**《数据结构》复习资料**

一、**单选题**

1. 以下数据结构中哪一个是线性结构？ ( )

A. 有向图 B. 队列 C. 线索二叉树 D. B 树

2. 对一个算法的评价，不包括如下（ ）方面的内容。

A ．健壮性和可读性 B ．并行性 C ．正确性 D ．时空复杂度

3. 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 ( ) 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 ；

4. 对线性表，在下列哪种情况下应当采用链表表示？ ( )

A.经常需要随机地存取元素 B.经常需要进行插入和删除操作

C.表中元素需要占据一片连续的存储空间 D.表中元素的个数不变

5. 以下哪一个不是队列的基本运算？（ ）

A. 在队列第 i 个元素之后插入一个元素 B. 从队头删除一个元素

C. 判断一个队列是否为空 D. 读取队头元素的值

6. 一个栈的输入序列为 1 2 3 ，则下列序列中不可能是栈的输出序列的是 ( )

A. 2 3 1 B. 3 2 1 C. 3 1 2 D. 1 2 3

7. 下述哪一条是顺序存储方式的优点？（ ）

A．存储密度大 B.插入和删除运算方便

C. 获取符合某种条件的元素方便 D.查找运算速度快

8. 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，问 *A* [2][3] (10) 存放在什么位置？ ( 脚注 (10) 表示用 10 进制表示 ,m>3)

A ． 658 B ． 648 C ． 633 D ． 653

9. 在稀疏矩阵的带行指针向量的链接存储中，每个单链表中的结点都具有相同的（ ）。 A ．行号 B ．列号 C ．元素值 D ．非零元素个数

10. 广义表 A= (10,(19,1),((2,1),1) ， 0), 则它的长度为（ ）。

A ． 3 B ． 6 C ． 4 D ． 5

11. 广义表 A= (10,(19,1),((2,1),1) ， 0), 则它的深度为（ ）。

A ． 3 B ． 5 C ． 4 D ． 2

12. 广义表 A= (10,(19,1),((2,1),1) ， 0), 则它的表尾是（ ）。

A.( (19,1),((2,1),1) ) B.(((2,1),1) ， 0)

C. 0 D. ((19,1),((2,1),1) ， 0)

13. 二叉树的第 k层的结点数最多为( ).

A．2 k -1 B.2K+1 C.2K-1 　　　D. 2 k-1

14. 设有 6 个结点的无向图，该图至少应有 ( ) 条边才能确保是一个连通图。

A.5 B.6 C.7 D.8

15. 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 ( ) 。 A． 11 B.35 C. 19 D. 53

16. AOV 网是一种（ ）。

A ．有向图 B ．无向图 C ．无向无环图 D ．有向无环图

17.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 ?( )

A. a， g ，m，q， z B. a， g ，m，z,q

C. g ，m，q，a，z D. g ， m， a，q，z

18. 对线性表进行二分法查找，其前提条件是 ( ).

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

19. 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 ( )

A. 1 ， 2 ， 3 B. 9 ， 5 ， 2 ， 3

C. 9 ， 5 ， 3 D. 9 ， 4 ， 2 ， 3

20. 采用开放定址法处理散列表的冲突时，其平均查找长度（ ）。

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

21. 快速排序在最坏情况下的时间复杂度为（ ）。

A ． O(log 2 n) B ． O(nlog 2 n) C ． 0(n) D ． 0(n 2 )

22. 从二叉 排序 树中查找一个元素时，其时间复杂度大致为 ( )。

A. O(n) B. O(1) C. O(log 2 n) D. O(n 2 )

二、**填空题**

1. 数据的逻辑结构被分为 \_\_\_\_\_\_\_\_\_ 、 \_\_\_\_\_\_\_\_ 、 \_\_\_\_\_\_\_\_\_\_ 和 \_\_\_\_\_\_\_\_\_\_\_ 四种。

2. 数据的物理结构被分为 \_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_\_四种。

3. 一个算法的时间复杂度为 (3 *n* 3 +2000 *n* log 2 *n* +90)/ *n 2* ，其数量级表示为 \_\_\_\_\_\_\_\_ 。

4. 数据结构是指数据及其相互之间的 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 。当结点之间存在 M 对 N （ M ： N ）的联系时，称这种结构为 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 。

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 \_\_\_\_\_\_\_\_\_ ，在表尾插入元素的时间复杂度为 \_\_\_\_\_\_\_\_\_\_\_\_ 。

6. 队列的插入操作是在队列的 \_\_\_\_\_\_\_\_\_ 进行，删除操作是在队列的 \_\_\_\_\_\_\_\_\_\_ 进行。

7. 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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 。

8. 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 \_----\_\_\_\_\_\_\_\_\_ 个，树的深度为 \_\_\_\_\_\_\_\_\_\_\_ ，树的度为 \_\_\_\_\_\_\_\_\_ 。

9. 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 \_\_\_\_\_\_\_\_ ，若它有右孩子，则右孩子结点的编号为 \_\_\_\_\_\_\_\_ ，若它有双亲，则双亲结点的编号为 \_\_\_\_\_\_\_\_ 。

10. 二叉树是指度为 2 的 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 树。一棵结点数为 n 的二叉树，其所有结点的度的总和是 \_\_\_\_\_\_\_\_\_\_\_\_\_ 。

11.对一棵二叉排序树进行中序遍历时，得到的结点序列是一个 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 。对一棵由算术表达式组成的二叉语法树进行后序遍历得到的结点序列是该算术表达式的 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 。

12.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 \_\_\_\_\_\_\_\_\_\_\_\_\_ 个，其中 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 个用于指向孩子， \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 个指针是空闲的。

13.在一个具有 n 个顶点的无向完全图中，包含有 \_\_\_\_\_\_\_\_ 条边，在一个具有 n 个顶点的有向完全图中，包含有 \_\_\_\_\_\_\_\_ 条边。

14.在 n 个带权叶子结点构造出的所有二叉树中，带权路径长度最小的二叉树称为 \_\_\_\_\_\_\_\_ 。 WPL 称为 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 。

15.当向一个大根堆插入一个具有最大值的元素时，需要逐层 \_\_\_\_\_\_\_\_\_ 调整，直到被调整到 \_\_\_\_\_\_\_\_\_\_\_\_ 位置为止。

16.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 \_\_\_\_\_\_\_\_ 。

17.表示图的三种常用的存储结构为 \_\_\_\_\_\_\_\_\_\_\_\_\_ 、 \_\_\_\_\_\_\_\_\_\_\_\_ 和 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 。

18.对于线性表（ 70 ， 34 ， 55 ， 23 ， 65 ， 41 ， 20 ）进行散列存储时，若选用 H （ K ） =K %7 作为散列函数，则散列地址为 0 的元素有 \_\_\_\_\_\_\_\_ 个，散列地址为 6 的有 \_\_\_\_\_\_\_ 个。

19.在堆排序的过程中，对任一分支结点进行筛运算的时间复杂度为 \_\_\_\_\_\_\_\_ ，整个堆排序过程的时间复杂度为 \_\_\_\_\_\_\_\_ 。

20.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 \_\_\_\_\_\_\_\_ 。

三、**运算题**

1.写出下列中缀表达式的后缀形式：

(1). 3X/(Y-2)+1

(2). 2+X\*(Y+3)

2. 已知一棵二叉树的前序遍历的结果是 ABKCDFGHIJ, 中序遍历的结果是 KBCDAFHIGJ, 试画出这棵二叉树。

3. 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。

4. 已知一个图的顶点集 V 为： V={1,2,3,4,5,6,7} ；

其共有10条边。该图用如下边集数组存储：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 起点 | 1 | 2 | 2 | 5 | 5 | 2 | 2 | 6 | 1 | 3 |
| 终点 | 6 | 4 | 5 | 4 | 7 | 6 | 7 | 7 | 7 | 5 |
| 权 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 4 | 5 | 7 |

试用克鲁斯卡尔算法 **依次** 求出该图的最小生成树中所得到的各条边及权值。

5.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。

6.判断以下序列是否是小根堆 ? 如果不是 , 将它调整为小根堆。

**（ 1 ）** {12,65,33,70,24,56,48,92,86,33}

**（ 2 ） {** 05, 23, 20, 28, 40, 38, 29, 61, 35, 76, 47, 100 **}**

7.画出向小根堆中加入数据 4, 2, 5, 8, 3, 6, 10, 1 时，每加入一个数据后堆的变化情况。

8.假定一组元素的关键字为（ 50 ， 43 ， 56 ， 34 ， 86 ， 78 ， 54 ， 04 ），哈希函数为 H(k)=k%7 ，采用线性探测法解决冲突，试：

(1).画出依序构造出的、表长为 8 的哈希表存储结构示意图；

(2).求给每个元素对应的查找长度和平均查找长度。

9.假定一组元素的关键字为（ 45 ， 23 ， 56 ， 34 ， 26 ， 78 ， 90 ， 54 ， 31 ， 21 ， 67 ），采用快速排序，给出第一趟排序后的结果。

10.试给出小根堆 **{** 05, 23, 20, 28, 40, 38, 29, 61, 35, 76, 47, 100 **}** 前两趟排序的结果。

四、**阅读算法**

1. void p1( Seq Stack &S)

{ int i, n, a[100] ；

n=0 ；

while(!EmptyStack(S))

{n++ ； Pop(S, a[n]) ； }

for(i=1 ； i<=n ； i++)

Push(S, a[i]) ；

}

该算法的功能是：

2. int Prime(int n)

{

int i=1 ；

int x=(int) sqrt(n) ；

while (++i<=x)

if (n%i==0) break ；

if (i>x) return 1 ；

else return 0 ；

}

3. 指出该算法的功能；

4. 该算法的时间复杂度是多少？

5. int ListLength(LNode\* H)

{

if（H->next==NULL）

return 0；

else return 1+ListLength(H->next)；

}

（ 1 ）指出该算法的功能；

（ 2 ）该算法的时间复杂度是多少（假设表长为 n ）？

void ABC (BTreeNode \*BT, int &c1,int &c2) {

if (BT !=NULL ) {

ABC(BT->lchild,c1,c2) ；

c1++ ；

if (BT->lchild= =NULL&&BT->rchild= =NULL) c2++ ；

ABC(BT->rchild,c1,c2) ；

}//if

}

该函数执行的功能是什么？

6. int BTreeDepth (BTreeNode \* BT)

{

if( BT==NULL)

return 0；

else

{

int depl= BTreeDepth(BT->lchild)；

int depr= BTreeDepth(BT->rchild)；

if(depl>depr)

return depl+1；

else

return depr+1；

}

}

该函数执行的功能是什么？

五、**算法填空**

1. 向单链表的末尾添加一个元素的算法。

void InsertRear(LNode\*& HL,const ElemType& item)

{

LNode\* newptr；

newptr=new LNode；

if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

{

cerr<<"Memory allocation failare!"<<endl；

exit(1)；

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_=item；

newptr->next=NULL；

if (HL==NULL)

HL=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_；

else{

LNode\* p=HL；

while (p->next!=NULL)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_；

p->next=newptr；

}

}

2. 折半查找的递归算法

int BitSearch(ElemType D[],int low,int high,KeyType key)

{

if(low<=high)

{ int mid=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_；

if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) return mid；/\*查找成功，返回元素的下标\*/

else if(key<D[mid].key)

return BitSearch(D,low,mid-1,key)； /\*在前半段继续查找\*/

else

return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_； /\*在后半段继续查找\*/

}

else return -1； /\*查找失败，返回-1\*/

}

3. 二叉排序树的递归查找算法 :

bool Find(BTreeNode\* BST,ElemType& item)

{

if (BST==NULL)

return false； //查找失败

else {

if (item==BST->data){

item=BST->data；//查找成功

return ---------\_\_\_\_\_\_\_\_\_\_\_；}

else if(item<BST->data)

return Find(\_\_\_\_\_\_\_\_\_\_\_\_\_\_,item)；

else return Find(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,item)；

}//if

}

4. 以广义表的形式输出二叉树的递归算法：

void PrintBTree (BTreeNode \* BT)

{

if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

{

cout<<BT->data；

if (BT->lchild!=NULL||BT->rchild!=NULL)

{

cout<<'('；

PrintBTree(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)；/\*输出左子树\*/

if (BT->rchild!=NULL)

cout<<','；

PrintBTree(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)；/\*输出右子树\*/

cout<<')'；

}

}

}

参考答案

**一、单选题**

1.B 2.B 3.D 4.B 5.A 6.C 7.A 8.D 9.A 10.C 11.A 12.D 13.D 14.A 15.B 16.D 17.B 18.C 19.D 20.B 21.D 22.C

**二、填空题**

1. 集合结构 线性结构 树结构 图结构

2. 顺序存储结构 链式存储结构 索引存储结构 散列存储结构

3. O(n)

4. 联系 图（或图结构）

5. O(n) O(1)

6. 尾 头

7. top= =0

8. 7 3 2

9. 2i 2i+1 ? i/2 ? （或 i/2）

10. 有序 n-1

11.有序序列 后缀表达式（或逆波兰式）

12.2n n-1 n+1

13. n(n-1)/2 n(n-1)

14.赫夫曼树 带权路径长度

15.向上 根

16.2.9

17.邻接矩阵 邻接表 边集数组

18.1 4

19.O(log 2 n) O(nlog 2 n)

20.n/m

**三、运算题**

1. (1) 3 X \* Y 2 - / 1 +

(2) 2 X Y 3 + \* +

2. 略

3. 普里姆算法从顶点 1 出发得到最小生成树为：

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

4. 用克鲁斯卡尔 算法得到的最小生成树为：

(1,6)1, (2,4)1, (2,5)2, (5,7)2, (2,6)3, (3,5)7

5. 拓朴排序为： 4 3 6 5 7 2 1

6. (1) 不是小根堆。调整为： **{** 12,24,33,65,33,56,48,92,86,70 **}**

(2) 是小根堆。

7. 略

8.（ 1 ）哈希表如下：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 56 | 50 | 43 | 86 | 78 | 54 | 34 | 04 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

（ 2 ） 50 ， 43 ， 56 ， 34 ， 86 ， 78 ， 54 ， 04 对应的查找长度分别为：

1 ， 2 ， 1 ， 1 ， 2 ， 4 ， 1 ， 4

平均查找长度为：（ 1+2+1+1+2+4+1+4 ） /8=2

9. 快速排序，给出第一趟排序后的结果如下：

[31 23 21 34 26] 45 [90 54 78 56 67]

10. 小根堆前两趟排序的结果如下：

20 23 29 28 40 38 100 61 35 76 47 05

23 28 29 35 40 38 100 61 47 46 20 05

**四、阅读算法**

1. 该算法的功能是：利用数组 a 将 S 栈中的数据倒序。

2. (1) 判断 n 是否是素数（或质数）

（ 2 ） O （ ）

3. （ 1 ）该递归函数是求出 H 指向的链表长度并由 ListLength带回 。

（ 2 ）该算法的时间复杂度是 O （ n ）。

4. 该函数的功能是：求出由 BT 所指向的二叉树的结点总数和叶子总数。

5. 该函数的功能是：求出由 BT 所指向的二叉树的深度。

**五、算法填空**

1. newptr==NULL newptr->=data newptr p=p->next

2. (low+high)/2 key==D[mid].key BitSearch(D,mid+1,high,key)

3. true BST->left BST->right

4. BT!=NULL BT->lchild BT->rchild