**一、选择题(30分)**

　　1. 设一组权值集合W={2，3，4，5，6}，则由该权值集合构造的哈夫曼树中带权路径长度之和为( )。

　　(A) 20 (B) 30 (C) 40 (D) 45

　　2.执行一趟快速排序能够得到的序列是( )。

　　(A) [41，12，34，45，27] 55 [72，63]

　　(B) [45，34，12，41] 55 [72，63，27]

　　(C) [63，12，34，45，27] 55 [41，72]

　　(D) [12，27，45，41] 55 [34，63，72]

　　3.设一条单链表的头指针变量为head且该链表没有头结点，则其判空条件是( )。

　　(A) head==0 (B) head->next==0

　　(C) head->next==head (D) head!=0

　　4.时间复杂度不受数据初始状态影响而恒为O(nlog2n)的是( )。

　　(A) 堆排序 (B) 冒泡排序 (C) 希尔排序 (D) 快速排序

　　5.设二叉树的先序遍历序列和后序遍历序列正好相反，则该二叉树满足的条件是( )。

　　(A) 空或只有一个结点 (B) 高度等于其结点数

　　(C) 任一结点无左孩子 (D) 任一结点无右孩子

　　6.一趟排序结束后不一定能够选出一个元素放在其最终位置上的是( )。

　　(A) 堆排序 (B) 冒泡排序 (C) 快速排序 (D) 希尔排序

　　7.设某棵三叉树中有40个结点，则该三叉树的最小高度为( )。

　　(A) 3 (B) 4 (C) 5 (D) 6

　　8.顺序查找不论在顺序线性表中还是在链式线性表中的时间复杂度为( )。

　　(A) O(n) (B) O(n2) (C) O(n1/2) (D) O(1og2n)

　　9.二路归并排序的时间复杂度为( )。

　　(A) O(n) (B) O(n2) (C) O(nlog2n) (D) O(1og2n)

　　10. 深度为k的完全二叉树中最少有( )个结点。

　　(A) 2k-1-1 (B) 2k-1 (C) 2k-1+1 (D) 2k-1

　　11.设指针变量front表示链式队列的队头指针，指针变量rear表示链式队列的队尾指针，指针变量s指向将要入队列的结点X，则入队列的操作序列为( )。

　　(A) front->next=s;front=s; (B) s->next=rear;rear=s;

　　(C) rear->next=s;rear=s; (D) s->next=front;front=s;

　　12.设某无向图中有n个顶点e条边，则建立该图邻接表的时间复杂度为( )。

　　(A) O(n+e) (B) O(n2) (C) O(ne) (D) O(n3)

　　13.设某哈夫曼树中有199个结点，则该哈夫曼树中有( )个叶子结点。

　　(A) 99 (B) 100 (C) 101 (D) 102

　　14.设二叉排序树上有n个结点，则在二叉排序树上查找结点的平均时间复杂度为( )。

　　(A) O(n) (B) O(n2) (C) O(nlog2n) (D) O(1og2n)

　　15.设用邻接矩阵A表示有向图G的存储结构，则有向图G中顶点i的入度为( )。

　　(A) 第i行非0元素的个数之和 (B) 第i列非0元素的个数之和

　　(C) 第i行0元素的个数之和 (D) 第i列0元素的个数之和

**二、判断题(20分)**

　　1.调用一次深度优先遍历可以访问到图中的所有顶点。( )

　　2.分块查找的平均查找长度不仅与索引表的长度有关，而且与块的长度有关。( )

　　3.冒泡排序在初始关键字序列为逆序的情况下执行的交换次数最多。( )

　　4.满二叉树一定是完全二叉树，完全二叉树不一定是满二叉树。( )

　　5.设一棵二叉树的先序序列和后序序列，则能够唯一确定出该二叉树的形状。( )

　　6.层次遍历初始堆可以得到一个有序的序列。( )

　　7.设一棵树T可以转化成二叉树BT，则二叉树BT中一定没有右子树。( )

　　8.线性表的顺序存储结构比链式存储结构更好。( )

　　9.中序遍历二叉排序树可以得到一个有序的序列。( )

　　10.快速排序是排序算法中平均性能最好的一种排序。( )

**三、填空题(30分)**

　　1.for(i=1，t=1，s=0;i<=n;i++) {t=t\*i;s=s+t;}的时间复杂度为\_\_\_\_\_\_\_\_\_。

　　2.设指针变量p指向单链表中结点A，指针变量s指向被插入的新结点X，则进行插入操作的语句序列为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(设结点的指针域为next)。

　　3.设有向图G的二元组形式表示为G =(D，R)，D={1，2，3，4，5}，R={r}，r={<1,2>，<2,4>，<4,5>，<1,3>，<3,2>，<3,5>}，则给出该图的一种拓扑排序序列\_\_\_\_\_\_\_\_\_\_。

　　4.设无向图G中有n个顶点，则该无向图中每个顶点的度数最多是\_\_\_\_\_\_\_\_\_。

　　5.设二叉树中度数为0的结点数为50，度数为1的结点数为30，则该二叉树中总共有\_\_\_\_\_\_\_个结点数。

　　6.设F和R分别表示顺序循环队列的头指针和尾指针，则判断该循环队列为空的条件为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

　　7.设二叉树中结点的两个指针域分别为lchild和rchild，则判断指针变量p所指向的结点为叶子结点的条件是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

　　8.简单选择排序和直接插入排序算法的平均时间复杂度为\_\_\_\_\_\_\_\_\_\_\_。

　　9.快速排序算法的空间复杂度平均情况下为\_\_\_\_\_\_\_\_\_\_，最坏的情况下为\_\_\_\_\_\_\_\_\_\_。

　　10.散列表中解决冲突的两种方法是\_\_\_\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_\_\_\_。

**四、算法设计题(20分)**

1.1. 设计在顺序有序表中实现二分查找的算法。

　　2.2. 设计判断二叉树是否为二叉排序树的算法。

3.3. 在链式存储结构上设计直接插入排序算法。

**下面请看2015年考研：计算机数据结构测试题（五）答案**

**一、选择题**

　　1.D 2.A 3.A 4.A 5.D

　　6.D 7.B 8.A 9.C 10.B

　　11.C 12.A 13.B 14.D 15.B

**二、判断题**

　　1.错 2.对 3.对 4.对 5.错

　　6.错 7.对 8.错 9.对 10.对

**三、填空题**

　　1. 1. O(n)

　　2. 2. s->next=p->next; p->next=s

　　3. 3. (1，3，2，4，5)

　　4. 4. n-1

　　5. 5. 129

　　6. 6. F==R

　　7. 7. p->lchild==0&&p->rchild==0

　　8. 8. O(n2)

　　9. 9. O(nlog2n)， O(n)

　　10. 10. 开放定址法，链地址法

**四、算法设计题**

　　1. 1. 设计在顺序有序表中实现二分查找的算法。

　　struct record {int key; int others;};

　　int bisearch(struct record r[ ], int k)

　　{

　　int low=0,mid,high=n-1;

　　while(low<=high)

　　{

　　mid=(low+high)/2;

　　if(r[mid].key==k) return(mid+1); else if(r[mid].key>k) high=mid-1; else low=mid+1;

　　}

　　return(0);

　　}

　　2. 2. 设计判断二叉树是否为二叉排序树的算法。

　　int minnum=-32768,flag=1;

　　typedef struct node{int key; struct node \*lchild,\*rchild;}bitree;

　　void inorder(bitree \*bt)

　　{

　　if (bt!=0) {inorder(bt->lchild); if(minnum>bt->key)flag=0; minnum=bt->key;inorder(bt->rchild);}

　　}

　　3. 3. 在链式存储结构上设计直接插入排序算法

　　void straightinsertsort(lklist \*&head)

　　{

　　lklist \*s,\*p,\*q; int t;

　　if (head==0 || head->next==0) return;

　　else for(q=head,p=head->next;p!=0;p=q->next)

　　{

　　for(s=head;s!=q->next;s=s->next) if (s->data>p->data) break;

　　if(s==q->next)q=p;

　　else{q->next=p->next; p->next=s->next; s->next=p; t=p->data;p->data=s->data;s->data=t;}

　　}

　　}