**2016年全国硕士研究生统一入学考试自命题试题（A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. 在线索化二叉树中，T所指结点没有左子树的充要条件是（ **B** ）。

A. T-> lchild=NULL B. T->ltag=1

C. t->ltag=1且t-> lchild =Null D. 以上都不对

2. 一个带有头结点的单链表为空的判定条件是 ( **B** )。

A. head == NULL B. head->next == NULL

C. head->next == head D. head != NULL

3. 线性链表不具有的特点是（ **A** ）。

A. 随机访问 B. 不必预估所需存储空间大小

C. 插入与删除时不必移动元素 D. 所需空间与线性表长度成正比

4. 在下面的排序方法中，稳定的是（ **C**  ）。

A. 希尔排序 B. 堆排序 C. 插入排序 D. 快速排序

5.设有n个待排序的记录关键字，则在堆排序中需要（ **A** ）辅助记录空间。

A．O(1) B. O(n) C. O(nlog2n) D. O(n2)

6. 数组A［５］［６］的每个元素占5个字节，将其按行优先次序存储。假设A[1][1]元素的存储地址为1000，则元素A［５，５］的存储地址为( **A** )。

A. 1140 B. 1145 C. 1120 D. 1125

7. 高度为n的完全二叉树的结点数至少为（　**A** 　）。

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

8. 设有一个无向图G=（V，E）和G’=（V’，E’），如果G’为G的生成树，则下面不正确的说法是（ **B** ）。

A．G’为G 的子图 B．G’为G 的连通分量

C．G’为G的极小连通子图且V’=V D．G’为G的一个无环子图

9. 在有向图的邻接表存储结构中，顶点V在表结点中出现的次数是（ **C**  ）。

A. 顶点V的度 B. 顶点V的出度

C. 顶点V的入度 D. 依附于顶点V的边数

10. 关键路径是事件结点网络中( **D** )。

A．最短的回路 B．从源点到汇点的最短路径

C．最长的回路 D．从源点到汇点的最长路径

11. 一个有n个结点的无向图最多有（  **D**  ）条边。

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

12. 对某个无向图的邻接矩阵来说，(  **A**  )。

A．第i行上的非零元素个数和第i列的非零元素个数一定相等

B．矩阵中的非零元素个数等于图中的边数

C．第i行上，第i列上非零元素总数等于顶点vi的度数

D．矩阵中非全零行的行数等于图中的顶点数

13. 平衡二叉树的平均查找长度是 ( **D** )。

A. O(n2） B. O(nlog2n） C. O(n） D. O(log2n）

14. 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

15. 设一数列的顺序为1,2,3,4,5,6, 通过栈操作可以得到（ **A** ）的输出序列。

A. 3,2,5,6,4,1 B. 1,5,4,6,2,3

C. 6,4,3,2,5,1 D. 3,5,6,2,4,1

**二．填空题(每空2分，共20分)**

1. 在一个长度为n的顺序表中删除第i个元素时，需向前移动  **n-i**  个元素。

2. 设数组Data[0..m]作为循环队列SQ的存储空间，front为队头指针，rear为队尾指针

则执行出队操作时front指针的值应更新为 front= **(SQ.front+1)%(m+1)**  。

3. 在单链表中，若要删除指针p所指结点的后一结点，则需要执行下列语句：（设q为指针

变量）q=p->next;  **p->next=p->next->next**  ； **free(q)**  。

4. 在有n个结点的二叉链表中，值为NULL的链域的个数为 **n+1** 。

5. 二叉树中度为0的结点数为30，度为1的结点数为30，总结点数为  **89** 。

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

7. 对于n个记录（假设每个记录含d个关键字）进行链式基数排序，总共需要进行 **d** 趟分配和收集。

8. 设有向图G中有向边的集合E={<1，2>，<2，3>，<1，4>，<4，2>，<4，3>}，则该图的一种拓扑序列为  **1,4,2,3**  。

**三．判断题（每题1分，共10分，正确的选t，错误的选f）**

1. 在n个顶点的无向图中，若边数>n-1，则该图必是连通图。 （ **×** ）

1. 具有n个结点的二叉排序树有多种，其中树高最小的二叉排序树是最佳的（ **√** ）

3. 使用散列法存储时，哈希表的大小可随意选取，通常取10的倍数。（ **×** ）

4. 向一个二叉排序树插入新的结点时，新插入的结点总是叶子结点（ **√** ）

5. 数据元素是数据的最小单位。（ **√** ）

6. 普里姆(Prim)算法相对于克鲁斯卡尔(Kruskal)算法更适合求一个稀疏图G的最小生成树。（ **×** ）

7. 向二叉排序树中插入一个新结点，需要比较的次数可能大于此二叉树的高度h。（ **×** ）

8. 向一棵B\_树插入元素的过程中，若最终引起树根结点的分裂，则新树高度为原树的高度加1。（ **√** ）

9. 无向图的邻接矩阵一定是对称阵。 ( **√** )

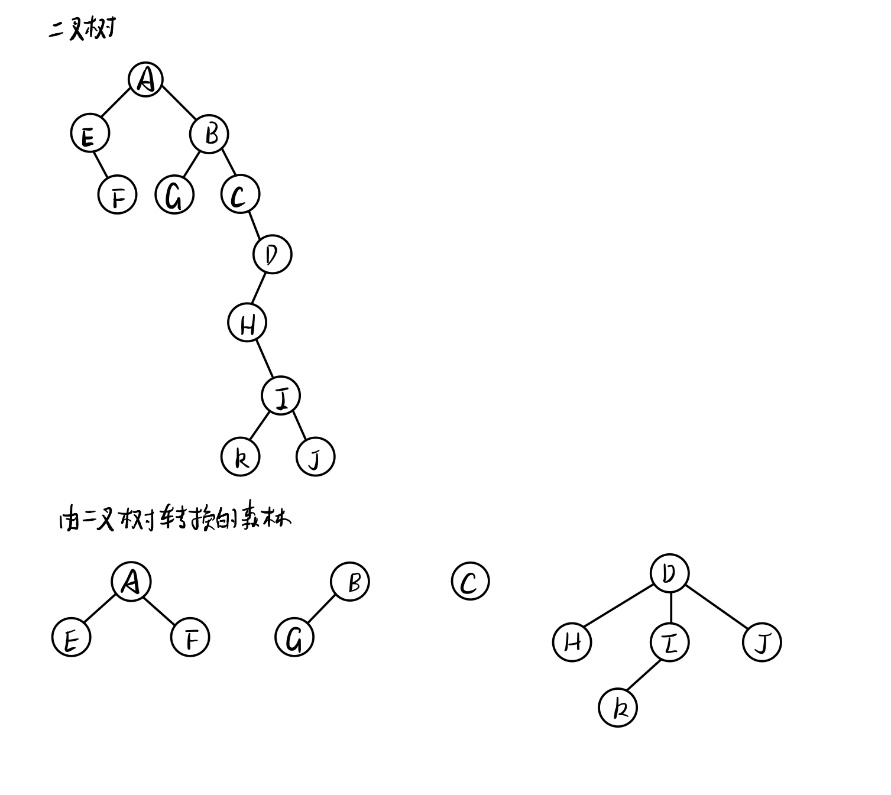
10. 对小根堆进行层次遍历可以得到一个有序序列。（ **×** ）

**四. 简答题（45分）**

1. 已知二叉树的前序遍历序列是AEFBGCDHIKJ，中序遍历序列是EFAGBCHKIJD，求解下列问题：

(1) 画出此二叉树。（4分）

(2) 将该二叉树转换成森林。（4分）

2. 设有一组关键字（71,23,73,14,55,89,33,43,48）,采用哈希函数：H(key)=key %10，采

用开放地址的二次探测再散列方法解决冲突，试在散列地址空间中对该关键字序列(按从左

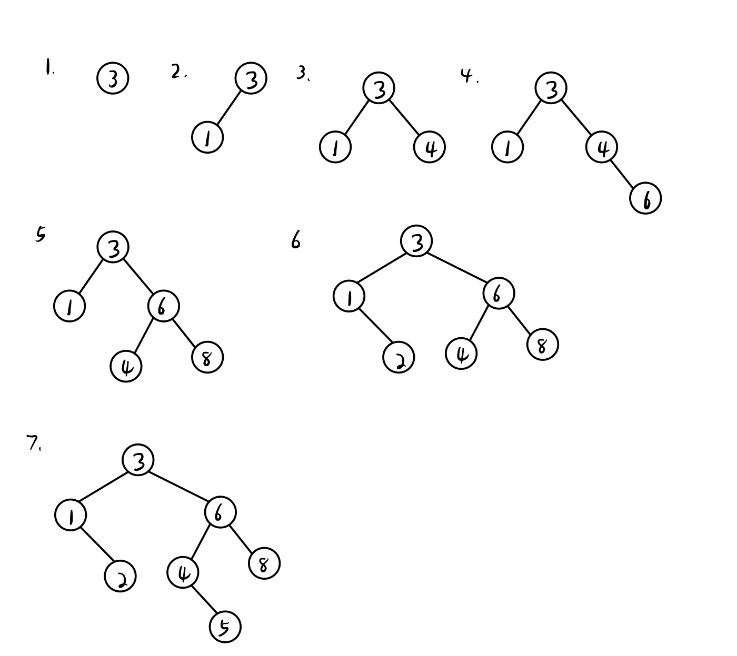
到右的次序)构造哈希表，并计算在查找概率相等的前提下，成功查找的平均查找长度。

（7分）

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  | 71 | 33 | 23 | 73 | 14 | 55 | 43 | 48 | 89 |

ASL（成功）= (1+1+2+2+2+1+2+4+1)/9=16/9

3. 设有一组初始记录关键字为(3,1,4,6,8,2,5)，要求构造一棵平衡二叉树，并给出构造过程。

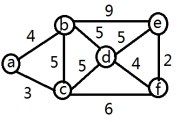
（5分）

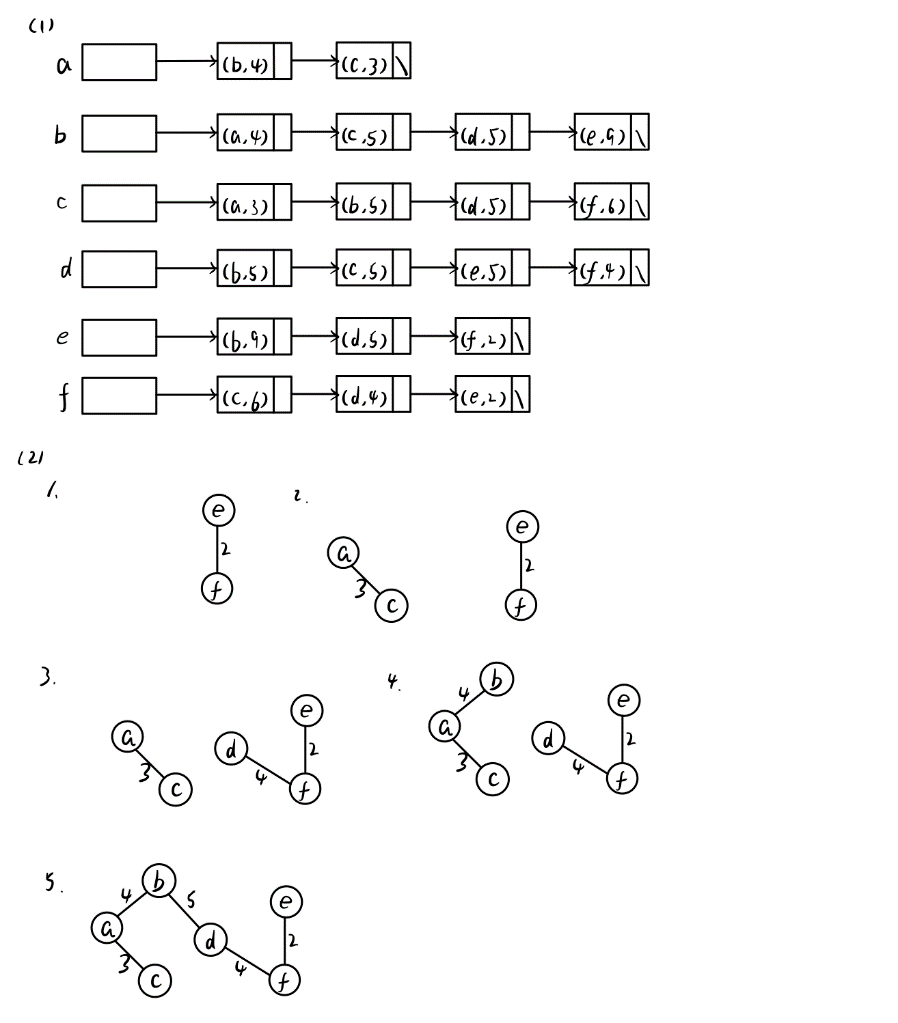
4. 对图1所示的无向加权图完成下列要求：

（1）写出它的邻接表；（5分）

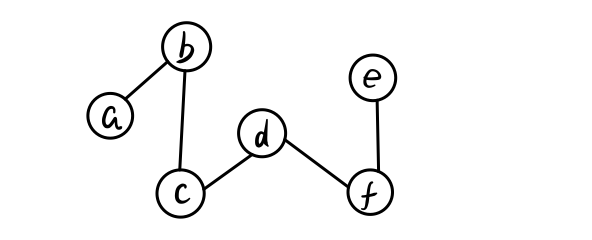
（2）按克鲁斯卡尔（Kruskal）算法求其最小生成树，并给出其过程。（6分）

（3）给出从顶点a开始的深度优先搜索序列和深度优先生成树。（4分）



 图 1

(3)深度优先搜索序列：a,b,c,d,e,f

深度优先生成树

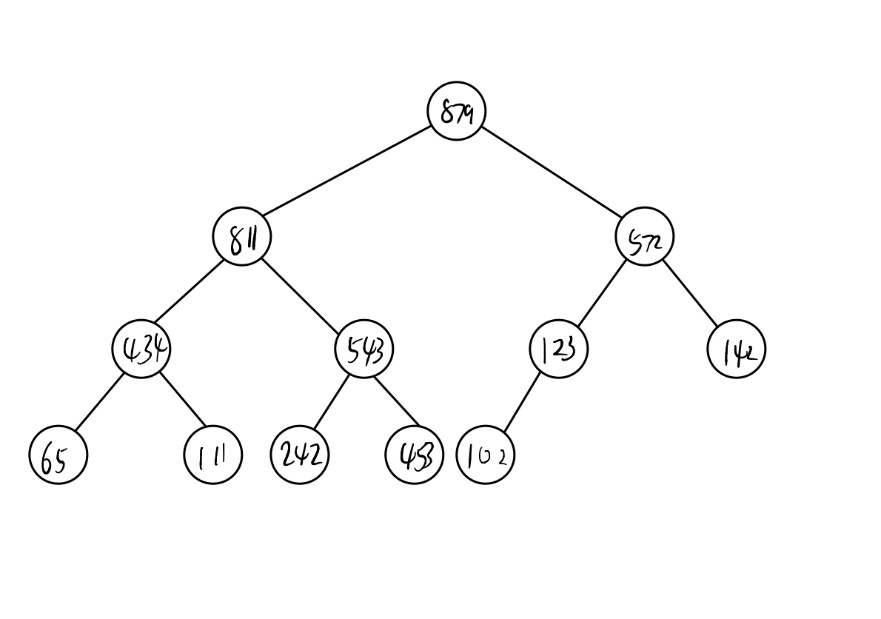
5. 已知序列(142,543,123,65,453,879,572,434,111,242,811,102)。

(1) 采用希尔排序对该序列作升序排序，请给出第一趟排序的结果（初始步长为7）。（5分）

1. 采用堆排序对该序列作升序排序，请给出初始堆以及第一趟排序的结果。（5分）

答：（1）142，111，123，65，102，879，572，434，543，242，811，453

（2）初始堆：



第一趟：811，543，572，434，453，123，142，65，111，242，102，879

**五．算法填空，（每空2分,共20分）**

1. 下面算法实现对一个不带头结点的单链表L进行就地（不增加额外存储空间）逆置。请在\_\_\_\_\_\_\_\_\_\_处填上适当内容，使其成为一个完整算法。

typedef int DataType;

typedef struct {

DataType data;

struct Node \*next;

}Node;

typedef Node \* LinkList;

LinkList Reverse(LinkList L)

{

LinkList p, q;

if (!L) return; //链表为空返回

p=L->next; q=L->next; L->next=NULL;

while(q)

{

q=q->next;

**(1)p->next=L**

**(2)L=p**

p=q;

}

return L;

}

2. 下面是一个采用二叉链表存储结构, 中序遍历线索二叉树T的算法。 Visit是对结点操作的应用函数。请在\_\_\_\_\_\_\_\_\_\_处填上适当内容，使其成为一个完整算法。???

/\*二叉树的二叉线索存储表示\*/

Typedef enum PointerTag{Link, Thread};

typedef struct BiThrNode {

TelemType data;

struct BiThrNode \*lchild, \*rchild;

PointerTag LTag, RTag;

} BiThrNode, \*BiThrTree;

Status InOrderTraverse\_Thr(BiThrTree T, Status(\* Visit)(TelemType e))

{

BiThrNode \*p;

p= (3)T->lchild;

while(p!=T){ //空树或遍历结束时p==T

while(p->LTag==Link) (4)p=p->lchild;

if(!Visit(p->data)) return ERROR;

while (p->RTag==Thread && (5)p->rchild!=NULL

{

(6)p = p->rchild;

Visit（p->data）；

}

(7) p=p->rchild;

}

return OK;

}

}

1. 下面是一个利用递归对二叉排序树进行查找的算法。请在\_\_\_\_\_\_\_\_\_\_处填上适当内容，使其成为一个完整算法。

typedef struct BTreeNode {

TelemType data;

struct BTreeNode \*lchild, \*rchild;

} BTreeNode;

bool Find(BTreeNode\* T, TelemType& item)

{

if( (8)T==NULL )

return FALSE; //查找失败

else {

if (item==T->data) //查找成功

return TRUE;

else if(item<T->data)

return Find( (9)T->lchild , item );

else

return Find( (10)T->rchild , item );

}

}

**六．编写算法（25分）**

1. 设有一组初始记录关键字序列（K1，K2，…，Kn），要求设计一个算法能够在O(n)的时间复杂度内将线性表划分成两部分，其中左半部分的每个关键字均小于Ki，右半部分的每个关键字均大于等于Ki。（10分）

**void solve(elem \*w, int n, int i){**

**int l = 0, r = n-1;**

**int point = w[i];**

**while(l < r){**

**while(w[l] < point) l++;**

**while(w[r] > point) r--;**

**if(l < r) swap(w[l], w[r]);**

**}**

**}**

1. 设有一整型数组w保存n个字符的权值（均大于0），请写出
2. 构造赫夫曼树(Huffman)的算法。（8分）
3. 求各字符赫夫曼编码的算法。（7分）

**#define maxn 500**

**#define INF 0x3f3f3f3f;**

**typedef struct BitTree{**

**int data;**

**int parent;**

**int lchild, rchild;**

**};**

**BitTree\* getHuffman(int w[maxn], int n){**

**bool \*flag = (bool \*)malloc((2\*n+1)\*sizeof(bool));**

**BitTree \*tree = (BitTree \*)malloc((2\*n+1)\*sizeof(BitTree));**

**for(int i = 0; i < 2\*n+1; i++){**

**flag[i] = false;**

**}**

**int cnt = 0;**

**while(n-cnt>=2){**

**int lchildMin = INF, lchild = -1;**

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

**if(!flag[i] && min > w[i]){**

**lchild = i;**

**lchildMin = w[i];**

**}**

**}**

**falg[lchild] = true;**

**int rchildMin = INF, rchild = -1;**

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

**if(!flag[i] && min > w[i]){**

**rchild = i;**

**rchildMin = w[i];**

**}**

**}**

**falg[rchild] = true;**

**n+=1;**

**tree[lchild]->data = w[lchild];**

**tree[lchild]->parent = n;**

**tree[rchild]->data = w[rchild];**

**tree[rchild]->parent = n;**

**tree[n]->data = tree[lchild]->data+tree[rchild]->data;**

**tree[n]->lchild = lchild;**

**tree[n]->rchild = rchild;**

**}**

**tree[n]->parent = -1;**

**return tree;**

**}**

**void print(int w[maxn], int n){**

**BitTree \*tree = getHuffman(w, n);**

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

**printf("第%d个字符的huffman编码：");**

**char \*s = (char \*)malloc(n\*sizeof(char));**

**int length = 0;**

**int v = i;**

**while(tree[v].data!=-1){**

**int u = tree[v].parent;**

**if(tree[u].lchild == v) s[length++] = '0';**

**else s[length++] = '1';**

**v = u;**

**}**

**for(int j = length-1; j >= 0; j--) printf("%c",s[j]);**

**printf("\n");**

**}**

**}**