**□开卷**

**🗹闭卷**

**□公共课**

**🗹专业课**

**课程名称：** 数据结构 **课程类别 考核形式**

**学生类别\_\_\_\_\_\_\_\_\_\_\_考试日期\_\_\_\_\_\_\_\_\_\_\_\_\_\_学生所在院系\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**学号\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_姓名\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_任课教师\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 题 号 | 一 | 二 | 三 | 四 | 五 | 总 分 |
| 得 分 |  |  |  |  |  |  |

|  |  |
| --- | --- |
| 得分 | 评卷人 |
|  |  |

### 一、单项选择题（每题1分，共15分）

1. 线性表采用链式存储时，其地址\_\_\_D\_\_\_\_\_。

A．必须是连续的 B．一定是不连续的

C．部分地址必须是连续的 D．连续与否均可以

2. 在一个链队列中，假定front和rear分别为队首和队尾指针，则删除一个结点的操作为\_\_\_A\_\_\_\_\_。

A．front=front->next 　 B．rear=rear->next

C．rear=front->next 　D．front=rear->next

3. 两个字符串相等的条件是（ D ）。

A.两串的长度相等

B.两串包含的字符相同

C.两串的长度相等，并且两串包含的字符相同

D.两串的长度相等，并且对应位置上的字符相同

4. 稀疏矩阵一般的压缩存储方法有两种，即（C ）。

A.二维数组和三维数组 B.三元组和散列

C.三元组和十字链表 D.散列和十字链表

5. 在\_C\_\_\_\_\_运算中，使用顺序表比链表好。

A．插入   B．删除

C．根据序号查找   D．根据元素值查找

6. 已知一棵完全二叉树的结点总数为9个，则最后一层的结点数为（ B ）。

A. 1 B. 2 C. 3 D. 4

7. 由权值分别为3,8,6,2,5的叶子结点生成一棵哈夫曼树，它的带权路径长度为（ D ）。

A. 24 B. 48 C. 72 D. 53

8. 设n , m 为一棵二叉树上的两个结点，在中序遍历序列中n在m前的条件是（ B ）。

A. n在m右方 B. n在m 左方

C. n是m的祖先 D. n是m的子孙

9. 在一个具有n个顶点的有向完全图中，所含的边数为( B )。

A. n B. n(n-1) C. n(n-1)/2 D. n(n+1)/2

10. 若一个图的边集为{<1,2>,<1,4>,<2,5>,<3,1>,<3,5>,<4,3>}，则从顶点1开始对该图进行深度优先搜索，得到的顶点序列可能为( A )。

A. 1,2,5,4,3 B. 1,2,3,4,5

C. 1,2,5,3,4 D. 1,4,3,2,5

11. 由一个具有n个顶点的连通图生成的最小生成树中，具有( B )条边。

A. n B. n-1 C. n+1 D. 2×n

12. 对于长度为18的顺序存储的有序表，若采用折半查找，则查找第15个元素的比较次数为(　B　)。

A. 3 B. 4 C. 5 D. 6

13. 若根据查找表(23,44,36,48,52,73,64,58)建立哈希表，采用h(K)=K%13计算哈希地址，则元素64的哈希地址为(　C　)。

A. 4 B. 8 C. 12 D. 13

14. 对下列四个序列进行快速排序，各以第一个元素为基准进行第一次划分，则在该次划分过程中需要移动元素次数最多的序列为（ D ）。

A. 1, 3, 5, 7, 9 B. 9, 7, 5, 3, 1

C. 5, 3, 1, 7, 9 D. 5, 7, 9, 1, 3

15. 假定对元素序列（7, 3, 5, 9, 1, 12）进行堆排序，并且采用小根堆，则由初始数据构成的初始堆为（ B ）。

A. 1, 3, 5, 7, 9, 12 B. 1, 3, 5, 9, 7, 12

C. 1, 5, 3, 7, 9, 12 D. 1, 5, 3, 9, 12, 7

|  |  |
| --- | --- |
| 得分 | 评卷人 |
|  |  |

### **二、填空题 （每空1分，共25分）**

1. 下面程序段的时间复杂度是\_\_\_\_\_ O()\_\_\_\_\_\_\_\_\_\_\_\_\_。

i=s=0;

while(s<n)

{ i++;

s+=i;

}

2. 在双向链表中，每个结点含有两个指针域，一个指向\_前趋\_结点，另一个指向\_后继\_结点。

3. 根据线性表的链式存储结构中每个结点所含指针的个数，链表可分为\_单链表\_和\_双链表\_；而根据指针的联接方式，链表又可分为\_\_非循环链表\_\_和\_\_循环链表\_\_。

4. 空串是指\_不含任何字符的串\_\_，空格串是指\_\_仅含空格字符的字符串\_\_\_\_。

5. 一个稀疏矩阵为 ，则对应的三元组线性表为\_\_((0，2，2)，(1，0，3)，(2，2，-1)，(2，3，5))\_。

6. 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[5，8]的存储首地址为\_\_ BA+180\_\_。

7. 线索链表中的rtag域值为\_\_1\_\_时，表示该结点无右孩子，此时\_ RChild \_域为指向该结点后继线索的指针。

8. 一个图的边集为{(a,c),(a,e),(b,e),(c,d),(d,e)}，从顶点a出发进行深度优先搜索遍历得到的顶点序列为\_ acdeb \_，从顶点a出发进行广度优先搜索遍历得到的顶点序列为\_ acedb \_。(答案不唯一)

9. 假定一个有向图的顶点集为{a,b,c,d,e,f}，边集为{<a,c>, <a,e>, <c,f>, <d,c>, <e,b>, <e,d>}，则出度为0的顶点个数为\_\_2\_\_，入度为1的顶点个数为\_\_\_4\_\_。

10. 假定对长度n=50的有序表进行折半查找，则对应的判定树高度为\_6\_，最后一层的结点数为\_19\_。

11. 假定对线性表(38,25,74,52,48)进行哈希存储，采用H(K)=K%7作为哈希函数，采用线性探测法处理冲突，则在建立哈希表的过程中，将会碰到\_\_5\_\_\_\_\_\_次存储冲突。

12. 假定一个顺序表的长度为40，并假定查找每个元素的概率都相同，则在查找成功情况下的平均查找长度\_\_20.5\_\_\_\_\_\_，在查找不成功情况下的平均查找长度\_\_41\_\_\_\_\_\_。

13. 假定一组记录为(46,79,56,38,40,84)，则利用堆排序方法建立的初始小根堆为\_\_\_38,40,56,79,46,84\_\_\_\_。

14. 假定一组记录为(46,79,56,38,40,80)，对其进行快速排序的第一次划分后的结果为\_\_[40 38] 46 [56 79 80]\_\_。

1. 15. 在所有排序方法中，\_\_堆排序\_方法使数据的组织采用的是完全二叉树的结构。

|  |  |
| --- | --- |
| 得分 | 评卷人 |
|  |  |

### 简答题（每题3分，共15分）

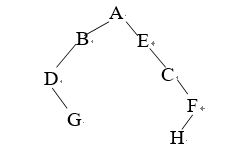
1. 现有5个数据元素，其入栈次序为：A，B，C，D，E。请问：在各种可能的出栈次序中，以元素C，D最先出栈（即C第一个且D第二个出栈）的次序有哪几个？

三个：CDEBA，CDBEA，CDBAE

2. 一颗共有n个结点（编号从1至n）并采用顺序存储结构的完全二叉树，求该树中编号最小的叶子结点的编号。

根据完全二叉树的性质，最后一个结点（编号为n）的双亲结点的编号是n/2，这是最后一个分枝结点，在它之后是第一个终端（叶子）结点，故编号最小的叶子结点的编号是：n/2+1

3. 设一棵二叉树的先序序列为ABDGECFH，中序序列为：DGBAECHF。试画出该二叉树。



4. 已知图G的邻接矩阵如下所示，根据Prim算法，求图G从顶点①出发的最小生成树，要求写出其每一步生成过程。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | ① | ② | ③ | ④ | ⑤ | ⑥ |  |
|  |  | | | | | | | |



5. 给出稀疏矩阵十字链表的数据结构定义。

typedef struct OLNode {

int i, j;

ElemType e;

struct OLNode \*down, \*right;

}OLNode, \*OLink;

typedef struct {

Olink \*rhead, \*chead;

int mu, nu, tu;

}CrossList;

|  |  |
| --- | --- |
| 得分 | 评卷人 |
|  |  |

### 程序阅读题（每题5分，共5分）

1. 写出下面算法的功能。

int abc\_Bitree(Bitree T) {

InitQueue(Q); //建立工作队列

flag=0;

EnQueue(Q,T); // 入队

while(!QueueEmpty(Q)) {

DeQueue(Q,p); // 出队

if(!p)

flag=1;

else if(flag)

return 0;

else {

EnQueue(Q,p->lchild);

EnQueue(Q,p->rchild);

}

}//while

return 1;

}

该算法功能是判断一颗树是否为完全二叉树

|  |  |
| --- | --- |
| 得分 | 评卷人 |
|  |  |

### 五．算法和应用题（每题10分，共40分）

1. 图(a)、(b)所示为一个有向网图及其带权邻接矩阵，要求对有向图采用Dijkstra算法，求从V0 到其余各顶点的最短路径。

(a) 有向带权图

V1

V0

V5

V4

V3

V2

5

10

60

30

100

50

20

10

∞ ∞ 10 ∞ 30 100

∞ ∞ 5 ∞ ∞ ∞

∞ ∞ ∞ 50 ∞ ∞

∞ ∞ ∞ ∞ ∞ 10

∞ ∞ ∞ 20 ∞ 60

∞ ∞ ∞ ∞ ∞ ∞

(b) 带权邻接矩阵

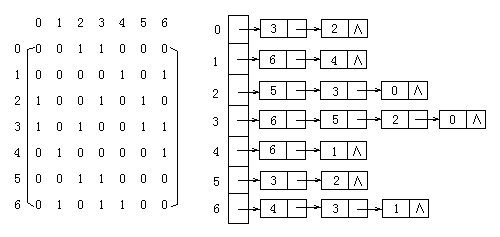
图 有向带权图及其邻接矩阵

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 终点 | 从v0到各终点的D值和最短路径的求解过程 | | | | |  |
| i=1 | i=2 | i=3 | i=4 | i=5 |  |
| V1 | ∞ | ∞ | ∞ | ∞ | ∞  无 |  |
| V2 | 10 (v0,v2) |  |  |  |  |  |
|  |
| V3 | ∞ | 60 (v0,v2,v3) | 50  (v0,v4,v3) |  |  |  |
|  |
| V4 | 30  (v0,v4) | 30  (v0,v4) |  |  |  |  |
|  |
| V5 | 100  (v0,v5) | 100  (v0,v5) | 90  (v0,v4,v5) | 60 (v0,v4,v3,v5) |  |  |
|  |
| Vj | V2 | V4 | V3 | V5 |  |  |
| S | {v0,v2} | {v0,v2,v4} | {v0,v2,v3,v4} | {v0,v2,v3,v4,v5} |  |  |

2. 已知一个无向图的邻接矩阵如图(a)所示，试写出从顶点0出发分别进行深度优先和广度优先搜索遍历得到的顶点序列。

已知一个无向图的邻接表如图(b)所示，试写出从顶点0出发分别进行深度优先和广度优先搜索遍历得到的顶点序列。

|  |
| --- |
|  |



图(a) 图(b)

(a) 深度优先搜索序列：0,2,3,5,6,1,4

广度优先搜索序列：0,2,3,5,6,1,4

(b) 深度优先搜索序列：0,3,6,4,1,5,2

广度优先搜索序列：0,3,2,6,5,4,1

3. 给定一个带有头结点 head 的非空单链表，试设计一个高效算法返回链表的中间结点。如果有两个中间结点，则返回第二个中间结点。

a. 设计算法思路及步骤；

b. 实现算法代码；

c. 给出算法时间复杂度。

方法一：输出到数组

按顺序将每个结点放入数组 A 中。然后中间结点就是 A[A.Length/2]，因为我们可以通过索引检索每个结点。

ListNode\* middleNode(ListNode\* head) {

vector<ListNode\*> A = {head};

while (A.back()->next != NULL)

A.push\_back(A.back()->next);

return A[A.size() / 2];

}

时间复杂度O(N)

方法二：快慢指针法

当用慢指针 slow 遍历列表时，让另一个指针 fast 的速度是它的两倍。当 fast 到达列表的末尾时，slow 必然位于中间。

ListNode\* middleNode(ListNode\* head) {

ListNode\* slow = head;

ListNode\* fast = head;

while (fast != NULL && fast->next != NULL) {

slow = slow->next;

fast = fast->next->next;

}

return slow;

}

时间复杂度O(N)

4. 试比较插入排序、简单选择排序、冒泡排序、快速排序、归并排序、堆排序的时间复杂度。如果有n个无序元素，要求按非递减次序排序，但只想得到前面长度为k的部分序列，其中k << n，最好采用上述哪种排序方法？为什么？

快速排序、插入排序、归并排序等时间性能好的排序，都要等到最后才能确定各元素位置。一趟排序后简单选择排序和冒泡排序可以选出一个最大（或最小）元素，并加入到已有的有序子序列中，其时间复杂度是O(n2)。而堆排序，在未结束全部排序前，可以有部分排序结果。建立堆后，堆顶元素就是最大（或最小，视大堆或小堆而定）元素，然后，调堆又选出次大（小）元素。凡要求在n个元素中选出k(k<<n，k>2)个最大（或最小）元素，一般均使用堆排序。因为堆排序建堆比较次数至多不超过4n，对深度为k的堆，在调堆算法中进行的关键字的比较次数至多为2(k-1)次。