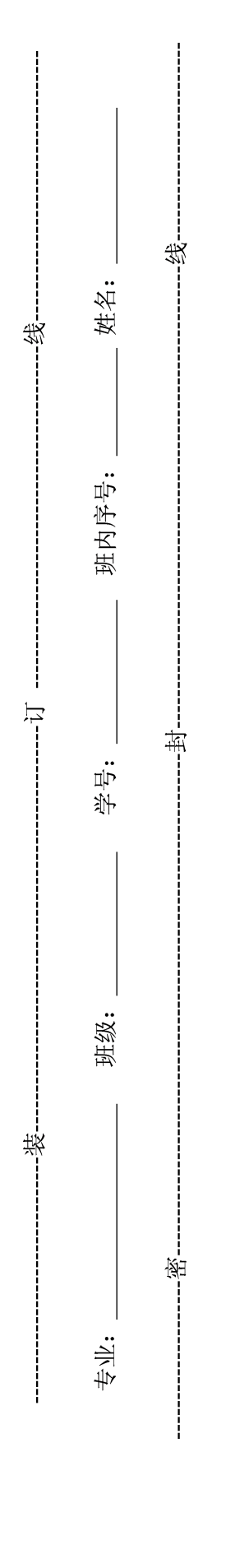
**北京邮电大学2019——2020学年第一学期**

**《数据结构》**期末考试试题（**B卷**）

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 考 试 注 意 事 项 | 一、学生参加考试须带学生证或学院证明，未带者不准进入考场。学生必须按照监考教师指定座位就坐。  二、书本、参考资料、书包等物品一律放到考场指定位置。  三、学生不得另行携带、使用稿纸，要遵守《北京邮电大学考场规则》，有考场违纪或作弊行为者，按相应规定严肃处理。  四、学生必须将答题内容做在试题答卷上，做在试题及草稿纸上一律无效。  五、答卷应字迹清楚、语义确切。  六、算法应说明基本思路，应对主要数据类型、变量给出说明，所写算法应结构清晰、简明易懂，应加上必要的注释。 | | | | | | | | | | |
| 考试  课程 | 数据结构 | | | | 考试时间 | | | 2020年 06 月 29 日 | | | |
| 题号 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 九 | 十 | 总分 |
| 满分 | 20 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |  | 100 |
| 得分 |  |  |  |  |  |  |  |  |  |  |  |
| 阅卷  教师 |  |  |  |  |  |  |  |  |  |  |  |

**一、选择题（每题2分，共20分）**

1．下列函数的时间复杂度为（ ）。

int A( int n ) {

int i=0, sum = 0;

while (sum < n) sum += ++i;

return i; }

**A．**O(log2n) **B．**O(n1/2) **C．**O(n) **D．**O(nlog2n)

2．带头结点的双循环链表，删除指针p所指结点的正确语序是（ ）。

**A．**p->next->prev = p->prev; p->prev->next = p->prev; free(p);

**B．**p->next->prev = p->next; p->prev->next = p->next; free(p);

**C．**p->next->prev = p->next; p->prev->next = p->prev; free(p);

**D．**p->next->prev = p->prev; p->prev->next = p->next; free(p);

3．设栈S和队列Q的初始状态均为空，元素abcdefg依次进入栈S，若每个元素出栈后立即进入队列Q，且7个元素的出队顺序为bdcfeag，则栈S的容量至少为（ ）。

**A．**1 **B．**2 **C．**3 **D．**4

4．字符串‘pqpqppqpq’ 的nextval 为（ ）。

**A．**(0,1,0,1,0,4,1,0,1) **B．**(0,1,0,1,0,2,1,0,1)

**C．**(0,1,0,1,0,0,0,1,1) **D．**(0,1,0,1,0,1,0,1,1 )

5．适合于压缩存储稀疏矩阵的两种存储结构是（ ）。

**A．**三元组表和十字链表 **B．**三元组表和邻接矩阵

**C．**十字链表和二叉链表 **D．**邻接矩阵和十字链表

6．一颗完全二叉树有768个结点，则该二叉树中叶子的数目是（ ）。

**A．**383 **B．**384 **C．**385 **D．**无法确定的

7．下列选项给出的是从根到两个叶子结点路径上的结点权值序列，能属于同一颗哈夫曼树的是（ ）。

**A．**24、10、5和24、10、7 **B．**24、10、5和24、12、7

**C．**24、10、10和24、14、11 **D．**24、10、5和24、14、6

8．在有n个顶点的有向图中，每个顶点的度最大可达（ ）。

**A．**n **B．**n-1 **C．**2n **D．**2n-2

9．对有2500个记录的表进行分块查找，则理想的块长为（ ）。

**A．**50 **B．**51 **C．**500 **D．**501

10．下列排序算法中，对初始状态为递增序列的表按递增顺序排序，最省时间的是（ ）。

**A．**快速排序 **B．**起泡排序 **C．**归并排序 **D．**简单选择排序

**二、判断题（每题1分，共10分）**

1．（ ）数据的逻辑结构是指数据的各数据项之间的逻辑关系。

2．（ ）线性表采用链表存储时，结点和结点内部的存储空间可以是不连续的。

3．（ ）广度优先遍历算法通常借助队列来实现。

4．（ ）串是一种数据对象和操作都特殊的线性表。

5．（ ）若一个广义表的表尾为空表，则此广义表亦为空表。

6．（ ）用一维数组存储二叉树时，总是以先序遍历顺序存储结点。

7．（ ）无向图的邻接矩阵一定是对称矩阵，有向图的邻接矩阵不一定是非对称矩阵。

8．（ ）m阶B-树所有叶子都在同一层上。

9．（ ）快速排序算法不是稳定排序算法，其空间复杂度也不是O(1)。

10．（ ）外部排序是把外存文件调入内存，可利用内部排序的方法进行排序,因此排序所花的时间取决于内部排序的时间。

**三、**已知某二叉树的先序序列和中序序列如下所示，画出这颗二叉树及其对应的森林（树）。**（10分）**

**先序序列：53 17 09 45 23 78 94 88**

**中序序列：09 17 23 45 53 78 88 94**

**四、**请看下边的有向无环图。**（10分）**

（1）画出它的逆邻接表；（3分）

（2）画出它的邻接矩阵；（3分）

（3）从V1出发按照上述邻接矩阵的存储结构，写出深度优先遍历的次序；（4分）



**五、**已知散列表的地址空间为A[0..10]，散列函数H（k）=（3k+5） mod 11，采用线性探测再散列法处理冲突。**（10分）**

（1）请将关键字序列{25, 17, 92, 51, 33, 29, 83, 123, 42, 105}依次插入到下面的散列表中，给出下表中各空的值；

（2）并计算出在等概率情况下查找成功和不成功时的平均查找长度。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 关键字 | 25 | 17 | 92 | 51 | 33 | 29 | 83 | 123 | 42 | 105 |
| H(k) |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 散列地址 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 关键字 | |  |  |  |  |  |  |  |  |  |  |  |
| 比较次数 | 查找成功 |  |  |  |  |  |  |  |  |  |  |  |
| 查找失败 |  |  |  |  |  |  |  |  |  |  |  |

ASL成功 =

ASL不成功 =

**六、**给定一组十个关键字的集合：{ 41H，94H，11H，A6H，23H，53H，F7H，28H，88H，75H}，每个关键字是两位的一个十六进制数。对关键字进行快速排序。请回答：**（10分）**

（1）描述快速排序的处理过程（4分）

（2）写出快速排序第一趟和第二趟的结果（6分）

**七、**一棵有n个节点的完全二叉树，采用顺序存储的方式存于数组a[n]中，其中每个节点存储的都是正整数，编写一个算法判断这棵完全二叉树是否是一个大根堆，将判断结果返回，1代表是，0代表不是。**（10分）**

**int IsBigRootHeap ( int a[ ])**

// 判断a[n]存储的是否是大根堆，函数返回值1代表是，0代表不是；

**八、**已知二叉排序树的根指针及其中一个结点的值（树中一定存在该结点），请编写算法，判断该结点是否叶子结点，是返回1，否则返回0。**（10分）**

**typedef struct node**

**{**

**char data;**

**node \*lc, rc;**

**} bitptr;**

**int Leaf ( bitptr &t, char x ) // t为二叉排序树根指针，x为某结点值**

**九、**假设包含n个顶点的有向加权图（顶点编号从1到n）采用邻接矩阵存储，其邻接矩阵和邻接表的存储结构定义在下面给出。请编制算法将图的存储结构由邻接矩阵(Adjmatrix)转换为邻接表(Adjlist) 。在邻接矩阵中定义一个最大权值（MAXINT）表示无弧相连。**（10分）**

**邻接矩阵结构体定义：**

**typedef char vtype;**

**typedef  double  Adjmatrix[vtxnum][vtxnum]; //邻接矩阵，vtxnum为顶点个数**

**typedef  vtype Adjvexs[vtxnum];//顶点数组，vtxnum为顶点个数**

**邻接表结构体定义：**

**typedef  struct { //邻接表边表节点；**

**int       adjvex;**

**double      weight;**

**arcnode   \*nextarc;**

**} arcnode;**

**typedef  struct {  //邻接表顶点**

**vtype   vexdata;     //顶点相关信息**

**arcnode    \*firstarc;**

**} vexnode;**

**typedef  vexnode Adjlist[vtxnum]; //邻接表，vtxnum为顶点个数**

**下面给出了一个不完整的转换算法，请添加算法描述语句，补充完成算法。**

**void  Change(Adjvexs v, Adjmatrix  m,  Adjlist  &adj,  int   n )**

**//v为邻接矩阵顶点数组，m为邻接矩阵，adj为邻接表，n为顶点数**

**{**

#define MAXINT 32767;

arcnode \*p,\*q;

for (int i=1;i<=n; i++) v[i].firstarc = NULL; // 邻接表初始化；

for (int i =1; i<=n; i++) { // 按行遍历邻接矩阵

FIRSTARC=TRUE

for (int j=1; j<=n; j++) { // 访问某行的各列

if m[i,j] < MAXINT { // 存在一条弧，添加到邻接表结点i的队列中，j是邻接点。

p= new arcnode; // 新的结点

***// 请补充完成算法中所缺少的语句，并且语句带有注释说明其功能。***

}; // 存在一条弧的处理完成。

} // j循环

} // i循环

} 函数结束