

选择题:

1. 在数据结构中, 从逻辑上可以把数据结构分成 ( )。  
A. 动态结构和静态结构      B. 紧凑结构和非紧凑结构  
C. 线性结构和非线性结构      D. 内部结构和外部结构
2. 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 ( )。  
A. 存储结构      B. 存储实现  
C. 逻辑结构      D. 运算实现
3. 数据结构是指 ( )。  
A. 数据定义      B. 数据存储结构  
C. 数据类型      D. 数据元素的组织形式
4. 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 ( )。  
A. 必须是连续的      B. 部分地址必须是连续的  
C. 一定是不连续的      D. 连续或不连续都可以
5. 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 ( )。  
A. 110      B. 108      C. 100      D. 120
6. 单链表的存储密度 ( )。  
A. 大于 1      B. 等于 1      C. 小于 1      D. 不能确定
7. 栈在 ( ) 中有所应用。  
A. 递归调用      B. 函数调用      C. 表达式求值      D. 前三个选项都有
8. 栈和队列的共同特点 ( )。  
A. 都是先进先出      B. 只允许在端点处插入和删除元素  
C. 都是先进后出      D. 没有共同点
9. 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。  
A. 5, 4, 3, 2, 1      B. 2, 1, 5, 4, 3  
C. 4, 3, 1, 2, 5      D. 2, 3, 5, 4, 1
10. 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 ( )。  
A.  $n$       B.  $2n-1$       C.  $2n$       D.  $n-1$
11. 用链接方式存储的队列, 在进行删除运算时 ( )。  
A. 仅修改头指针      B. 仅修改尾指针  
C. 头、尾指针都要修改      D. 头、尾指针可能都要修改
12. 串的长度是指 ( )。  
A. 串中所含不同字母的个数      B. 串中所含字符的个数  
C. 串中所含不同字符的个数      D. 串中所含非空格字符的个数
13. 在下列存储结构表示法中, ( ) 不是树的存储结构表示法。  
A. 顺序存储表示法      B. 孩子链表表示法  
C. 孩子兄弟表示法      D. 双亲表示法

14. 把一棵树转换为二叉树后，这棵二叉树的形态是 ( )。
- A. 唯一的 B. 有多种  
C. 有多种，但根结点都没有左孩子 D. 有多种，但根结点都没有右孩子
15. 一个具有 513 个结点的二叉树的高  $h$  为 ( )。
- A. 10 B. 9 C. 10 至 513 之间 D. 9 至 513 之间
16. 图的广度优先遍历类似于二叉树的 ( )。
- A. 先序遍历 B. 中序遍历  
C. 后序遍历 D. 层次遍历
17. 图的深度优先遍历类似于二叉树的 ( )。
- A. 先序遍历 B. 中序遍历 C. 后序遍历 D. 层次遍历
18. 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。
- A. 栈 B. 队列 C. 树 D. 图
19. 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 ( )。
- A. 所有的结点均无左孩子 B. 所有的结点均无右孩子  
C. 是任意一棵二叉树 D. 只有一个叶子结点
20. 在一个无向图中，所有顶点的度数之和等于图的边数的 ( ) 倍。
- A.  $1/2$  B. 1 C. 2 D. 4
21. 设无向图  $G$  中有  $n$  个顶点，则该无向图的最小生成树上有 ( ) 条边。 \*
- A.  $n$  B.  $n-1$  C.  $2n$  D.  $2n-1$
22. 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。
- A.  $1/2$  B. 1 C. 2 D. 4
23. 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。
- A. 3 B. 4 C. 5 D. 6
24. 下面关于散列查找的说法，正确的是 ( )。
- A. 散列函数构造得越复杂越好，因为这样随机性好、冲突小  
B. 除留余数法是所有散列函数中最好的  
C. 不存在特别好与坏的散列函数，要视情况而定  
D. 散列表的平均查找长度有时也和记录总数有关
25. 适用于折半查找的表的存储方式及元素排列要求为 ( )。
- A. 链接方式存储，元素无序 B. 链接方式存储，元素有序  
C. 顺序方式存储，元素无序 D. 顺序方式存储，元素有序
26. 二叉排序树中左子树上所有结点的值均 ( ) 根结点的值。
- A.  $<$  B.  $=$  C.  $>$  D.  $!=$
27. 下述几种排序方法中，( ) 是稳定的排序方法。
- A. 希尔排序 B. 归并排序  
C. 快速排序 D. 堆排序

28. 从未排序序列中依次取出元素与已排序序列中的元素进行比较, 将其放入已排序序列的正确位置上的方法, 这种排序方法称为 ( )。

- A. 归并排序                      B. 冒泡排序  
C. 插入排序                      D. 选择排序

29. 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 ( )。

- A.  $O(n)$                       B.  $O(n^2)$                       C.  $O(n \log_2 n)$                       D.  $O(n^3)$

30. 堆的形状是一棵 ( )。

- A. 完全二叉树                      B. 满二叉树                      C. 二叉排序树                      D. 平衡二叉树

填空题:

1. 数据结构中评价算法的两个重要指标是时间复杂度和 空间复杂度。
2. 数据结构包括两个方面的内容: 数据的 逻辑结构 和存储结构。
3. 栈和队列都是操作受限的线性表, 栈的操作特点是后进先出, 队列的操作特点是 先进先出。
4. 树中度数为零的结点称为叶子或终端结点, 度数不为零的结点称为 非终端结点。
5. 中序遍历二叉排序树所得到的序列是 有序 序列 (填有序或无序)。
6. 设广义表  $L = ((a, b, c))$ , 则  $L$  的长度为 1, 深度为 2。
7. 通常从四个方面评价算法的质量: 正确性、易读性、强壮性和高效性。
8. 广义表  $((a, b, c, d))$  的表头是  $(a, b, c, d)$ , 表尾是 ( )。
9. 数据结构从逻辑上可以划分为三种基本类型: 线性结构、树形结构和图状结构 (或称为图形结构)。
10. 图的遍历包括 深度优先搜索遍历 和广度优先搜索遍历两种方法。
11. 线性结构元素之间的关系是 一对一 关系, 树形结构元素之间的关系是一对多关系, 图形结构元素之间的关系是多对多关系。
12. 链式存储结构中每个结点由数据域和 指针域 两部分组成。
13. 对于一个具有  $n$  个顶点和  $e$  条边的有向图和无向图, 在其对应的邻接表中, 所含边结点分别有  $e$  个和  $2e$  个。
14. 散列表中解决冲突的两种方法是 开放定址法 和链地址法。
15. 在快速排序, 冒泡排序, 希尔排序和堆排序中, 快速排序 的空间复杂度最大。

### 算法设计题：

以下是二叉树的二叉链表存储表示：

```
typedef struct BiTNode {  
    TElemType data;  
    struct BiTNode *lchild, *rchild; //左右孩子指针  
} BiTNode, *BiTree;
```

请给出二叉树的 先序 、 中序 、 后序 遍历算法。(三种算法中的一种)

//先序遍历算法

```
Void PreOrderTraverse (BiTree T) {  
    if (T) {  
        cout << T->data;  
        PreOrderTraverse (T->lchild);  
        PreOrderTraverse (T->rchild);  
    }  
}
```

//中序遍历算法

```
Void InOrderTraverse (BiTree T) {  
    if (T) {  
        InOrderTraverse (T->lchild);  
        cout << T->data;  
        InOrderTraverse (T->rchild);  
    }  
}
```

//后序遍历算法

```
Void PostOrderTraverse (BiTree T) {  
    if (T) {  
        PostOrderTraverse (T->lchild);  
        PostOrderTraverse (T->rchild);  
        cout << T->data;  
    }  
}
```

### 算法分析题：

6. 试分析下面各程序段的时间复杂度。

```
(1) x=90; y=100;
    while(y>0)
        if(x>100)
            {x=x-10;y--;}
        else x++;
```

答案： $O(1)$

解释：程序的执行次数为常数阶。

```
(2) for (i=0; i<n; i++)
    for (j=0; j<m; j++)
        a[i][j]=0;
```

答案： $O(m*n)$

解释：语句  $a[i][j]=0;$  的执行次数为  $m*n$ 。

```
(3) s=0;
    for i=0; i<n; i++)
        for(j=0; j<n; j++)
            s+=B[i][j];
```

sum=s;

答案： $O(n^2)$

解释：语句  $s+=B[i][j];$  的执行次数为  $n^2$ 。

```
(4) x=0;
    for(i=1; i<n; i++)
        for (j=1; j<=n-i; j++)
            x++;
```

答案： $O(n^2)$

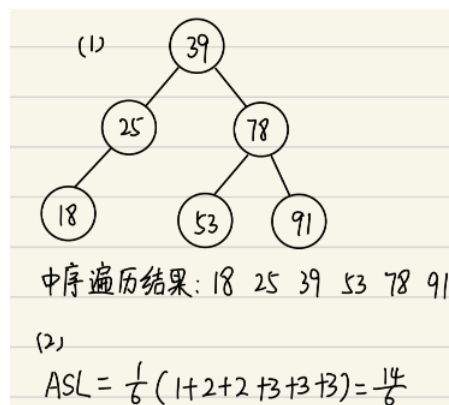
解释：语句  $x++;$  的执行次数为  $n-1+n-2+\cdots+1= n(n-1)/2$ 。

### 应用题：

1. 现有一棵空的二叉排序树，请根据关键字序列 (39, 25, 18, 78, 53, 91)，完成以下问题。

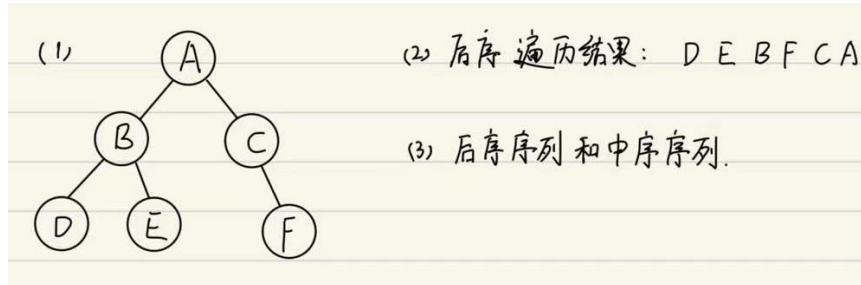
1) 画出插入关键字后的二叉排序树，并给出中序遍历结果；

2) 计算出该二叉排序树的平均查找长度。



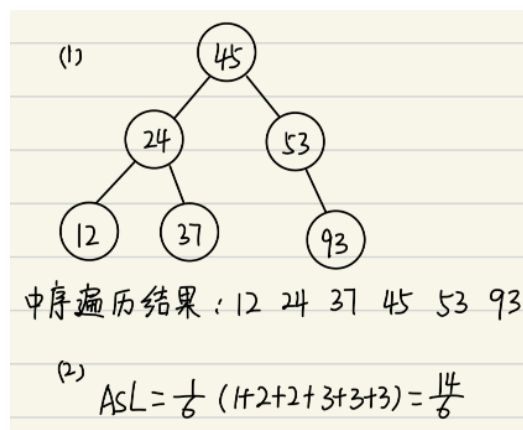
2. 已知一棵二叉树的先序序列为 ABDECF, 中序序列为 DBEACF.

- 1) 请画出这棵 **二叉树**;
- 2) 请给出该二叉树的 **后序遍历结果**;
- 3) 除先序序列和中序序列组合外, 还可以根据二叉树遍历的 **哪两种遍历序列**, 能唯一的确定一棵二叉树?



3. 现有一棵空的二叉排序树, 请根据关键字序列 (45, 24, 53, 12, 37, 93), 完成以下要求。

- 1) 画出插入关键字后的 **二叉排序树**, 并给出 **中序遍历结果**;
- 2) 计算出该二叉排序树的 **平均查找长度**。

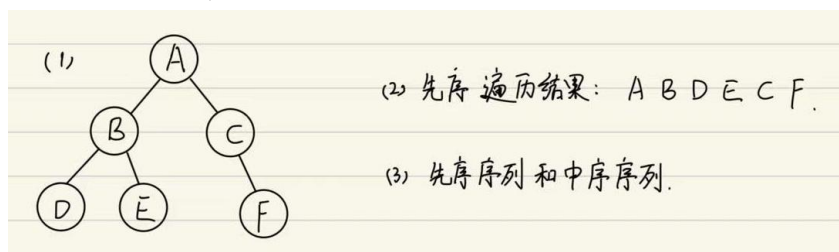


4. 设待排序的关键字序列为 {21, 25, 49, 25\*, 16, 8}, 试写出使用 **冒泡排序** 方法, 每趟排序结束后关键字序列的状态。

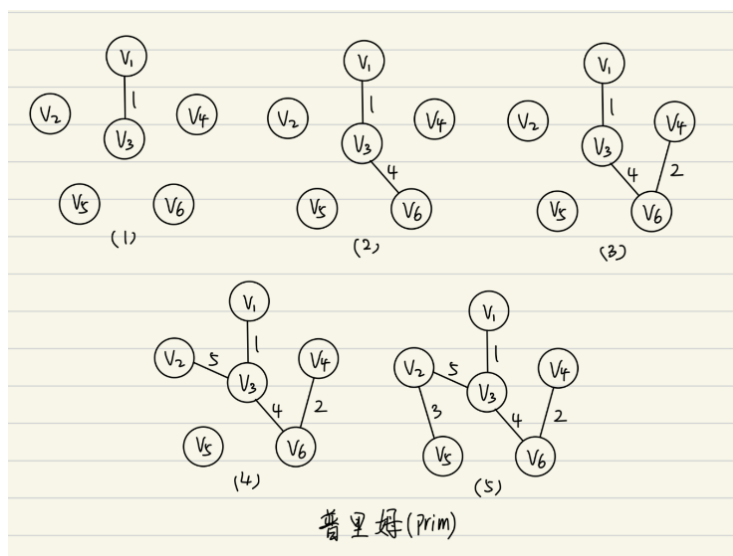
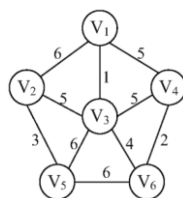
1、	21	25	25*	16	8	49
2、	21	25	16	8	25*	49
3、	21	16	8	25	25*	49
4、	16	8	21	25	25*	49
5、	8	16	21	25	25*	49

5. 已知一棵二叉树的后序序列为 DEBFCA, 中序序列为 DBEACF.

- 1) 请画出这棵 **二叉树**;
- 2) 请给出该二叉树的 **先序遍历结果**;
- 3) 除后序序列和中序序列组合外, 还可以根据二叉树遍历的 **哪两种遍历序列**, 能唯一的确定一棵二叉树?



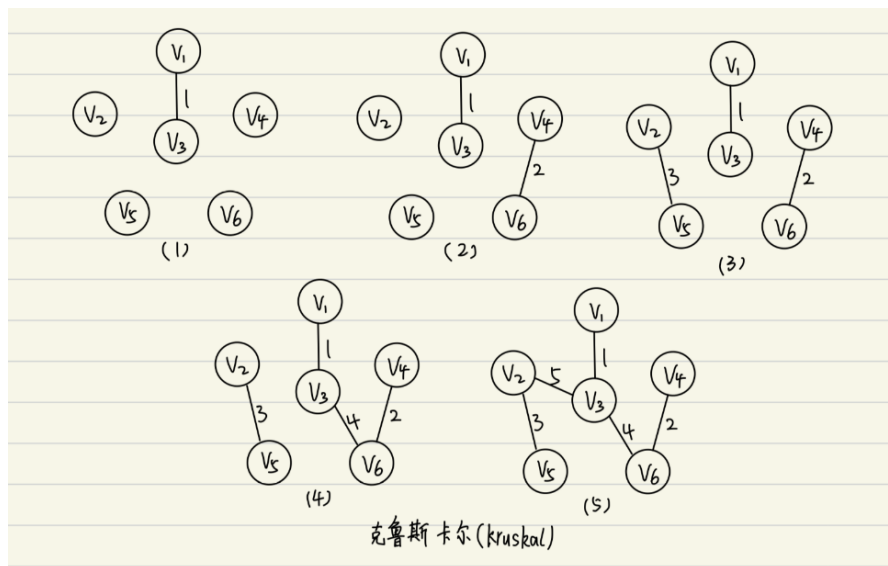
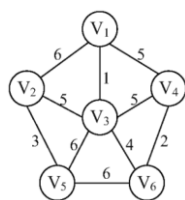
6. 根据给定的无向图, 采用 **普里姆 (Prim)** 算法思想画出下图的最小生成树 (注: 画出完整的构造过程).



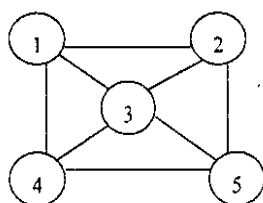
7. 设待排序的关键字序列为 {62, 34, 48, 25, 12, 22}, 试写出使用 **直接插入排序** 方法, 每趟排序结束后关键字序列的状态.

1.	34	62	48	25	12	22
2.	34	48	62	25	12	22
3.	25	34	48	62	12	22
4.	12	25	34	48	62	22
5.	12	22	25	34	48	62

8. 根据给定的无向图，采用克鲁斯卡尔 (Kruskal) 算法思想画出下图的最小生成树 (注：画出完整的构造过程)。



9. 请给出下图的邻接矩阵和邻接表。



邻接矩阵：

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

邻接表：

