**2015年全国硕士研究生统一入学考试自命题试题（B卷）**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

学科、专业名称：计算机科学与技术、软件工程

研究方向：计算机系统结构081201，计算机软件与理论081202，计算机应用技术081203，软件工程083500，计算机技术(专业学位) 085211，软件工程(专业学位) 085212

考试科目名称及代码：数据结构830

**一. 单项选择题(每题2分，共30分)**

1．线性表采用链式存储时,其地址( **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.连续与否均可以

2．若有一个栈的输入序列是1，2，3，…，n，输出序列的第一个元素是n,则第i个输出元素是（ **C** ）。

A. n-i B. n-i-1 C. n-i+1 D. 不确定

3. 已知单链表上一结点的指针为p，则删除该结点后继的正确操作语句是（ **C** ）。

A. s= p->next; p=p->next; free(s); B．p=p->next; free(p);

C. s= p->next; p->next=s->next; free(s); D．p=p->next; free(p->next);

4. 若使用邻接矩阵表示某有向图，则矩阵中非零元素的个数等于（ **B** ）。

A. 图中顶点的数目 B. 图中边的数目

C. 图中边的数目的两倍 D. 无法确定

5. 下列哪种排序需要的附加存储开销最大( **C** )。

A．快速排序 B.堆排序 C.归并排序 D.插入排序

6. 下面哪一方法可以判断出一个有向图是否有环（即回路）（ **A** ）。

A．拓扑排序 B. 求最短路径 C. 求最小生成树 D. 广度优先遍历

7. 具有n个顶点的无向图至少应有（ ）条边才能确保是一个连通图.

A．n-1 B．n C．n+1 D．2n

8. 对线性表进行折半查找时,要求线性表必须 ( **B** ) 。

A.以顺序方式存储 B. 以顺序方式存储,且结点按关键字有序排序

C. 以链接方式存储 D.以链接方式存储,且结点按关键字有序排序

9．若使用二叉链表作为树的存储结构，在有n个结点的二叉链表中非空的链域的个数为( **A** ) 。

A. n-1 B. 2n-1 C. n+1 D. 2n+1

10．在内部排序中,排序时不稳定的有( **C**  ) 。

A. 插入排序 B. 冒泡排序 C. 快速排序 D.归并排序

11. 一个具有500 个结点的完全二叉树具有一个孩子的结点个数最多为（ **A**  ）。

A．1 B．250 C．0 D．249

12. 从未排序序列中取出一个元素，并将其依次插入已排序序列的方法，称为 （ **C** ）。

Ａ. 希尔排序 Ｂ. 归并排序 Ｃ. 插入排序 Ｄ. 选择排序

13. 如果希望对二叉排序树遍历的结果是升序的，应采用（ **B** ）遍历方法。

A．先序 B．中序 C．后序 D．层次

14. 队列操作的原则是（ **A** ）

A．先进先出 B．后进先出 C．只能进行插入 D．只能进行删除

15. 在用邻接表表示有向图的情况下,假设n为图的顶点数目, e为图的边数目,建立图的算法的时间复杂度为( **A** )。

A．O(n+e) B．O(n2) C．O(n×e) D．O(n3)

**二．填空题(每空2分，共20分)**

1．循环链表的主要优点是 **可以方便的从尾部增加元素且可以方便的删除表头元素** 。

2．根据数据元素之间关系的不同特性, 基本逻辑结构分为 **集合** 、 线性结构 、 树形结构和 **图状结构** 四种。

3．对一棵完全二叉树中按照从上到下、从左到右的顺序从1开始顺序编号，则编号为11双亲结点的编号为 **5** ，编号为10的结点的左孩子结点（若存在）的编号为 **20** 。

4．下面程序段的时间复杂度是  **O(n^2)** 。

S=0;

for( i=0;i<N; i++)

for(j=0; j<2N+1; j++)

S++;

5．深度为h 的满二叉树共有 **2^h-1** 个结点。

6．一棵m阶非空B-树，每个结点最多有 **m** 棵子树，除根之外的所有非终端结点至少有 **[m/2](向上取整)** 棵子树。

7．在单链表中，若要在指针p所指结点后插入指针s所指结点，则需要执行下列两条语句： **s->next = p->next;p->next=s;** 。

**三．判断题（每题1分，共10分，正确的选t，错误的选f）**

1．数据元素是数据的基本单位。（ **√** ）

2．线性表中的每一个元素都有一个前驱和后继元素。（ **×** ）

3．当向二叉排序树中插入一个结点，则该结点一定成为叶子结点。（ **√** ）

4．带权无向图的最小生成树是唯一的。( **×** )

5．设有序的关键字序列是（2，5，8，9，12，14，16，18，20，22，25），当用折半查找方法查找关键字22时，需经3次比较运算。（ **√** ）

6．一棵m阶B+树中每个结点最多有m个关键码，最少有2个关键码。( **×** )

7．根据拓扑排序结果可以判断一个有向图中是否存在环路。 （ **√** ）

8．图的深度优先搜索中可以采用栈来暂存刚访问过的顶点。 （ **√** ）

9．已知一颗树的先序序列和后序序列，一定能构造出该树。 （ **×** ）

10．存储图的邻接表中，邻接表的大小不但与图的顶点个数有关，而且与图的边数也有关。（ **√** ）

**四. 简答题（45分）**

1. 简述下列算法的功能。（6分）

typedef struct BiTNode { TElemType data; struct BiTNode \*lchild, \*rchild; } BiTNode, \*BiTree;void Process(BiTree T){

IniStack(S);

P=T;

while (P||!StackEmpty(S)) {

if (P) {

push(&S, P); P=P->lchild;

}

else {

pop(&S, P); printf(P->data);

P=P->rchild;

}

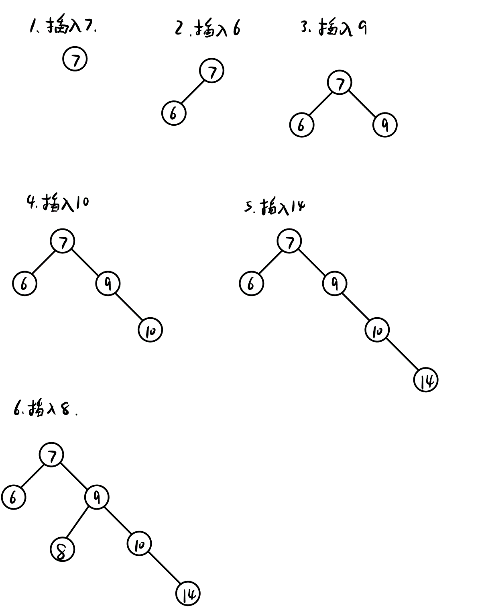
}

}

**答：该算法的功能是实现了对链表存储的二叉树进行非递归先序遍历。**

1. 假设表中关键字序列为（7,6,9,10，14,8），将关键字依次插入一棵初始为空的二叉排序树。画出二叉排序树的生成过程。（10 分）

**答：**



1. 关键字序列 T=(63，55，48，37，20，90，84，32），对其从小到大排序，以第一个关键字为枢轴（支点），写出快速排序具体实现过程（10分）。

**答：第一趟：32，55，48，37，20，63，84，90**

**第二趟：20，32，48，37，55，63，84，90**

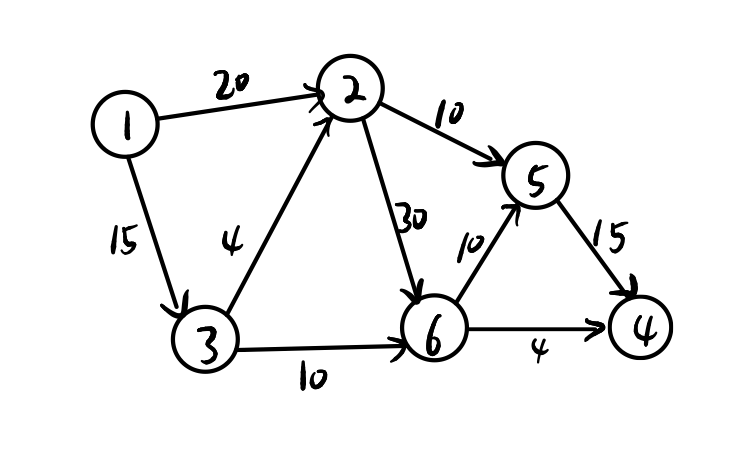
**第三趟：20，32，37，48，55，63，84，90**

1. 一个有六个顶点{v1,v2,v3,v4,v5,v6}的网的邻接矩阵如图1所示，解答下列问题：
   1. 画出该网（2分）
   2. 能否写出一种拓扑排序序列，若可以，写出一种拓扑排序序列（2分）

（3） 求出从顶点v1到其他各顶点之间的最短路径，并写出计算过程。（8）



图 1.

答：

**拓扑序列：v1，v3，v2，v6，v5，v4**

**第一趟：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **V1** | **V2** | **V3** | **V4** | **V5** | **V6** |
| **0** | **∞** | **∞** | **∞** | **∞** | **∞** |

**第二趟**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **V1** | **V2** | **V3** | **V4** | **V5** | **V6** |
| **0** | **20** | **15** | **∞** | **∞** | **∞** |

**第三趟**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **V1** | **V2** | **V3** | **V4** | **V5** | **V6** |
| **0** | **19** | **15** | **∞** | **∞** | **25** |

**第四趟**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **V1** | **V2** | **V3** | **V4** | **V5** | **V6** |
| **0** | **19** | **15** | **∞** | **29** | **25** |

**第五趟**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **V1** | **V2** | **V3** | **V4** | **V5** | **V6** |
| **0** | **19** | **15** | **29** | **29** | **25** |

**第六趟**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **V1** | **V2** | **V3** | **V4** | **V5** | **V6** |
| **0** | **19** | **15** | **29** | **29** | **25** |

1. 设用于通信的电文由字符集{a, b,c,d,e,f,g}中的字母构成，它们在电文中出现的频度分别为{0.34,0.12,0.10,0.08,0.13,0.20,0.03},如何为这7个字母设计二进制前缀编码使得电文总长最短，写出编码过程。（7分）

**可以通过构造哈夫曼树对字母进行编码**

**a:10**

**b:110**

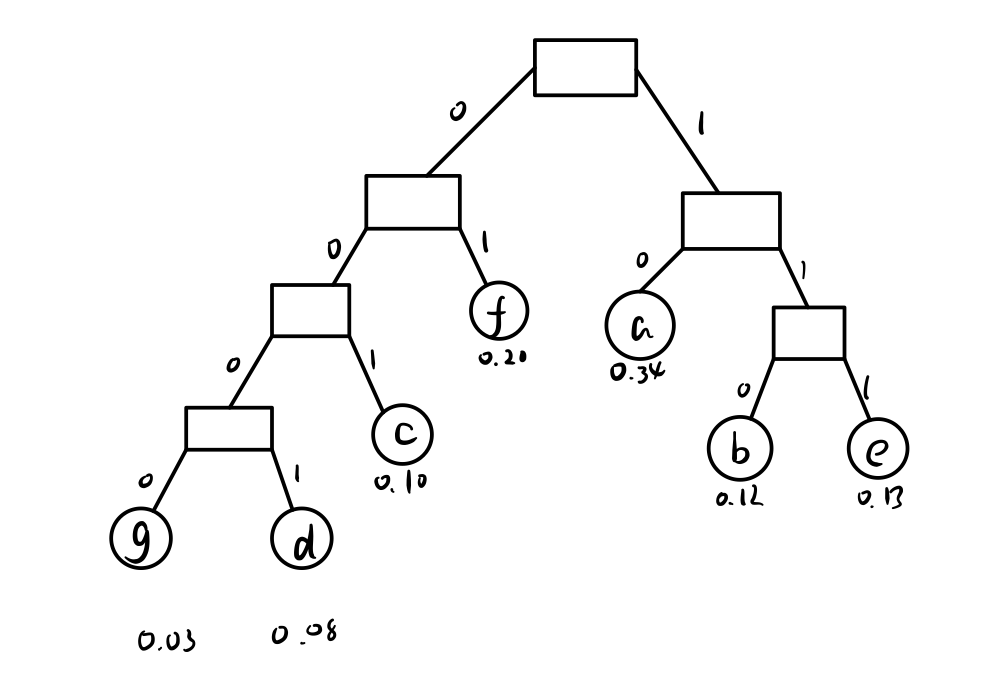
**c:001**

**d:0001**

**e:111**

**f:01**

**g:0000**

****

**五．算法填空，（每空2分,共20分）**

1. 以下算法功能是:插入元素e到队列Q中, 完成算法的空格部分。

typedef struct Qnode

{ QElemType data;

struct Qnode \*next;

}Qnode, \*QueuePtr;

typedef struct

{ QueuePtr front; //队头

QueuePtr rear; //队尾

} LinkQueue;

status EnQueue(LinkQueue &Q, QElemType e)

{ P= (QueuePtr)malloc(sizeof(Qnode));

if ( ① ? ) exit(OVERFLOW);

P->data= ② **e**

P->next= ③**NULL**

④ **Q.rear->next** =P;

Q.rear= ⑤ **P** ;

Return OK;

}

2. 以下程序为图的深度优先遍历算法，完成算法的空格部分。

Boolean visited[Max]; //访问标志数组

Status (\*VisitFunc)(int v); //访问函数变量

void DFStraverse( Graph G , Status(\*visit)(int v)) {

vistFunc=visit;

for (v=0;v<G.vexnum;++v) visited[v]=False;

for (v=0; ⑥ **v<G.vexnum** ； ++v)

if ( ⑦ **visted[v]==False** ) DFS(G, v); }

void DFS(Graph G, int v) {

visited[v]= ⑧**TRUE** ; VisitFunc(V);

for (w=FirstAdjvex(G,v); ⑨**w==-1** ; w= NextAdjVex(G,v,w) )

if (!visited[w]) ⑩ **DFS(G,w)** ;

}

**六．编写算法（25分）**

1．已知线性表中的元素按值递增有序排列，并以带头结点的单链表作存储结构。试编写算法 ，删除表中所有值大于x且小于y的元素（若表中存在这样的元素）, 同时释放被删除结点空间。（10分）

**typdef struct listNode{**

**element data;**

**listNode \*next;**

**};**

**void Process(listNode \*L){**

**listNode \*p = L, \*s;**

**while(p->next!=NULL){**

**if(p->next->data > x || p->next->data < y){**

**s = p->next;**

**p->next = p->next->next;**

**free(s);**

**}**

**p = p->next;**

**}**

**}**

2．设计一个算法，求不带权无向连通图G中距离顶点v的最远顶点。（15分）

**int getDistance(int G[][n], int n, int v){**

**//G是无向连通图，n是顶点数**

**bool flag[n];**

**//flag表示当前的顶点是否已经被访问过**

**int D[n], queue[n], first = 0, last = 0;**

**//D表示各顶点与顶点v的距离，queue是模拟队列，first和last分别是队列的首位指针**

**int result = -1, distance = -1;**

**//restult和distance分别是距离顶点v最远的顶点和其距离**

**for(int i = 0; i < n; i++) flag[n] = false;**

**D[v] = 0;**

**queue[last++] = v;**

**flag[v] = true;**

**while(first != last){**

**int u = queue[first++];**

**if(D[u] > distance){**

**D[u] = distance;**

**result = u;**

**//当发现有比上一轮得到的距离顶点v最远的顶点更远的顶点时更新最远顶点**

**}**

**for(int i = 0; i < n; i++){**

**if(!flag[i] && G[u][i]){**

**D[i] = D[u]+1;**

**queue[last++] = i;**

**flag[i] = true;**

**}**

**}**

**}**

**return result;**

**}**