->next=s;

25、设查找表为(16,15,20,53,64,7)，用冒泡法对该表进行排序，在排序后的有序表的基础上进行折半查找，在等概率条件下,成功查找的平均查找长度为（）。

答案：D.14/6

26、设查找表为(20,19,24,57,68,11)

（1）用冒泡对该表进行排序，要求写出每一趟的排序过程，通常对 n 个元素进行冒泡排序要进行多少趟冒泡？第 j 趟要进行多少次元素间的比较？

（2）在排序后的有序表的基础上，画出对其进行折半查找所对应的判定树。（要求以数据元素作为树结点）

（3）求在等概率条件下，对上述有序表成功查找的平均查找长度。

答：（1）原序列 16 15 20 53 64 7

15 16 20 53 7 64 n-1 趟

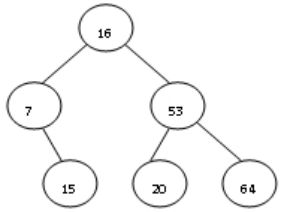
15 16 20 7 53 64 n-j 次

15 16 7 20 53 64

15 7 16 20 53 64

7 15 16 20 53 64

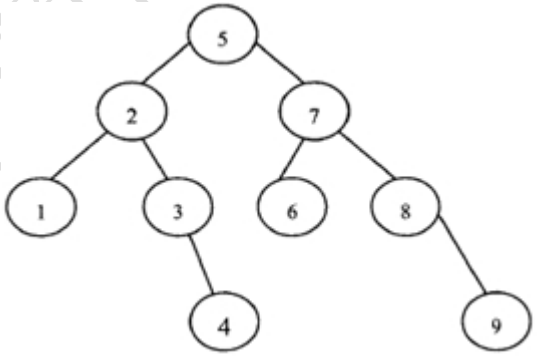
(2)



(3)平均查找长度=（1*1+2*2+3*3）/6=14/6

27、设查找表为{1,10,11,14,23,27,29,55,68},元素的下标依次为 1,2,3,……,9。

(1)画出对上述查找表进行折半查找所对应的判定树(树中结点用下标表示)。



(2)说明成功查找到元素 14，需要依次经过与哪些元素的比较？共几次比较？

按序号 5,2,3,4。按元素 23,10,11,14

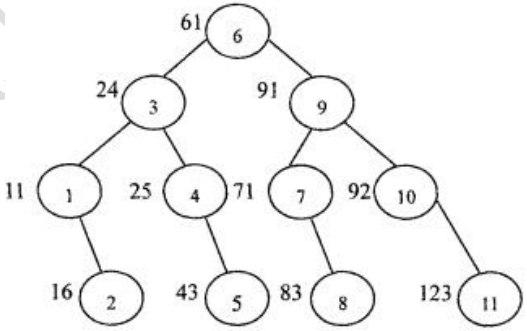
(3)求在等概率条件下，成功查找的平均比较次数？

ASL=(1+2*2+3*4+4*2)/9=25/9

28、设查找表为{1,2,3,4,5,6,7,8,9,10,11}

序号	1	2	3	4	5	6	7	8	9	10	11
序列	11	16	24	25	43	61	71	83	91	92	123

（1）画出对上述查找表进行折半查找所对应的判定树（树中结点用下标表示）。

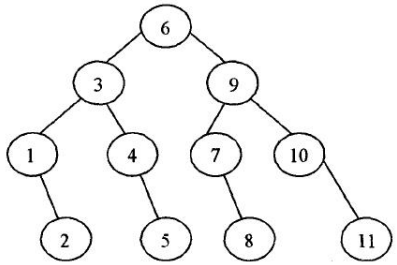


（2）说明不成功查找元素 45 需要经过多少次比较？4 次

（3）求在等概率条件下，成功查找的平均比较次数？-->平均查找长度=（1+2*2+3*4+4*4）/11=3

29、设查找表为{1,2,3,4,5,6,7,8,9,10,11}

(1)画出对上述查找表进行折半查找所对应的判定树(树中结点用数值表示)。

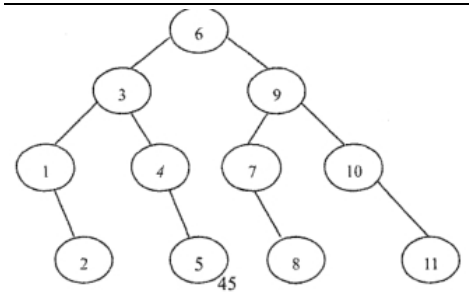


(2)说明成功查找到元素 5，9 各需要经过多少次元素间的比较？(4 次；2 次)

(3)说明查找不到元素 4.2，5.5 各需要经过多少次元素间的比较？(3 次；4 次)

30、设查找表为{10,18,24,28,45,58,68,80,88,89,120},元素的下标依次为 1,2,3,……,11。

(1)画出对上述查找表进行折半查找所对应的判定树(树中结点用下标表示)。



(2)说明要成功查找到元素 45,依次与哪些元素进行了比较(给出查找路径)。

元素序号 6,3,4,5.元素为(58,24,28,45)。

(3)求在等概率条件下，成功查找的平均比较次数？

ASL=(1+2*2+3*4+4*4)/11=3

31、**设查找表为{16,15,20,53,64,7},**

(1)用冒泡法对该表进行排序(要求升序排列)，写出每一趟的排序过程，通常对 n 个元素进行冒泡排序要进行多少趟冒泡?第 j 趟要进行多少次元素间的比较?

原序列16 15 20 53 64 7

15 16 20 53 7 64

15 16 20 7 53 64

15 16 7 20 53 64

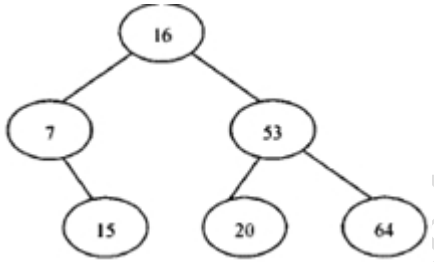
15 7 16 20 53 64

7 15 16 20 53 64

n－1 趟

n－j 次

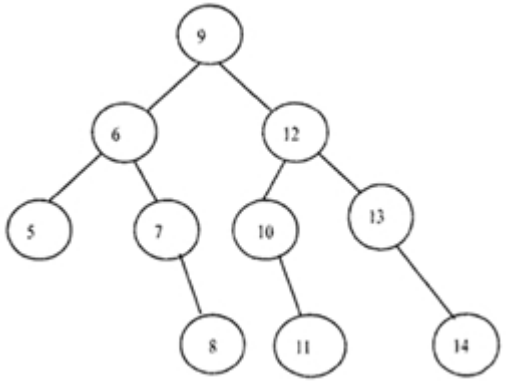
(2)在排序后的有序表的基础上，画出对其进行折半查找所对应的判定树。(要求以数据元素作为树结点)。



（3）平均查找长度=（1*1+2*2+3*3）/6=14/6

32、**设查找表为{5,6,7,8,9,10,11,12,13,14}**

(1)画出对上述有序表进行折半查找所对应的判定树(要求以数据元素作为树结点)



(2)给出二叉排序树的定义，针对_L 述折半查找所对应的判定树的构造过程，说明判定树是否是二叉排序树(设树中没有相同结点)?

答：二叉排序树或者是一棵空树，或者是一棵具有下列性质的二叉树：若它的左子树非空，则左子树的所有结点的值都小于它的根结点的值;若它的右子树非空，则右子树的所有结 1036 点的值都大于(若允许结点有相同的值，则大于等于)它的根结点的值;左，右子树也是一棵二叉排序树，按定义判定树是二叉排序树。

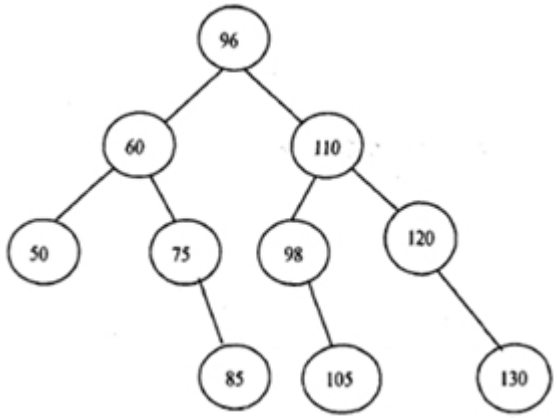
(3)为了查找元素 5.5，经过多少次元素间的比较才能确定不能查到?-->)3 次

33、**设查找表为{50,60,75,85,96,98,105,110,120,130}**

(1)说出进行折半查找成功查找到元素 120 需要进行多少次元素间的比较?3 次

(2)为了折半查找元素 95，经过多少次元素间的比较才能确定不能查到?4 次

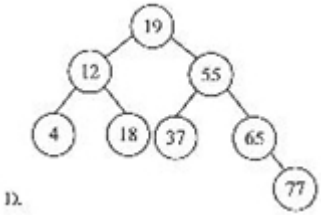
(3)画出对上述有序表进行折半查找所对应的判定树(要求以数据元素作为树结点)。



34、**设查找表为：**

序号	1	2	3	4	5	6	7	8
序列	4	12	18	19	37	55	65	77

(1)画出对上述查找表进行折半查找所对应的判定树是(D)。

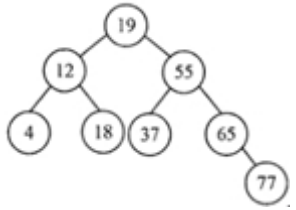


(2)用折半查找在该查找表成功查找到元素 55 需要经过(B.2)次比较。

35、**设查找表为：**

序号	1	2	3	4	5	6	7	8
序列	4	12	18	19	37	55	65	77

D.



答：D

36、**设查找表为：用折半查找在该查找表成功查找到元素 55 需要经过（）次比较。**

序号	1	2	3	4	5	6	7	8
序列	4	12	18	19	37	55	65	77

答案：B.2

37、**设关键字序列为：(36,69,46,28,30,74)**

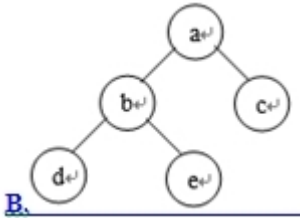
(1)将此序列用快速排序的方法，以第一个记录为基准得到的一趟划分的结果为**(D.30，28，36，46，69，74)**。

(2)用冒泡法对上述序列排序，经过两趟冒泡的结果序列为**(A.36，28，30，46，69，74)**。

38、**设关键字序列为：(36,69,46,28,30,74),将此序列用快速排序的方法,以第一个记录为基准得到的一趟划分的结果为（）。**

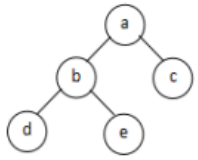
30,28,36,46,69,74

39、**设关键字序列为：（36，69，46，28，30，74），将此序列用快速排序的方法，以第一个记录为基准得到的一趟划分的结果为（ ）。**
答案：D.30，28，36，46，69，74
40、**设关键字序列为：（36，69，46，28，30，74），用冒泡法对上述序列排序，经过两趟冒泡的结果序列为（ ）。**
答案：A.36，28，30，46，69，74
41、**设某二叉树先序遍历为 abdec,中序遍历为 dbeac**
(1)该二叉树的图形是()。

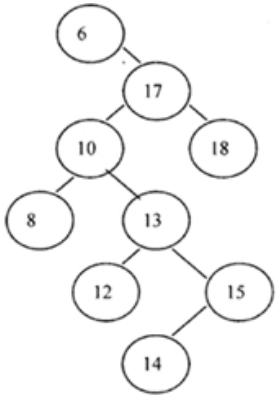


B.

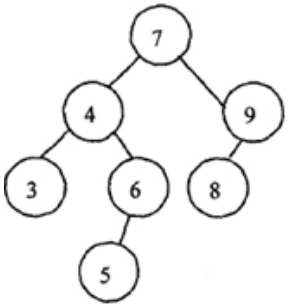
(2)写出该二叉树后序遍历的结果(B.debca)。
42、**设某二叉树先序遍历为 abdec，中序遍历为 dbeac。该二叉树的图形是（ ）。**



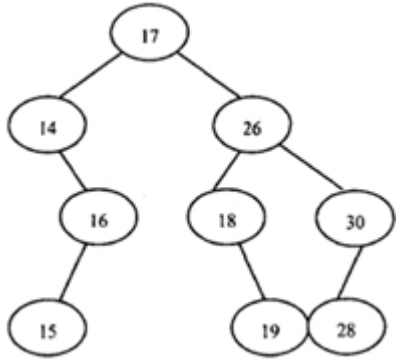
答案：B.【图片】
43、**设某二叉树先序遍历为 abdec，中序遍历为 dbeac。写出该二叉树后序遍历的结果（ ）。**
答案：B.debca
44、**设数据集合 a={6,17,10,13,8,15,12,18,14}**
(1)依次取 a 中各数据，构造一棵二叉排序树。



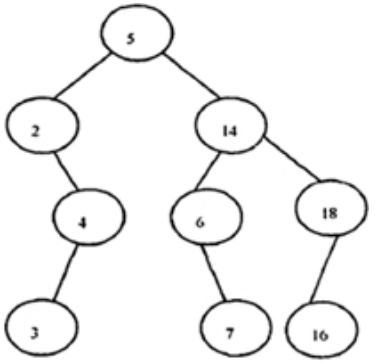
(2)给出对该二叉树中序遍历的序列。
中序遍历 6,8,10,12,13,14,15,17,18
(3)对该二叉树进行查找，成功查找到 14 要进行多少次元素间的比较?6 次
45、**设数据集合 a={7,4,9,8,6,5,3},依次取 a 中各数据,构造一棵二叉排序树。**
(1)设数据集合 a={7，4，9，8，6，5，3}，依次取 a 中各数据，构造一棵二叉排序树。



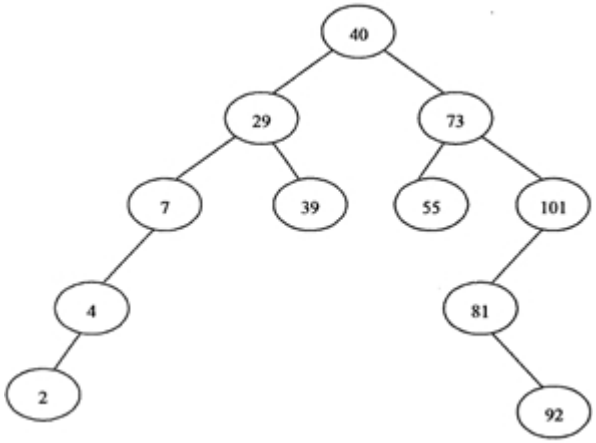
(2)对该二叉树进行查找，成功查找到 5 要进行多少次元素间的比较?-->4
(3)给出对上述二叉排序树进行中序遍历的序列-->3,4,,5,6,7,8,9
46、**设数据序列为：{53,30,37,12,45,24,96)**
(1)从空二叉树开始逐个插入该数据序列来形成二叉排序树，若希望高度最小，应该选择的序列是(B．或 37,24,12, 30,53,45,96)。
(2)用链接地址法将该数据序列构造哈希表，哈希函数为 H(key)=keymod13，则散列地址为 1 的链中有(B．或 1)个记录。
47、**设数据序列为：{53,30,37,12,45,24,96}，从空二叉树开始逐个插入该数据序列来形成二叉排序树，若希望高度最小，应该选择的序列是（ ）。**
答案：B.37,24,12,30,53,45,96
48、**设一组记录的关键字序列为（49，83，59，41，43，47），采用堆排序算法建立初始小根堆，在该小根堆上逐次取走堆顶元素后，经调整得到的 4 个元素的堆为（ ）。**
答案：A.47，49，59，83
49、**设有查找表{17,26,14,16,15,30,18,19,28},依次取表中数据构造一棵二叉排序树。**
(1)设有查找表{17,26,14,16,15,30,18,19,28}，依次取表中数据构造一棵二叉排序树。



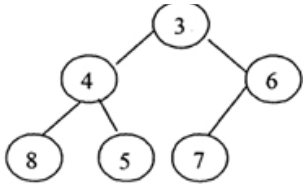
(2)对上述二叉树给出后序遍历的结果-->15,16,14,19,18,28,30,26,17
(3)对上述二叉树给出中后序遍历的结果-->14,15,16,17,18,19,26,28,30
(4)在上述二叉树中查找元素 15 共要进行多少次元素的比较？-->4
50、**设有查找表{5,14,2,6,18,7,4,16,3},依次取表中数据,构造一棵二叉排序树。**
(1)设有查找表{5,14,2,6,18,7,4,16,3}，依次取表中数据，构造一棵二叉排序树。



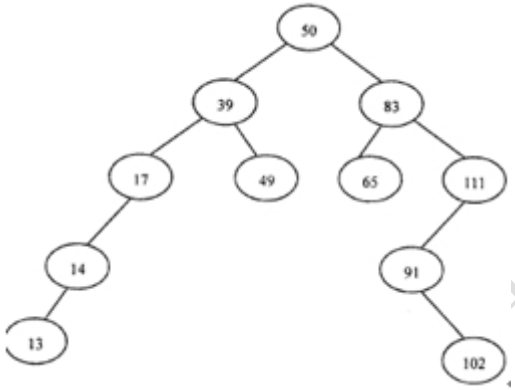
(2)说明如何通过序列的二叉排序树得到相应序列的排序结果。-->中序遍历
51、**设有查找表为(5,14,2,6,18,7,4,16,3),依次取表中数据，构造一棵二叉排序树，对该二叉树进行后序遍历的结果序列为（ ）。**
答案：A.3,4,2,7,6,16,18,14,5
52、**设有数据集合{40,29,7,73,101,4,55,2,81,92,39},依次取集合中各数据构造一棵二叉排序树。**
(1)设有数据集合{40,29,7,73,101,4,55,2,81,92,39},依次取集合中各数据构造一棵二叉排序树。



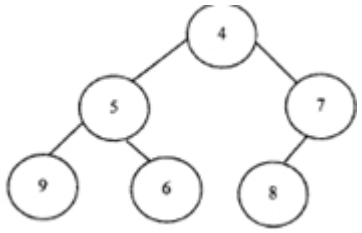
(2)一组记录的关键字序列为(5,8,6,3,4,7),利用堆排序(堆顶元素是最小元素)的方法建立初始堆。(要求用完全二叉树表示)。
-->3,4,6,8,5,7



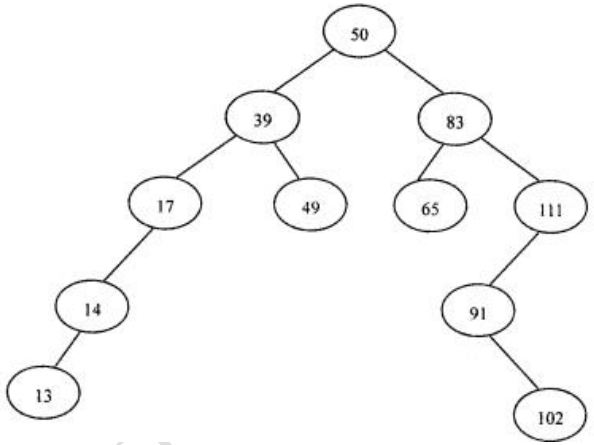
53、**设有数据集合{50,39,17,83,111,14,65,13,91,102,49},依次取集合中各数据构造一棵二叉排序树。**
(1)设有数据集合{50,39,17,83,111,14,65,13,91,102,49}，依次取集合中各数据构造一棵二叉排序树。



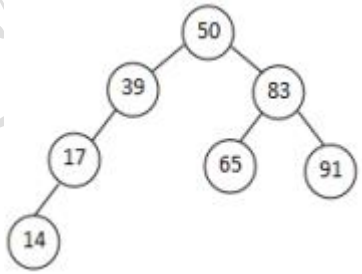
(2)一组记录的关键字序列为(6,9,7,4,5,8)，利用堆排序(堆顶元素是最小元素)的方法建立初始堆。(要求用完全二叉树表示)



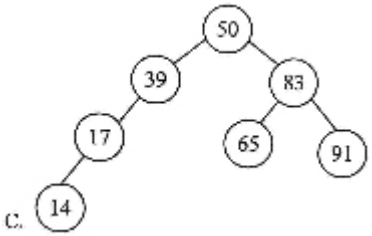
4,5,7,9,6,8
54、**设有数据集合{50,39,17,83,111,14,65,13,91,102,49},依次取集合中各数据构造一棵二叉排序树。**



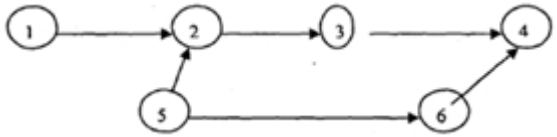
55、**设有数据集合{50,39,17,83,111,14,65,13,91,102,49},依次取集合中各数据构造一棵二叉排序树。**
50831765111149113102
56、**设有数据集合{50,39,17,83,91,14,65},依次取集合中各数据构造一棵二叉排序树,是如下的（）。**



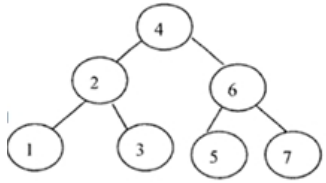
57、**设有数据集合{50，39，17，83，91，14，65}，此二叉排序树的（ ）遍历是有序序列。**
答案：B.中序
58、**设有数据集合：{50,39,17,83,91,14,65)**
(1)依次取集合中各数据构造一棵二叉排序树是下图中的()。



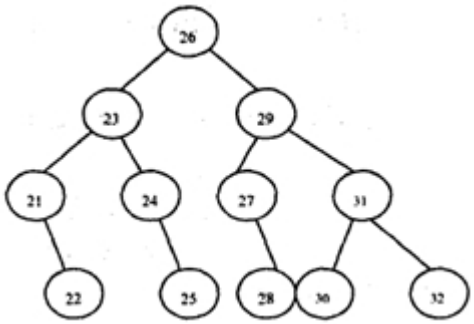
(2)二叉排序树的()遍历是有序序列。**B.或中序**
59、**设有向图如图所示下,写出首先删除顶点 1 的 1 种拓扑序列。**



152364 或 152634 或 156234
60、**设有序表为{2,5,11,12,30,48,58},元素的序号依次为 1,2,3…,7**
(1)画出对上述查找表进行折半查找所对应的判定树(树中结点用序号表示)。

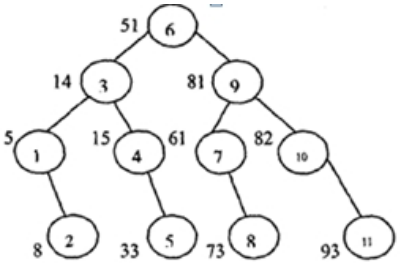


(2)说明成功查找到元素 11 需要经过多少次比较？3 次
(3)在等概率条件下，给出成功查找的平均查找长度？
(1+2*2+3*4)/7=17/7
61、**设有序表为{21,22,23,24,25;26,27,28,29,30,31,32},元素的下标从 0 开始。**
(1)说出有哪几个元素需要经过 4 次元素间的比较才能成功查到。
(2)画出对上述有序表进行折半查找所对应的判定树(树结点用数值表示)



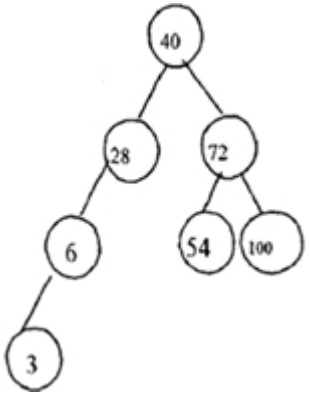
(3)设查找元素为 5,需要进行多少次元素间的比较才能确定不能查到。-->3 次
(4) 求在等概率条件下，成功查找的平均比较次数?ASL=(1+2+2+3*4+5*4)/12=37/12

62、**设有序表为{5,8,14,15,33,51,61,73,81,82,93},元素的序号依次为1,2,3……•,11.**
(1)画出对上述查找表进行折半查找所对应的判定树(树中结点可用序号表示)



(2)说明成功查找到元素 33 需要经过多少次比较？4 次
(3) 在等概率条件下，给出成功查找的平均查找长度？（1+2*2+3*4+4*4）/11=33/11=3

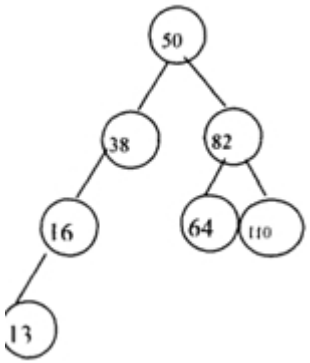
63、**设有一个整数序列{40,28,6,72,100,3,54}依次取出序列中的数,构造一棵二叉排序树。**
(1)设有一个整数序列{40,28,6,72,100,3,54}依次取出序列中的数，构造一棵二叉排序树。



(2)对上述二叉排序树，在等概率条件下，求成功查找的平均查找长度。

ASL=(1x1+2x2+3x3+4)/7=18/7

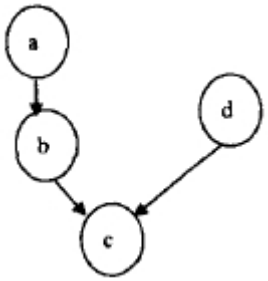
64、**设有一个整数序列{50,38,16,82,110,13,64},依次取出序列中的数,构造一棵二叉排序树。**
(1)设有一个整数序列{50,38,16,82,110,13,64}，依次取出序列中的数，构造一棵二叉排序树。



(2)利用上述二叉排序树，为了查找 110，经多少次元素 i 司的比较能成功查到，为了查找 15，经多少次元素间的比较可知道查找失败?-->三次，四次

65、**顺序查找算法如下，完成程序中空格部分。**
B.或 i++;

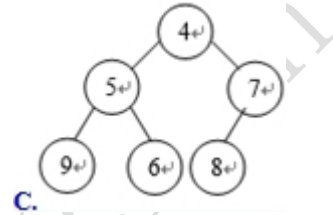
66、**说明什么是顶点活动网(AOV 网)和拓扑序列**



(1)说明什么是顶点活动网(AOV 网)和拓扑序列
答：用顶点表示活动，边表示活动间先后关系的有向图称为顶点活动网
在顶点活动网中，若不存在回路，则所有活动可排列成一个线性序列，使每个活动的所有前驱活动都排在该活动的前面，称此序列为拓扑序列
(2)设有向图 G 如下，写出 3 种拓扑序列，（abdc adbc dabc）
(3)在图 G 中增加一条边，使图 G 仅有一条拓扑序列-->在 b 和 d 间添加有向边

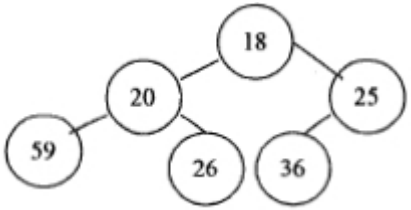
67、**写出如下图所示的二叉树的先序、中序和后序遍历序列。**
答：二叉树的定义是递归的，所以，一棵二叉树可看作由根结点，左子树和右子树这三个基本部分组成，即依次遍历整个二叉树，又左子树或者右子树又可看作一棵二叉树并继续分为根结点、左子树和右子树三个部分……，这样划分一直进行到树叶结点。
（1）先序为“根左右”，先序序列为：fdbacegihl
（2）中序为“左根右”，中序序列为：abcdefghij
（3）后序为“左右根”，后序序列为：acbedhjigf

68、**一组记录的关键字序列为(6,9,7,4,5,8),利用堆排序(堆顶元素是最小元素)的方法建立初始堆是如下哪个图？**



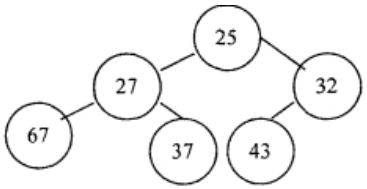
C.

69、**一组记录的关键字序列为{26,59,36,18,20,2},给出利用堆排序(堆顶元素是最小元素)**
(1)一组记录的关键字序列为{26,59,36,18,20,2}，给出利用堆排序(堆顶元素是最小元素)的方法建立的初始堆(要求以完全二叉树描述)。



18,20,25,59,26,36
(2) 对关键字序列(26,59 ,36 ,18 ,20 ,64}采用快速排序，给出以第一个关键字为分割元素，经过一次划分后的结果。
20,18,26,36,59,64

70、**一组记录的关键字序列为{37,67,43,25,27,32},给出利用堆排序(堆顶元素是最小元素)的方法建立的初始堆(要求以完全二叉树描述)。**
(1)一组记录的关键字序列为{37,67,43,25,27,32}，给出利用堆排序(堆顶元素是最小元素)的方法建立的初始堆(要求以完全二叉树描述)。
25,27,32,67,37,43



(2)对关键字序列{40,73,49,31,33,77}采用快速排序，给出以第一个关键字为分割元素，经过一次划分后的结
33,31,40,49,73,77

71、**一组记录的关键字序列为{37,70,47,29,31,85}，利用快速排序，以第一个关键字为分割元素，经过一次划分后结果为（）。**
答案：B.31，29，37，47，70，85

72、**一组记录的关键字序列为{45,40,65,43,35,95},写出利用快速排序的方法,以第一个记录为基准得到的一趟划分的结果(要求给出一趟划分中每次扫描和交换的结果)。**

(1)一组记录的关键字序列为{45,40,65,43,35,95}，写出利用快速排序的方法，以第一个记录为基准得到的一趟划分的结果(要求给出一趟划分中每次扫描和交换的结果)。

[45] 40 65 43 35 95

35 40 65 43 [35] 95

35 40 [65] 43 65 95

35 40 43 [43] 65 95

35 40 43 [45] 65 95

(2)对序列{45,40,65,43,35,95}利用直接插人排序，写出逐次插入过程(从第一个元素一直到第六个元素)。

40 45 65 43 35 95

40 43 45 65 35 95

35 40 43 45 65 95

73、**一组记录的关键字序列为{45,40,65,43,35,95}写出利用快速排序的方法,**

(1)一组记录的关键字序列为(45,40,65,43,35,95)写出利用快速排序的方法，以第一个记录为基准得到的一趟划分的结果(要求给出一趟划分中每次扫描和交换的结果)。

45 40 65 43 [35] 95

35 40 65 43 35 95

35 40 [65] 43 65 95

35 40 43 [43] 65 95

35 40 43 45 65 95

(2)同样对序列(45,40,65,43,35,95)利用直接插入排序，写出逐次插人过程'(从第一个元素一直到第六个元素)。

40 45 65 43 35 95

40 43 45 65 35 95

35 40 43 45 65 95

74、**一组记录的关键字序列为{46,79,56,38,40,84}**

(1)利用快速排序的方法，给出以第一个记录为基准得到的一次划分结果(给出逐次交换元素的过程，要求以升序排列)。

(1)初始序列

[46],79,56,38,40,84

40,79,56,38,[40],84

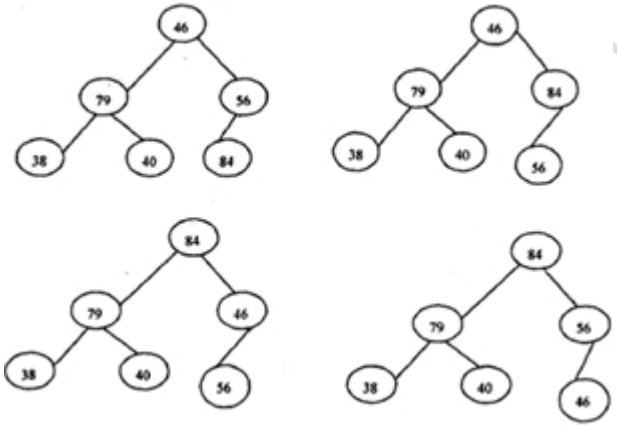
40,[79],56,38,79,84

40,38,56,[38],79,84

40,38,[56],56,79,84

40,38,[46],56,79,84

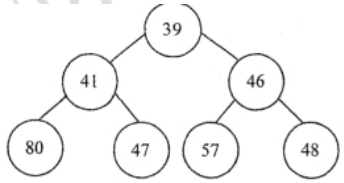
(2)对上述序列用堆排序的方法建立大根堆，要求以二叉树逐次描述建堆过程。



75、**一组记录的关键字序列为{47,80,57,39,41,46,48},**

(1)一组记录的关键字序列为{47,80,57,39,41,46,48},给出利用堆排序(堆顶元素是最小元素)的方法建立的初始堆(要求以完全二叉树描述)。

39,41,46,80,47,57,48

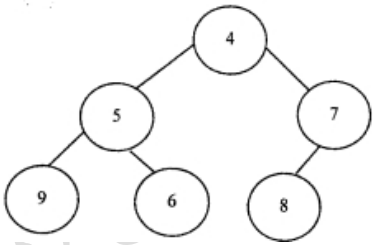


(2)对关键字序列(46,79,56,38,40,84,88,90)采用快速排序，给出以第一个关键字为分割元素，经过一次划分后的结果。

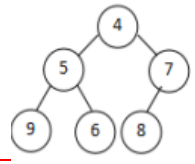
40,38,46,56,79,84,88,90

76、**一组记录的关键字序列为{6,9,7,4,5,8},利用堆排序(堆顶元素是最小元素)的方法建立初始堆。(要求用完全二叉树表示)**

4 , 5 , 7 9 , 6 , 8



77、**一组记录的关键字序列为{6,9,7,4,5,8}，利用堆排序（堆顶元素是最小元素）的方法建立初始堆是如下哪个图？（）**



答案：C.【图片】

78、**一组记录的关键字序列为（42，37，62，40，32，92），利用快速排序算法，以第一个关键字为分割元素，经过一次划分后结果为（）。**

（1）一组记录的关键字序列为（42，37，62，40，32，92），利用快速排序算法，以第一个关键字为分割元素，经过一次划分后结果为（B. 32，37，40，42，62，92）。

（2）利用筛选过程把序列（42，82，67，102，16，32，57，52）建成初始堆（小根堆）为（C. 16，42，32，52，82，67，57，102）。

79、**一组记录的关键字序列为（45，40，65，43，35，95），利用快速排序的方法，以第一个记录为基准得到的一趟划分的结果为（）。**

（1）一组记录的关键字序列为（45，40，65，43，35，95），利用快速排序的方法，以第一个记录为基准得到的一趟划分的结果为（C. 35，40，43，45，65，95）。

（2）对上述序列利用直接插入排序，逐次插入过程中，共进行了（D. 10）次元素间的比较。

80、**一组记录的关键字序列为（47，80，57，39，41，46），利用堆排序的方法建立的初始堆为（）（堆顶元素是最小元素,采用树的形式建堆）。**

（1）一组记录的关键字序列为（47，80，57，39，41，46），利用堆排序的方法建立的初始堆为（B. 39，41，46，80，47，57）（堆顶元素是最小元素,采用树的形式建堆）。

（2）输出堆顶元素后,调整后的堆为（A. 41，47，46，80，57）。

81、**已知某带权图的邻接矩阵如下所示：**

$$\begin{Bmatrix} 0 & 6 & 1 & 5 & \infty & \infty \\ 6 & 0 & 5 & \infty & 3 & \infty \\ 1 & 5 & 0 & 5 & 6 & 4 \\ 5 & \infty & 5 & 0 & \infty & 2 \\ \infty & 3 & 6 & \infty & 0 & 6 \\ \infty & \infty & 4 & 2 & 6 & 0 \end{Bmatrix}$$

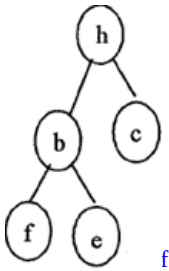
从顶点 1 出发的广度优先搜索序列为()。**A.或 1,2,3,4,5,6**

82、已知某二叉树的后序遍历序列是 febch.给出该二叉树的根结点？

(1)已知某二叉树的后序遍历序列是 febch.给出该二叉树的根结点？又该二叉树的中序遍历序列是 fbehc，分别给出该二叉树的左、右子树的结点？

根 h　左子树 b，f，e　　右子树 c

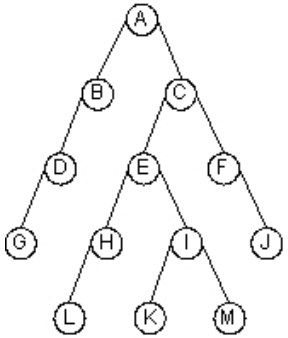
(2)画出上述二叉树？若上述二叉树的各个结点的字特分别代表不同的整数(其中没有相等的)，并恰好使该树成为一棵二叉排序树，试给出 h、b、c、f、e 的大小关系。



f<b<e<h<c

83、已知某二叉树的先序遍历结果是：A,B,D,G,C,E,H,L,I,K,M,F 和 J,它的中序遍历结果是：G,D,B,A,L,H,E,K,I,M,C,F 和 J,请画出这棵二叉树,并写出该二叉树后续遍历的结果。

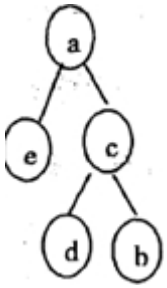
（1）二叉树图形表示如下：



（2）该二叉树后序遍历的结果是：G、D、B、L、H、K、M、I、E、J、F、C 和 A。

84、已知某二叉树的先序遍历序列是 aecdb,中序遍历序列是 eadcb,试画出该二叉树。

(1)已知某二叉树的先序遍历序列是 aecdb，中序遍历序列是 eadcb，试画出该二叉树。



(2)给出上述二叉树的后序遍历序列。-->edbca

(3)若上述二叉树的各个结点的字符分别是 1,2,3,4,5,并恰好使该树成为一棵二叉排序树，试问 a,b,C.d,e 的值各为多少？

-->e=1,a=2,d=3,c=4,b=5

85、已知图 G 的邻接矩阵如下所示：从顶点 1 出发的广度优先搜索序列为（）。

$$\begin{pmatrix} \infty & 6 & 1 & 5 & \infty & \infty \\ 6 & \infty & 5 & \infty & 3 & \infty \\ 1 & 5 & \infty & 5 & 6 & 4 \\ 5 & \infty & 5 & \infty & \infty & 2 \\ \infty & 3 & 6 & \infty & \infty & 6 \\ \infty & \infty & 4 & 2 & 6 & \infty \end{pmatrix}$$

A. 1；2，3，4；5；6

86、已知无向图 G 描述如下：

G=（V，E）

V={V1，V2，V3，V4，V5}

E={（V1，V2），（V1，V4），（V2，V4），（V3，V4），（V2，V5），（V3，V4），（V3，V5）}

（1）画出 G 的图示；

（2）然后给出 G 的邻接矩阵和邻接表；

（3）写出每个顶点的度。

答：①g1 的图示和图 g1 的邻接表如下图所示。

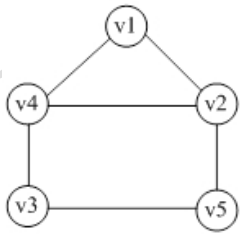


图 G

②图 G 的邻接矩阵如下图所示：

图 G 的邻接矩阵

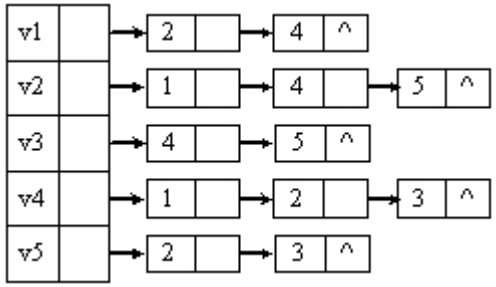


图 G 的邻接表

③V1、V2、V3、V4、V5 的度分别为：2，3，2，3，2

87、已知序列(10,18,4,3,6,12,1,9,15,8),请写出对此序列采用归并排序法进行升序排序时各趟的结果。

答：原始序列：10，18，4，3，6，12，1，9，15，8

第 1 趟：　[10，18][3，4][6，12][1，9][8，15]

第 2 趟：　[3，4，10，18，][1，6，9，12][8，15]

第 3 趟：　[3，4，10，18，][1，6，8，9，12，15]

第 4 趟：　[1，3，4，6，8，9，10，12，15，18]

88、已知序列(17,18,60,40,7,32,73,65,85)请给出采用冒泡排序法对该序列作升序排列时的每一趟结果。

答：原始序列：256，301，751，129，937，863，742，694，076，438

第 1 趟：　256，301，129，751，863，742，694，076，438，937

第 2 趟：　256，129，301，751，742，694，076，438，863，937

第 3 趟：　129，256，301，742，694，076，438，751，863，937

第 4 趟：　129，256，301，694，076，438，742，751，863，937

第 5 趟：　129，256，301，076，438，694，742，751，863，937

第 6 趟：　129，256，076，301，438，694，742，751，863，937

第 7 趟：　129，076，256，301，438，694，742，751，863，937

第 8 趟：　076，129，256，301，438，694，742，751，863，937

第 9 趟：　076，129，256，301，438，694，742，751，863，937

89、已知序列(70,83,100,105,10,32,7,9),请写出对此序列采用插入排序法进行升序排序时各趟的结果。

答：原始序列：（70），83，100，65，10，32，7，9

第 1 趟：　（70，83），100，65，10，32，7，9

第 2 趟：　（70，83，100），65，10，32，7，9

第 3 趟：　（65，70，83，100），10，32，7，9

第 4 趟：　（10，65，70，83，100），32，7，9

第 5 趟：　（10，32，65，70，83，100），7，9

第 6 趟：　（7，10，32，65，70，83，100），9

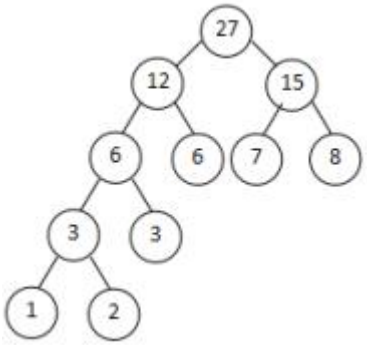
第 7 趟：　（7，9，10，32，65，70，83，100）

90、已知一个无向图的邻接矩如下所示,写出从顶点 0 出发按深度优先搜索遍历得到的顶点序列。

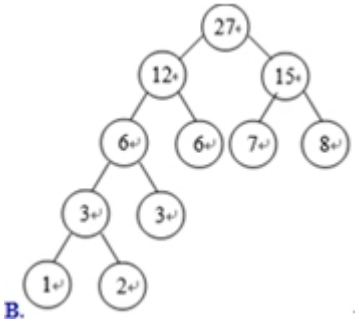
	0	1	2	3	4	5	6
0	0	0	1	1	0	0	0
1	0	0	0	0	1	0	1
2	1	0	0	1	0	1	0
3	1	0	1	0	0	1	1
4	0	1	0	0	0	0	1
5	0	0	1	1	0	0	0
6	0	1	0	1	1	0	0

答案：0,2,3,5,6,1,4

91、**以 1,2,3 ,6,7,8 作为叶结点的权,构造一棵哈夫曼树是如下哪个图？（）。**

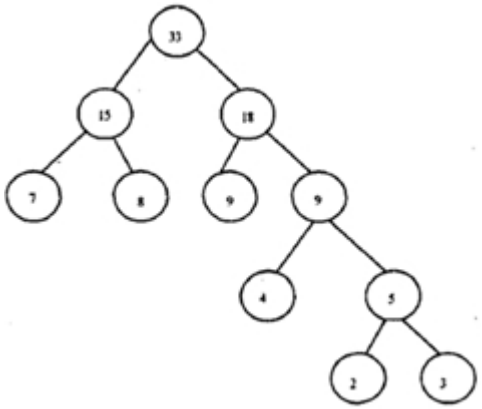


92、**以 1,2,3,6,7,8 作为叶结点的权,构造一棵哈夫曼树是如下哪个图？**



93、**以 2,3,4,7,8,9 作为叶结点的权,构造一棵哈夫曼树(要求每个结点的左子树根结点的权**

(1)以 2,3,4,7,8,9 作为叶结点的权，构造一棵哈夫曼树(要求每个结点的左子树根结点的权小于等于右子树根结点的权)，给出相应权重值叶结点的哈夫曼编码。



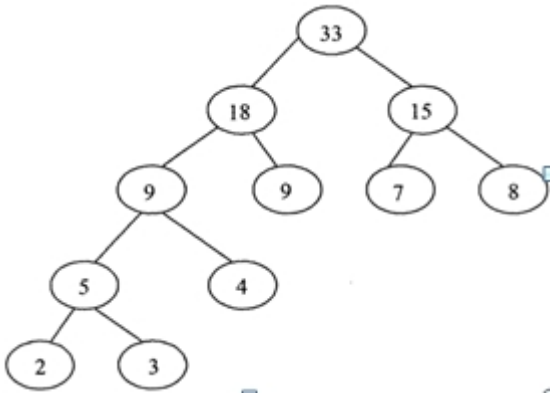
2: 1110
3: 1111
4: 110
7: 00
8: 01

9: 10

(2)一棵哈夫曼树有 n 个叶结点，它一共有多少个结点?简述理由。
-->2n-1 个，因为非叶结点数比叶结点数少一个。

94、**以 2,3,4,7,8,9 作为叶结点的权,构造一棵哈夫曼树。**

(1)以 2,3,4,7,8,9 作为叶结点的权，构造一棵哈夫曼树。



(2)给出上述哈夫曼树叶结点的哈夫曼编码。

2:0000
3 0001
4 001
7 10
8 11

9 01

(3)一组记录的关键字序列为(37,70,47,29,31,85)，利用快速排序，以第一个关键字为分割元素，给出经过一次划分后结果。(从小到大排序)。

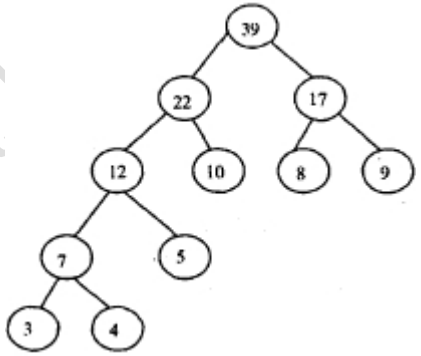
31,29,37,47,70,85

95、**以 2，3，4，7，8，9 作为叶结点的权，构造一棵哈夫曼树。权重值为 4 的叶结点的哈夫曼编码为（）**

B. 001

96、**以 3,4,5,8,9,10 作为叶结点的权,构造一棵哈夫曼树。**

(1)以 3,4,5,8,9,10 作为叶结点的权，构造一棵哈夫曼树。



(2)给出相应权重值叶结点的哈夫曼编码。

3 0000
4 0001
5 001
10 01
8 10

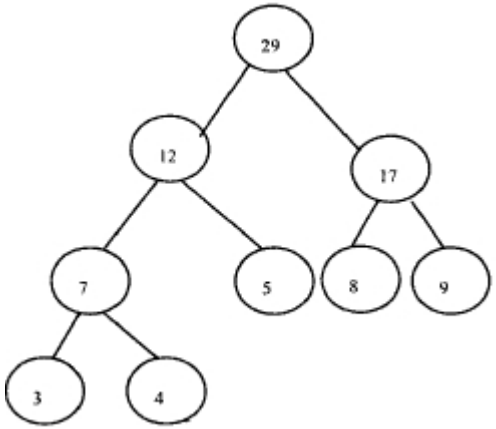
9 11

(3)一棵哈夫曼树有 2n-1 个结点，它是共有多少个权童值构造而成的？简述理由？

n 个，因为非叶结点数比叶结点数少一个，而权值个数=叶结点数

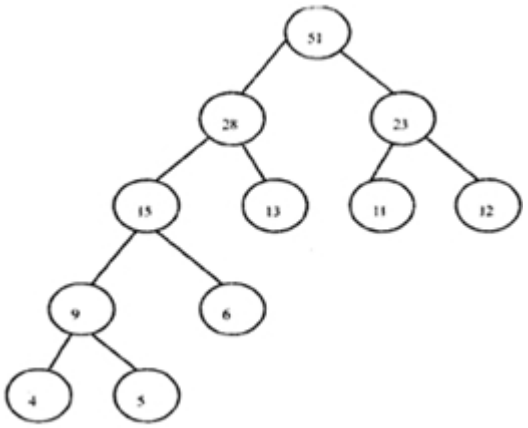
97、**以 3,4,5,8,9,作为叶结点的权,构造一棵哈夫曼树。**

(1)以 3,4,5,8,9，作为叶结点的权，构造一棵哈夫曼树。



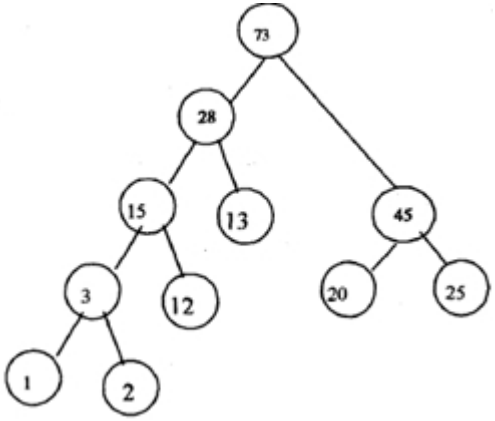
(2)给出相应权重值叶结点的哈夫曼编码。
3：000
4：001
5：01
8：10

9：11
(3)n 个叶结点的哈夫曼树，总共有多少个结点？-->2n-1
98、**以 3，4，5，8，9，作为叶结点的权，构造一棵哈夫曼树。该树的带权路径长度为（）**
D.65
99、**以 4,5,6,13,11,12 作为叶结点的权,构造一棵哈夫曼树。**
(1)以 4，5，6，13，11，12 作为叶结点的权，构造一棵哈夫曼树。

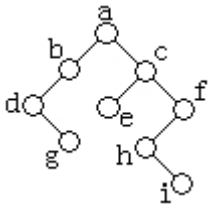


(2)给出相应权重值叶结点的晗夫曼编码。
4 0000
5 0001
6 001
13 01
11 10

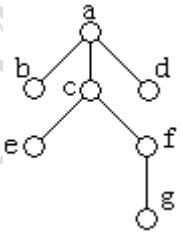
12 11
(3)一棵哈夫曼树有 n 个叶结点，该树共有多少个结点？简述理由？
2n-1 个，因为非叶结点数比叶结点数少一个，非叶结点数为 n-1，所以共有 2n-1 个。
(4)给出对上述哈夫曼树中序遍历的序列。
4，9，5，15，6，28，13，51，11，23，12
100、**以给定权重值 1,2,12,13,20,25 为叶结点,建立一棵哈夫曼树。**
(1)以给定权重值 1,2,12,13,20,25 为叶结点，建立一棵哈夫曼树。



(2)若哈夫曼树有 n 个非叶子结点，则树中共有多少结点。对给定的一组权重值建立的棵哈夫曼树是否一定唯一。
2n-1　不一定唯一
101、**由如图所示的二叉树，回答以下问题：**



（1）其中序遍历序列（B. gbaechif）；
（2）其前序遍历序列（C. abdgcefhi）；
（3）其后序遍历序列（A. gdbeihfca）；
102、**有一棵树如图所示，回答下面问题：**



（1）这棵树的根结点是（C. a）；
（2）这棵树的叶子结点是（E.b、e、d、g）；
（3）这棵树的度是（A. 3）；
（4）这棵树的深度是（B. 4）；
（5）c 结点的孩子结点是（D. e、f）；
（6）c 结点的父母结点是（C. a）。

程序选择填空题(59)--

1、假设队列顺序存储结构为：
2、假设队列顺序存储结构为：
3、假设链表结点存储结构为：
4、设 SeqStack 为顺序栈,写出下列程序段执行后...
5、设线性表为(6,10,16,4),以下程序用说明结构...
6、设线性表以不带头结点的单向链表存储,链表...
7、设线性表以不带头结点的单向链表存储,链表...
8、设有一个不带头结点的单向链表,头指针为 hea...
9、设有一个不带头结点的单向链表,头指针为 hea...
10、设有一个递归算法如下：intfact(intn)...
11、设有一个头指针为 head 的不带头结点单向链表...
12、设有一个头指针为 head 的不带头结点单向链表...
13、设有一个头指针为 head 的不带头结点单向链表...
14、设有一个头指针为 head 的单向链表中（结点类型...
15、顺序查找算法如下,完成程序中空格部分。...
16、下列是用头插法建立带头结点的且有 n 个结点...
17、下列是用尾插法建立带头结点的且有 n 个结点...
18、下列是在具有头结点单向链表中删除第 i 个结...
19、下列是在具有头结点单向列表中在第 i 个结点...
20、下面是实现对一个含 10 个整数从小到大冒泡排...
21、写出下列程序执行后的结果。SeqQueue Q;...
22、写出下列程序执行后的结果。SeqStack S;...
23、以下程序段的结果是：c 的值为（）。char *a[5]={...
24、以下程序段的结果是：c 的值为（）。char a[5]=”1...
25、以下程序是后序遍历二叉树的递归算法的程序...
26、以下程序是快速排序的算法，设待排序的记录序...
27、以下程序是快速排序的算法，完成程序中空格部...
28、以下程序是先序遍历二叉树的递归算法的程序...
29、以下程序是先序遍历二叉树的递归算法的程序...
30、以下程序是折半插入排序的算法，设待排序的记...
31、以下程序是中序遍历二叉树的递归算法的程序...
32、以下程序是中序遍历二叉树的递归算法的程序...
33、以下函数为链队列的入队操作,x 为要入队的结...
34、以下函数为直接选择排序算法，对 a[1],a[2],……
35、以下函数在 a[0]到 a[n-1]中,用折半查找算法...
36、以下函数在 a[0]到 a[n-1]中,用折半查找算法...
37、以下函数在 a[0]到 a[n-1]中，用折半查找算法查...
38、以下函数在 head 为头指针的具有头结点的单向...
39、以下利用直接插入排序算法对存放在 a[0],a[1...
40、以下冒泡法程序对存放在 a[1]，a[2]，……，a[n]中...
41、以下是采用选择排序对 10 个数按照从小到大排...
42、以下是冒泡排序算法对存放在 a[1],a[2],...,...
43、以下是用尾插法建立带头结点且有 n 个结点的...
44、以下是中序遍历二叉树的递归算法的程序,完...
45、以下为求二叉树深度的算法,完成程序中空格...
46、以下直接插入排序算法对存放在 a[0],a[1],•...
47、在下面空格处填写适当的语句，以使下面的循环...
48、在下面空格处填写一条语句,以使下面的串连...

49、在下面空格处填写一条语句,以使下面的进栈...
50、在下面空格处填写一条语句,以使下面的链式...
51、在下面空格处填写一条语句,以使下面的循环...
52、在下面空格处填写一条语句,以使下面的循环...
53、在下面空格处填写一条语句，以使下面的出栈算...
54、在下面空格处填写一条语句，以使下面的串比较...
55、在下面空格处填写一条语句，以使下面的串复制...
56、在下面空格处填写一条语句，以使下面的串连接...
57、在下面空格处填写一条语句，以使下面的进栈算...
58、在下面空格处填写一条语句，以使下面的顺序队...
59、在下面空格处填写一条语句，以使下面的循环队...

1、假设队列顺序存储结构为：

```
struct SeqQueue {
   ElemType data[MaxSize];
   int front,rear;
};
struct SeqQueue *sq;
```

请补充下面入队算法（不考虑空间循环使用）。

```
void InQueue(struct SeqQueue * sq,ElemtType x){
        if(   1   ){
                printf(“队列已满\n”);
                exit(1);
        }
        sq->data[sq->rear]=x;
           2
}
```

其中，1 和 2 处应该补充的代码是（）

答案：B.sq->rear==Maxsize,sq->rear++;

2、假设队列顺序存储结构为：

```
struct SeqQueue {
   ElemType data[MaxSize];
   int front,rear;
};
struct SeqQueue *sq;
```

请补充下面入队算法（数组空间循环使用）。

```
void InQueue(struct SeqQueue * sq,ElemtType x){
        if(   1   ){
                printf(“队列已满\n”);
                exit(1);
        }
        sq->data[sq->rear]=x;
           2
}
```

其中，1 和 2 处应该补充的代码是（）

答案：B.(sq->rear+1)%MaxSize==sq->front,sq->rear=(sq->rear+1)%MaxSize;

3、假设链表结点存储结构为：

```
 struct node {
   int data ;
   struct   node *next;
};
struct node *front,*next;
```

InitQueue()、InQueue()、OutQueue()、QueueEmpty()分别是链队的初始化、入队、出队、判空操作。

下面程序执行后，运行结果是（）。

```
int i;
InitQueue();
for(i=0;i<6;i++) InQueue(i++);
while(!QueueEmpty())
        printf(“%d”,OutQueue());
printf(“\n”);
```

答案：B.1 2 3 4 5 6

4、设 SeqStack 为顺序栈,写出下列程序段执行后的结果。

```
SeqStack S;
InitStack(S);
 Push(S，3);
 Push(S，4);
 Push(S，5);
 int x=Pop(S)+2*Pop(S);
 Push(S，x);
 int i，a[4]={5，8，12，15};
 for (i=0;i<4;i++) Push(S，a[i]);
 while(!StackEmpty(S)) Printf(“%d “，Pop(S));
```

执行后的输出结果为：A.15 12 8 5 13 3。

5、设线性表为(6,10,16,4),以下程序用说明结构变量的方法建立单向链表,并输出链表中各结点中的数据。#defineNULL0

```
#defineNULL0
voidmain()
{NODEa,b，c，d，*head，*p;
a.data=6;
b.data=10;
c.data=16;
d.data=4;/*d 是尾结点*/
head=(1)&a;
a.next=&b;
b.next=&c;
c.next=&d;
(2)d->next=NULL;/*以上结束建表过程*/
p=head;/*p 为工作指针，准备输出链表*/
do
{printf("%d\n",(3)p->data);
(4)p=p->next);
}while((5)p!=NULL);
}
```

6、设线性表以不带头结点的单向链表存储,链表头指针为 head,以下程序的功能是：

(1)输出链表中各结点中的数据域 data。

(2)把该单向链表改为以 p 作为尾指针的单向循环链表。

(链表中结点的指针域为 next，数据域为 data)。

```
        # define NULL0
        void main()
        { NODE * head ,*p;
        P=head; /*p 为工作指针*/
        do
        {printf("%d \n",(1) p->data);
        (2) p=p-->next;
        }while(p ->next ! =(3) NULL
        ); printf("%d\n" p->data);
        ((4) p->nex=head
        }
```

7、设线性表以不带头结点的单向链表存储,链表头指针为 head。以下程序的功能是输出链表中各结点中的数据域 data,完成程序中空格部分。

```
       # define NULL 0
       Void main()
       { NODE * head，* p ;
          p=head; / *p 为工作指针* /
          do
          { printf(“%d\n”,p->data);
          ①C.或 p=p->next;
          }while(_②B.或 p!=NULL);
       }
```

8、设有一个不带头结点的单向链表,头指针为 head,p、prep 是指向结点类型的指针,该链表在输入信息时不慎把相邻两个结点的信息重复输入,以下程序段是在该单向链表中查找这相邻两个结点,把该结点的数据域 data 打印出来,并把其中之一从链表中删除,填写程序中的空格。

```
 prep=head;
 p=prep->next;
 while（p->data!=prep->data）
 { prep=p;
   ①p=p->next;
 }
 printf（“%d”， p->data）;
 prep->next=②p->next;
```

9、设有一个不带头结点的单向链表,头指针为 head,p、prep 是指向结点类型的指针,要求表中各结点的数据域各不相同,但该表在输入信息时不慎把相邻两个结点的信息重复输入,

以下程序段是在该单向链表中查找这数据域相同的相邻两个结点，把该结点的数据域 data 打印出来，并把其中之一从链表中删除，填写程序中的空格。

程序片段如下：

```
        prep=head; p=prep->next;
        while(p-> data! =prep- >data)
        {
        prep=p;
(1) p=p->next
        }
```

printf("min=%d", (2) p->data prep- >data);
prep->next= (3) p->next
10、**设有一个递归算法如下：intfact(intn)**
intfact(intn)
{//n 大于等于 0
if(n<=0)return1;
elsereturnn*fact(n-1);
}
执行程序结果输出：**D.n+1**
11、**设有一个头指针为 head 的不带头结点单向链表,且 p、q 是指向链表中结点类型的指针变量,p 指向链表中某结点 a(设链表中没有结点的数据域与结点 a 的数据域相同),在以下程序段中,写出相关语句**
(1) 使该单向链表成为单向循环链表
(2) 删去 a 结点
q=p; x=p->data;
while (q->next! =NULL)q=q->next;
(1)q->next=head;
q=p; p=p->next;
while(p- >data! = x)
{ q=p;
(2)p= p->next;
}
(3)q- >next= p- >next;
12、**设有一个头指针为 head 的不带头结点单向链表中(结点类型为 NODE),p 为指向该链表中某个结点的指针。以下程序段为插入一个指针为 s 的结点,使它成为 p 结点的直接前驱,请选择其中空格的选项。**
NODE *q;
q=head;
while(q->next!=p)
①**C.s=s->next**;
s->next=p;
②**C.q->next=s**;
13、**设有一个头指针为 head 的不带头结点单向链表中（结点类型为 NODE），p 为指向该链表中某个结点的指针。以下程序段为插入一个指针为 s 的结点，使它成为 p 结点的直接前驱，请把合适选项填写到空行处。**
NODE *q;
q=head;
while(q->next!=p)
q=q->next;
s->next=p;
_____;
答案：C.q->next=s
14、**设有一个头指针为 head 的单向链表中（结点类型为 NODE），p 为指向该链表中某个结点的指针。以下程序段为插入一个指针为 s 的结点，使它成为 p 结点的直接前驱，请选择其中空格的选项。**
NODE *q;
q=head;
while(q->next!=p)
_____;
s->next=p;
q->next=s;
答案：B.q=q->next

15、**顺序查找算法如下,完成程序中空格部分。**
int search (NODE a[ ]， int n， int k)
/* 在 a[0]，a[1]…a[n-1]中查找关键字等于 k 的记录，查找成功返回记录的下标，失败时返回 -1*/
{ int i=0;
while (i< n && a[i].key!=k)
①**B.i++;**
if (②**B.a[i].key= =k**)
return i;
else return -1;
}
16、**下列是用头插法建立带头结点的且有 n 个结点的单向链表的算法,请在空格内填上适当的语句。**
NODE *create2(n)
/*对线性表(n,n-1,.....,1),建立带头结点的线性链表 */
{
NODE *head,*p,*q;
int i;
p=(NODE *)malloc(sizeof(NODE));
head=p;
p->next=NULL;
a=p;
for(i=1;i<=n;i++)
{
p=(NODE *)malloc(sizeof(NODE));
p->data=i;
if(i==1)
p->next=NULL;
else
p->next=q->next;
q->next=p;
}
return(head);
}
17、**下列是用尾插法建立带头结点的且有 n 个结点的单向链表的算法,请在空格内填上适当的语句。**
NODE *create1(n)
/* 对线性表(1,2,.....,n),建立带头结点的单向链表 */
{
NODE *head,*p,*q;
int i;
p=(NODE *)malloc(sizeof(NODE));
head=p; q=p; p->next=NULL;
for(i=1;i<=n;i++)
{
p=(NODE *)malloc(sizeof(NODE));
p->data=i;
p->next=NULL;
q->next=p;
a=p;
}
return(head);
}
18、**下列是在具有头结点单向链表中删除第 i 个结点的算法,请在空格内填上适当的语句。**
int delete(NODE *head,int i)
{
NODE *p,*q;
int j;
q=head;
j=0;
while((q!=NULL)&&(j<i-1)) /*找到要删除结点的直接前驱，并使 q 指向它*/
{
q=q->next;
j++;
}
if(q==NULL)
return(0);
p=q->next;
q->next=p->next;
free(p);
return(1);
}
19、**下列是在具有头结点单向列表中在第 i 个结点之前插入新结点的算法,请在空格内填上适当的语句。**
int insert(NODE *head,int x,int i)
{
NODE *q,*p;
int j;
q=head;
j=0;
while((q!=NULL)&&(j<i-1))
{ q=q->next;j++; }
if(q==NULL) return(0);
p= (NODE) malloc (sizeof (NODE));
p->data=x;
p->next=q->next;
return(1);
}
20、**下面是实现对一个含 10 个整数从小到大冒泡排序的代码,(1)处应该填写（）。**
for(j=1;j<=9;j++)
for(i=1;i<=10-j;i++)
if((1))
{t=a[i];a[i]=a[i+1];a[i+1]=t;}

执行后的输出结果为：**A.a[i]>a[i+1]**

21、**写出下列程序执行后的结果。SeqQueue Q;**

```
SeqQueue Q;
InitQueue(Q);
int a[4]={5,8,12,15};
for(int i=0;i<4;i++) InQueue(Q,a[i]);
InQueue(Q,OutQueue(Q));
InQueue(Q,30);
InQueue(Q,OutQueue(Q)+10);
while(!QueueEmpty(Q)) printf("%d ",OutQueue(Q));
```

执行后的输出结果为：B. 12 15 5 30 18。

22、**写出下列程序执行后的结果。SeqStack S;**

```
SeqStack S;
InitStack(S);
Push(S,3);
Push(S,4);
Push(S,5);
int x=Pop(S)+2*Pop(S);
Push(S,x);
int i,a[4]={5,8,12,15};
for (i=0;i<4;i++) Push(S,a[i]);
while(!StackEmpty(S)) Printf("%d ",Pop(S));
```

执行后的输出结果为：A. 15 12 8 5 13 3。

23、**以下程序段的结果是：c 的值为（）。char *a[5]={"12378","1237","1236789","1237","123708"}**

```
char *a[5]={"12378","1237","1236789","1237","123708"}
int i,c=0
for(i=0;i<5;i++)
if (strcmp(a[i],"1237")==0) c++;
```

答案选：A.2

24、**以下程序段的结果是：c 的值为（）。char a[5]="1236789";**

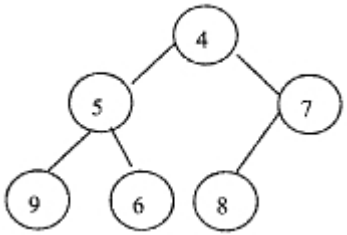
```
char a[5]="1236789";
int *p=a,c=0;
While (*p++) c++;
```

答案选：B.7

25、**以下程序是后序遍历二叉树的递归算法的程序,完成程序中空格部分(树结构中左、右指针域分别为 left 和 right,数据域 data 为字符型,BT 指向根结点)。完成程序中空格部分。**

```
void postorder (struct BTreeNode * BT)
{
if((1) BT! =NULL ){
postorder(BT->left);
(2) postorder(BT->right);
(3) printf("%c", BT->data);
}
```

利用上述程序对下图所示二叉树遍历的结果为(4) 9,6,5,8,7,4



26、**以下程序是快速排序的算法，设待排序的记录序列存放在 a[start],…a[end]中，按记录的关键字进行快速排序，先进行一次划分，再分别进行递归调用。void quicksort (NODE a[ ], int start ,int end)**

```
void quicksort ( NODE a[ ], int start ,int end )
{  int  i，j;
   NODE mid ;
   if (start>=end )
     return;
   i=start;
   j=end;
   mid=a[i];
   while (i<j)
   {  while(i<j && a[j]．key>miD.key)
        j- -;
      if(i<j)
       {  a[i]=a[j];
        (1) C. i++;
       }
      while(i<j && a[i]．key<=miD.key)
        (2) C. i++;
      if(i<j)
       { (3) A. a[j]=a[i];
        (4) D. j--;
       }
   }
   a[i]=mid;
   quicksort (a,stat, i-1);
   quicksort (5) B. (a, i+1,end);
}
```

27、**以下程序是快速排序的算法，完成程序中空格部分。 设待排序的记录序列存放在 a[start],…a[end]中，按记录的关键字进行快速排序，先进行一次划分，再分别进行递归调用。**

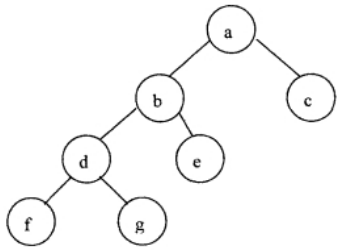
```
void quicksort(NODE a[],int start,int end)
 {   int   i,j;
     NODE mid;
     if (start>=end )
        return;
     i=start;
     j=end;
     mid=a[i];
     while (i<j)
     {   while(i<j && a[j].key>mid.key)
           j--;
         if(i<j)
           {   a[i]=a[j];
             i++;
           }
         while(i<j && a[i].key<=mid.key)
           i++;
         if(i<j)
           {   a[j]=a[i];
             j--;
           }
     }
     a[i]=mid;
     _____________________
     quicksort(a,i+1,end);
   }
```

答案：A.quicksort(a,start,i-1);

28、**以下程序是先序遍历二叉树的递归算法的程序,完成程序中空格部分(树结构中左、右指针域分别为 left 和 right,数据域 data 为字符型,BT 指向根结点).**

```
void Preorder(struct BTreeNode *BT)
if(BT! =NULL){
(1) (1) printf("%c", BT->data);
(2) (2) Preorder(BT->left);
Preorder(BT->right);}
}
```

利用上述程序对下图进行遍历，结果是(3)a b d f g e c



29、**以下程序是先序遍历二叉树的递归算法的程序,完成程序中空格部分(树结构中左、右指针域分别为 left 和 right,数据域 data 为字符型,BT 指向根结点)。**

```
void Preorder(struct BTreeNode *BT)
{if(BT！ =NULL)
  {printf("%c"，BT->data);
  Preorder(BT->left);
  Preorder(BT—>right);
  }
}
```

30、**以下程序是折半插入排序的算法，设待排序的记录序列存放在 a[1],…a[n]中，以 a[0]作为辅助工作单元，程序是要把 a[i] 插**

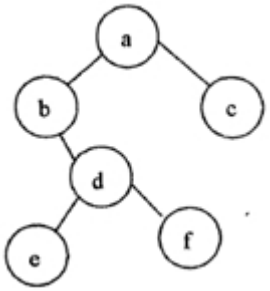
入到已经有序的序列 a[1],…a[i-1]中。 void binsort (NODE a[ ],int n)

```
void binsort (NODE a[ ],int n)
{   int x,i,j,s,k,m;
   for (i=2；i<=(1) D. n;i++)
   {   a[0]=a[i];
       x= a[i]．key;
       s=1;
       j=i-1;
       while (s<=j)
       {   m=(2) A. (s+j)/2;
         if( x<a[m]．key)
              (3) B. j=m-1;
         else
              (4) E.s=m+1;
       }
       for ( k=i-1;k>=j+1;k- -)
          (5) C. a[k+1]=a[k];
      a[j+1]=a[0];
   }
}
```

31、以下程序是中序遍历二叉树的递归算法的程序,完成程序中空格部分(树结构中,左、右指针域分别为 left 和 right,数据域 data 为字符型,BT 指向根结点)。

```
void Inorder (struct BTreeNode*BT)
}
if(BT! =NULL){
(1) Inorder(BT->left);
(2) printf("%c", BT->data);
Inorder(BT->right);
}
```

利用上述程序对右图进行遍历，结果是(3) bedfa



32、以下程序是中序遍历二叉树的递归算法的程序,完成程序中空格部分(树结构中左、右指针域分别为 left 和 right,数据域 data 为字符型,BT 指向根结点)。

```
voidlnorder(structBTreeNode*BT)
{
if(BT!=NULL){
(1)Inorder(BT 一>left);
(2)printf("%c"，BT 一>data);
Inorder(BT 一>right);}
}
```

利用上述程序对右图进行遍历，结果是(3)bedafc;

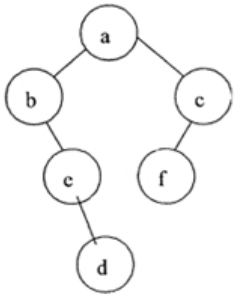


图 2

33、以下函数为链队列的入队操作,x 为要入队的结点的数据域的值,front、rear 分别是链队列的队头、队尾指针

```
struct node
        {   ElemType data;
            struct node *next;
        };
        struct node *front，*rear;
    void InQueue(ElemType x)
      {
        struct node *p;
        p= (struct node*) malloc(sizeof (struct node));
        p->data=x;
              p->next=NULL;
     rear->next=p;
     rear=p ;        }
```

34、以下函数为直接选择排序算法，对 a[1],a[2],…a[n]中的记录进行直接选择排序。typedef struct

```
typedef struct
{   int key;
      ……
}NODE;
 void selsort(NODE a[],int n)
{
    int i,j,k;
    NODE temp;
    for( i=1; i<= (1) E. n-1; i++)
    {
          k=i;
          for( j=i+1;j<= (2) A. n; j++)
                if(a[j]. key<a[k]. key) (3) C. k=j;
          if( i!=k)
          {
                temp=a[i];
                (4) B. a[i]=a[k];
                (5) D. a[k]=temp;
          }
    }
}
```

35、以下函数在 a[0]到 a[n-1]中,用折半查找算法查找关键字等于 k 的记录,查找成功返回该记录的下标,失败时返回-1,完成程序中的空格选项。

```
typedef struct
{ int key;
……
}NODE;
int Binary_Search（NODE a[ ]， int n， int k）
{ int low， mid， high;
  low=0;
  high=n-1;
  while （①low<=high）
  { mid=（①low<=high）
    if （a[mid].key==k）
      return mid;
    else
if （a[mid].key<k）
      low=mid+1;
     else
high=mid-1;
  }
  return -1
}
```

36、以下函数在 a[0]到 a[n-1]中,用折半查找算法查找关键字等于 k 的记录,查找成功区回该记录的下标,失败时返回-1,完成程序中的空格。

```
typedef struct
{ int key;
....
}NODE;
int Binary_Search(NODEa[], int n, int k)
{
int low, mid, high;
 low=0;
(1) high=n--1
while(_low<=high)
{
mid=(2) low+high)/2
if(a[ mid]. key==k)
return(3) mid
else if(a[ mid]. key<k)
low=mid+1;
else(4) high=mid-1}
(5) return-1
}
```

37、**以下函数在 a[0]到 a[n-1]中，用折半查找算法查找关键字等于 k 的记录，查找成功返回该记录的下标，失败时返回-1，完成程序中的空格选项。**

```
typedef    struct
{   int   key;
……
}NODE;
int Binary_Search(NODE a[ ], int n, int k)
{   int low, mid, high;
   low=0;
   high=n-1;
   while (_________)
   {   mid=(low+high)/2)
      if (a[mid].key==k)
         return mid;
      else
        if (a[mid].key<k)
          low=mid+1;
        else
        high=mid-1;
   }
   return -1
}
```

答案：D.low<=high

38、**以下函数在 head 为头指针的具有头结点的单向链表中删除第 i 个结点，**

```
   struct  node
            {    int   data;
                 struct   node   *next;
           };
           typedef   struct   node   NODE
           int delete(NODE *head,int i )
            {
            NODE *p,*q;
             int j;
             q=head;
            j=0;
             while((q!=NULL)&&( __j<i-1____))
            {
            __q=q->next___;
            j++;
            }
             if(q==NULL)
             return(0);
             p= __ q->next____;
            __q->next____=p->next;
             free(___p ____);
             return(1);
             }
```

39、**以下利用直接插入排序算法对存放在 a[0],a[1],...,a[n-1]中,长度为 n 的记录序列按关键字 key 由小到大排序,完成程序中空格部分。**

```
        Void disort (NODE a[], int n)
        {  f int i, j:
         NODE temp: for(i=l: i<n; i++)
                { temp=a[i]:
                j=i-l:
                while(j>=0& & temp. key<a[j]. key)
        {a[j+1]—a[j]:
        j—
        a[j+1]=temp;
        }
        }
```

40、**以下冒泡法程序对存放在 a[1]，a[2]，……，a[n]中的序列进行排序，其中 n 是元素个数，要求按升序排列。void bsort (NODE a[ ], int n)**

```
          void bsort (NODE   a[ ], int n)
           { NODE temp;
             int i,j,flag;
             for(j=1; (1) B. j<=n-1;j++)
              {   flag=0;
                for(i=1; (2) E.i<=n-j;i++)
                  if(a[i]．key>a[i+1]．key)
                          { flag=1;
                           temp=a[i];
                           (3) A. a[i]=a[i+1];
                           (4) C. a[i+1]=temp;
                           }
                 if(flag= =0) break;
                 }
           }
```

程序中 flag 的功能是(5) D. 当某趟冒泡中没有出现交换则已排好序结束循环。

41、**以下是采用选择排序对 10 个数按照从小到大排列的代码,(1)处应该是（）。**

```
for（i=1;i<10;i++）
{ k=i;
  for（j=i+1;j<=10;j++）
    if（a[j]<a[k]）k=j;<A[K]）k=“j;
 if（i!=k）
  { x=a[i];a[i]=a[k];a[k]=x;}
}
```

42、**以下是冒泡排序算法对存放在 a[1],a[2],...,a[n]中序列按关键字 key 由小到大排序，完成程序中空格部分。**

```
void bsort (NODE a[ ], int n)
{ int   i,j,flag;
         NODE temp;
         for (j=1;j<=n-1;j++)
         { flag=0;
         for (i=1;i<=n-j;i++)
          if (_________________)
          { flag=1;
          temp=a[i];
          a[i]=a[i+1];
          a[i+1]=temp;
          }
   if (flag==0) break;
          }
}
```

答案：A.a[i].key>a[i+1].key

43、**以下是用尾插法建立带头结点且有 n 个结点的单向链表的程序,结点中的数据域从前向后依次为 1,2,3,……,n,完成程序中空格部分。**

```
 NODE *create(n)
       {NODE *head , *p, *q;
        int   i;
        p=(NODE*)malloc(sizeof(NODE));
         head=_p____; q=p___;p◊next=NULL; /*建立头结点*/
         for(i=1; i<=n; i++)
         { p=_(NODE*)malloc(sizeof(NODE))__;
          p◊data=i;
          p◊next=NULL;
          q◊next= _p____;
          __q=p__;
          }
          return(head);
       }
```

44、**以下是中序遍历二叉树的递归算法的程序,完成程序中空格部分(树结构中左、右指针域分别为 left 和 right,数据域 data 为字符型,BT 指向根结点)。**

```
void Inorder (struct BTreeNode *BT)
{
  if(BT!=NULL){
    （1）Inorder(BT->left);
    （2）printf("%c",BT->data);
    Inorder(BT->right);}
 }
```

45、**以下为求二叉树深度的算法,完成程序中空格部分。**

```
intBTreeDepth(BTreeNode*BT)
        {if (BT==NULL)
        return O;
        else
        {(int dep1=BTreeDepth(BT->left); / *计算左子树的深度* /
        Int dep2=BTreeDepth(BT->right); / *计算右子树的深度* /
        if(dep1>dep2)
        return depl+l;
        else
        return dep2+!;
        }
        }
```

46、**以下直接插入排序算法对存放在 a[0],a[1],•••,a[n-1]中，长度为 n 的记录序列按关键字 key 由小到大排序。void disort (NODE a[ ], int n)**

```
    void disort (NODE a[ ], int n)
    {   int  i,j;
          NODE  temp;
          for (i=1;i<n;i++)
          {   temp=a[i];
              j=j-1;
              while ((1) B. j>=0&&temp．key<a[j]．key)
                {   a[j+1]= (2) D. a[j];
                      (3) A. j--;
                }
              a[j+1]= (4) C. temp;
          }
    }
```

47、**在下面空格处填写适当的语句，以使下面的循环队列的入队和出队算法完整。define MAXSIZE 100;**

```
define MAXSIZE 100;
typedef char Elemtype;
typedef struct
{
   Elemtype queue [MAXSIZE];
   int front,rear;
}sequeuetype;
Sequeuetype Q;
int encqueue(sequeuetype*Q,elemtype x)
□
if ((Q->rear+1)%MAXSIZE==Q->front)
□printf(“队列已满!\n”);
return 1;
□else
□       Q->rear=(Q->rear+1)%MAXSIZE;
          (1)
      return 0;
} /*入队算法*/
Elemtype  del_cqueue(sequeuetype *Q)
□
   if (      (2)      )
   □
      printf(“队列为空!\n”);
      return 1;
   else
          Q->front=(Q->front+1)%MAXSIZE;
      return(Q-□queue[Q->front]);
   □
□ /*出队算法*/
```

答案选：D. (1) Q->queue[Q->rear]=x; (2) Q->front==Q->rear

48、**在下面空格处填写一条语句,以使下面的串连接算法完整。char *strcat(char *s1,char *s2) {**

```
char *strcat(char *s1,char *s2) {
  char *p=s1;
while(*p!='\0') p++;
while(*s2!='\0') {
  *p=*s2;
 p++;
s2++;
} *p='\0';
return s1;
}
```

49、**在下面空格处填写一条语句,以使下面的进栈算法完整。**

```
void Push（struct SeqStack*s，ElemType x）
  { if （s->top==MaxSize-1）
{  printf（“栈满溢出错误！\n”）；
     exit（1）；
   }
   s->top++;
   s->data[s->top]=x;
```

50、**在下面空格处填写一条语句,以使下面的链式队列全部元素出队的算法完整。**

```
Int write（LinkQueue*q）
{QueueNode*p：if（q->front==q->rear）/*队空*/
{printf（“队空！无元素可取”）;
exit（0）；}
while（q->front->next！=NULL）
{p=q->front->next：
}
q->front->next=p->next；/*出队*/
printf（"%4d”，p->data）；free（p）；/*释放已出队结点*/
q->rear-q->front;
```

51、**在下面空格处填写一条语句,以使下面的循环队列入队算法完整。**

```
    {printf(“栈下溢错误！\n”);
    exit ;
    }
    x=s->data[s->top];
     s->top--;
    return x;
}
```

52、**在下面空格处填写一条语句,以使下面的循环队列入队算法完整。**

```
void InQueue(struct SeqQueue*sq，int x)
{if(sq->rear+1)%MaxSize==sq->front)
{printf(“循环队列已满！\n”);
exit(1);
}
sq->rear=(sq->rear+1)%MaxSize;
sq->data[sq->rear]=x;
}
```

53、**在下面空格处填写一条语句，以使下面的出栈算法完整。ElemType Pop(struct SeqStack*s,ElemType x)**

```
ElemType Pop(struct SeqStack*s,ElemType x)
  {
     If (StackEmpty(s)){
       printf(“栈下溢出错误！\n”);
       exit(1);
     }
  x=s->data[s->top];
  A. s->top--;
  return x;
  }
```

54、**在下面空格处填写一条语句，以使下面的串比较算法完整。**

```
int strcmp(char *s1,char *s2)
{   int i;
    for(i=0;s1[i]!='\0'&& s2[i]!='\0';i++)
        if(s1[i]>s2[i])
                return 1;
        else
        if(s1[i]<s2[i])
            return -1;
    if(s1[i]=='\0'&& s2[i]=='\0')
        _________
    else
       if(s1[i]!='\0')
         return 1;
       else
           return -1;
}
```

答案：A.return 0;

55、**在下面空格处填写一条语句，以使下面的串复制算法完整。**

```
char *strcpy(char *s1,char *s2)
{   char *p=s1;
     while(*s2!='\0')
     {   *p=*s2;
         s2++;
         _____
     }
    *p='\0';
         return s1;
}
```

答案：B.p++;

56、**在下面空格处填写一条语句，以使下面的串连接算法完整。**

```
char *strcat(char *s1,char *s2)
{   char *p=s1;
     while(*p!='\0')
     p++;
     while(*s2!='\0')
     {   *p=*s2;
         p++;
         _____
     }
```

```
    *p='\0';
        return s1;
}
```
答案：D.s2++;

57、**在下面空格处填写一条语句，以使下面的进栈算法完整。void Push(struct SeqStack*s,ElemType x)**

```
void Push(struct SeqStack*s,ElemType x)
  {
     If (s->top==MaxSize-1){
       printf("栈满溢出错误！\n");
       exit(1);
     }
  C. s->top++;
  s->data[s->top]=x;
  }
```

58、**在下面空格处填写一条语句，以使下面的顺序队列入队算法完整。**

```
void InQueue(struct SeqQueue *sq, int x)
{   if (sq->rear==MaxSize)
          {   printf("队列已满！\n");
        exit(1);
    }
    _____
    sq->rear++;
}
```
答案：D.sq->data[sq->rear]=x;

59、**在下面空格处填写一条语句，以使下面的循环队列出队算法完整。**

```
ElemType OutQueue(struct SeqQueue *sq)
{   if (sq->rear==sq->front)
          {   printf("队列已空，不能进行出队操作！\n");
        exit(1);
    }
    _____
    sq->front=(sq->front+1)%MaxSize;
        return x;
}
```
答案：C.x=sq->data[sq->front];