1. **填空题：**
2. 数据结构被形式地定义为（K，R），其中K是\_\_元素\_\_\_的有限集合，R是K上的\_\_关系\_\_有限集合。
3. 数据的逻辑结构被分为 \_\_集合\_\_\_ 、 \_\_线形\_\_ 、 \_\_\_树形\_\_ 和 \_\_\_\_图形\_\_ 四种。
4. 以二分查找方法从长度为 10 的有序表中查找一个元素时，平均查找长度为\_\_\_2.9\_\_\_ 。
5. 对于一个长度为 n 的顺序存储的线性表，在表头插入元素的时间复杂度为 \_\_O（n）\_\_\_\_ ，在表尾插入元素的时间复杂度为 \_\_\_\_O（1）\_\_\_\_\_ 。
6. 当用长度为 N 的数组顺序存储一个栈时，假定用 top==N 表示栈空，则表示栈满的条件是 \_\_\_\_ top==0\_\_\_\_\_\_ 。
7. 在线性表的单链接存储中，若一个元素所在结点的地址为p，则其后继结点的地址为\_\_p->next\_\_\_\_\_\_。
8. 删除带头结点的单链表Head的第一个结点的操作是\_\_\_ head->next= head->next ->next \_\_\_，删除不带头结点的单链表Head的第一个结点的操作是\_\_ head = head->next \_\_。
9. 在具有n 个单元的循环队列中，队满时共有\_\_n-1\_\_个元素。
10. 二叉树的第 k层的结点数最多为\_\_2k-1\_\_\_\_\_。
11. 由权值分别为 3,8,6,2 的叶子生成一棵哈夫曼树，它的带权路径长度为\_\_35\_\_。
12. 设有一个二维数组 A [m][n] ，假设 A [0][0] 存放位置在 600 (10) ， A [3][3] 存放位置在 678 (10) ，每个元素占一个字节， A [2][3] (10) 存放在\_\_653 \_位置。 ( 脚注 (10) 表示用 10 进制表示 ,m>3)
13. 对于一棵具有 n 个结点的二叉树，用二叉链表存储时，其指针总数为 \_\_2n\_\_ 个，其中 \_\_n-1\_ 个用于指向孩子， \_\_\_n+1\_\_\_ 个指针是空闲的。
14. 假定一棵树的广义表表示为 A （ D （ E ， G ）， H （ I ， J ）），则树中所含的结点数为\_\_\_7\_\_\_ 个，树的深度为 \_\_\_\_3\_\_\_\_\_ ，树的度为 \_\_\_2\_\_\_\_ 。
15. 对于一棵具有 n 个结点的二叉树，一个结点的编号为 i(1 ≤ i ≤ n) ，若它有左孩子则左孩子结点的编号为 \_\_\_2 i \_\_ ，若它有右孩子，则右孩子结点的编号为 \_\_2 i +1\_\_\_ ，若它有双亲，则双亲结点的编号为 \_\_\_i/2\_\_\_\_\_ 。
16. 在线性表的散列存储中，装填因子 a 又称为装填系数，若用 m 表示散列表的长度， n 表示待散列存储的元素的个数，则 a 等于 \_\_n/m\_\_\_\_\_\_ 。
17. 当向一个大根堆插入一个具有最大值的元素时，需要逐层 \_\_\_\_向上\_\_ 调整，直到被调整到 \_\_\_根结点\_\_\_\_ 位置为止。

**二、单项选择题：**

* 1. 下面关于图的存储的叙述中正确的是\_\_\_\_B\_\_\_\_。

A. 用邻接表法存储图,占用的存储空间大小只与图中边数有关,而与结点个数无关

B. 用邻接表法存储图,占用的存储空间大小只与图中边数和结点个数都有关

C. 用邻接矩阵法存储图,占用的存储空间大小只与图中边数和结点个数都有关

D. 用邻接矩阵法存储图,占用的存储空间大小只与图中边数有关,而与结点个数无关

* 1. 采用开放定址法处理散列表的冲突时，其平均查找长度\_\_\_\_\_B\_\_\_。

A. 低于链接法处理冲突 B. 高于链接法处理冲突

C. 与链接法处理冲突相同 D. 高于二分查找

* 1. 在一个单链表 HL 中，若要在当前由指针 p 指向的结点后面插入一个由 q 指向的结点，则执行如下\_\_\_D\_\_\_\_ 语句序列。

A. p=q； p->next=q； B. p->next=q； q->next=p；

C. p->next=q->next ； p=q ； D. q->next=p->next ； p->next=q ；

* 1. 若有 18 个元素的有序表存放在一维数组 A[19] 中，第一个元素放 A[1] 中，现进行二分查找，则查找 A ［ 3 ］的比较序列的下标依次为\_\_\_D\_\_\_\_。

A. 1，2，3 B. 9，5，2，3 C. 9，5，3 D. 9，4，2，3

* 1. 对线性表进行二分法查找，其前提条件是\_\_C\_\_\_。

A. 线性表以链接方式存储，并且按关键码值排好序

B. 线性表以顺序方式存储，并且按关键码值的检索频率排好序

C. 线性表以顺序方式存储，并且按关键码值排好序

D. 线性表以链接方式存储，并且按关键码值的检索频率排好序

* 1. 广义表 A= (10,(19,1),((2,1),1) ， 0), 则它的深度为\_\_\_\_A\_\_\_\_。

A. 3 B. 5 C. 4 D. 2

* 1. 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 \_\_\_D\_\_\_\_。

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

* 1. 带头结点的单链表head 空的判定条件是\_\_B\_\_\_。

A. head = = null B. head 🡪next = = null

C. head 🡪next = = head D. head ! = null

* 1. 在一个单链表中，已知q所指向的结点是p所指向结点的前驱结点，若在q 和p 之间插入s结点，则执行\_\_\_C\_\_\_。

A. s🡪next = p🡪next ； p🡪next=s B. p🡪next=s🡪next； s🡪next = p

C. q🡪next = s；s🡪next = p D. p🡪next=s；s🡪next=q

* 1. 设有关键码序列 (q ， g ， m ， z ， a) ，下面哪一个序列是从该序列出发建的小根堆的结果\_\_B\_\_\_。

A. a，g，m，q，z

B. a，g，m，z，q

C. g，m，q，a，z

D. g，m，a，q，z

* 1. 如果线性表中最常用的操作是存取第i个元素及其前驱的值，则采用\_\_D\_\_。

A. 单链表 B. 双链表 C. 单循环链表 D. 顺序表

**三、运算题：**

1. 计算由权值分别为4，7，6，2的叶子生成一棵赫夫曼树，它的带权路径长度为多少？
2. 若某二叉树结点的中序遍历的序列为A，B，C，D，E，F，G，后序遍历的序列为B，D，C，A，F，G，E，则该二叉树结点的前序遍历的序列为多少？

E，A，C，B，D，G，F

1. 已知一个图的顶点集 V 和边集 E 分别为：

V={1,2,3,4,5,6,7}；

E={(1,2)3,(1,3)5,(1,4)8,(2,5)10,(2,3)6,(3,4)15,(3,5)12,(3,6)9,(4,6)4,

(4,7)20,(5,6)18,(6,7)25}；

按照普里姆算法从顶点1出发得到最小生成树，试写出在最小生成树中依次得到的各条边。

(1,2)3, (1,3)5, (1,4)8, (4,6)4, (2,5)10, (4,7)20

1. 已知一个图的顶点集 V 和边集 E 分别为：

V={1,2,3,4,5,6,7}；

E={<2,1>,<3,2>,<3,6>,<4,3>,<4,5>,<4,6>,<5,1>,<5,7>,<6,1>,<6,2>,<6,5>}；

若存储它采用邻接表，并且每个顶点邻接表中的边结点都是按照终点序号从小到大的次序链接的，按主教材中介绍的拓朴排序算法进行排序，试给出得到的拓朴排序的序列。

4 3 6 5 7 2 1

1. 若某二叉树结点的中序遍历的序列为A，B，C，D，E，F，G，前序遍历的序列为E，A，C，B，D，G，F

则该二叉树结点的后序遍历的序列为多少？

B，D，C，A，F，G，E，

**四、算法题：**

1. 向单链表的末尾添加一个元素 :

Void InsertRear(LNode \*& HL,Elemtype item)

{ LNode \*newptr；Newptr=new LNode；

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_=item；Newptr->next=NULL；LNode \*p=HL；

While(p->next!=NULL) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_；

p->next=newptr；

}

Newptr->data

p=p->next;

1. Void AE(Stack &s)

{InitStack(S)；Push(S,3)；Push(S,4)；Int x=Pop(S)+2\*Pop(S)；Push(S,x)；

Int I， a[5]={1,5,8,12,15}；

For(i=0；i<5；;i++) Push(S，2\*a[i])；

While(!EmptyStack(S)) cout<<Pop(s)；

} 该算法被调用后得到的输出结果：30，24，16，10，2，10

1. 请写出二叉树前序遍历的递归算法。

Void ABC(BTreeNode \*BT)

{if(BT) { Cout<<BT->data；ABC(BT->lchild)；ABC(BT->rchild)；}

1. 请给出16，31，9，15，87，76，13，24，43的第一趟快速排序的结果。

15，13，9，16，87，76，31，24，43

1. 向单链表的末尾添加一个元素 :

Void InsertRear(LNode \*& HL,Elemtype item)

{ LNode \*newptr；Newptr=new LNode；

Newptr->data =item；Newptr->next=NULL；LNode \*p=HL；

While(p->next!=NULL) p=p->next；

p->next=newptr；

}

1. Void AE(Stack &s)

{InitStack(S)；Push(S,3)；Push(S,4)；Int x=Pop(S)+2\*Pop(S)；Push(S,x)；

Int I， a[5]={1,5,8,12,15}；

For(i=0；i<5；;i++) Push(S，2\*a[i])；

While(!EmptyStack(S)) cout<<Pop(s)；

} 该算法被调用后得到的输出结果：30，24，16，10，2，10

1. 请写出二叉树后序遍历的递归算法。

Void ABC(BTreeNode \*BT)

{if(BT) {ABC(BT->lchild)；ABC(BT->rchild)；Cout<<BT->data；}