SDU 2002 数据结构试题

说明：

1、仅供参考，错误之处在所难免，敬请指出

2、函数 ceil(x)取小于等于 x 的最大整数，floor(x)取大于等于 x 的最小整数

一、填空(10 分)

1、一个 m 阶 B-树中，每个结点最少有(*ceil(m/2)*)个儿子结点,m 阶 B+树中每个结点(除根外)最多有(*m*)个儿子结点.

2、n(n>0)个结点构成的二叉树，叶结点最多有(*floor((n+1)/2)*)个，最少有(*1*)个。若二叉树有 m 个叶结点，则度为 2 的结点有(*m-1*)个。

3、顺序查找方法适用于存储结构为(*顺序表和线性链表*)的线性表，使用折半查找方法的条件是(*查找表为顺序存贮的有序表*)

4、广义表 A=(( )，(a，(b，c))，d)的表尾 Gettail(A)为(*((a,(b,c)),d)*)

5、直接插入排序，起泡排序和快速排序三种方法中，(*快速排序*)所需的平均执行时间最小；(*快速排序*)所需附加空间最大。

二、选择(10 分)

1、倒排文件的主要优点是：(C)

A、便于进行插入和删除　　B、便于进行文件的合并

C、能大大提高基于非主关键字数据项的查找速度　　D、易于针对主关键字的逆向检索

2 下面程序段的时间复杂性为(C)

```
y=0;
while(n>=(y+1)*(y+1)) {
      y++;
}
```

A、O(n)　　B、$O(n^2)$　　C、 O(sqrt(n))　　D、 O(1)

3、若从二叉树的任一结点出发到根的路径上所经过的结点序列按其关键字有序，则该二叉树是(　C　)

A、二叉排序树　　B、哈夫曼树　　C、堆　　D、AVL 树

4、栈和队列都是（　B　）

A、顺序存储的线性结构　　B、限制存取点的线性结构

C、链式存储的线性结构　　D、限制存取点的非线性结构

5、用顺序查找方法查找长度为 n 的线性表时，在等概率情况下的平均查找长度为（　D　）

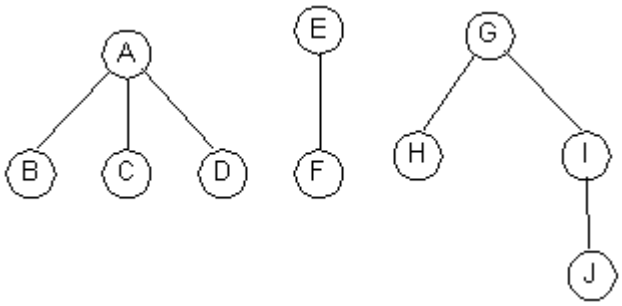
A、n　　B、n/2　　C、(n-1)/2　　D、(n+1)/2

三、简答（30 分）

1、已知一棵二叉树的前序扫描序列和中序扫描序列分别为 ABCDEFGHIJ 和 BCDAFEHJIG，试给出该二叉树的后序序列并绘出该二叉树对应的森林。

解：

后序序列为：DCBFJIHGEA



2、若对序列（7，3，1，8，6，2，4，5）按从小到大排序，请写出起泡排序的第一趟结果和堆排序的初始堆。

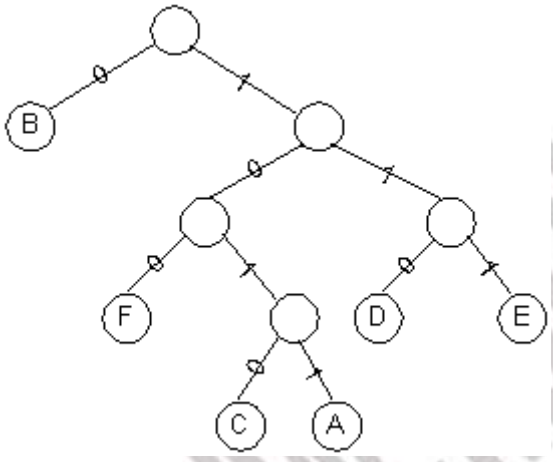
解：

冒泡：3 1 7 6 2 4 5 8

堆：8 7 4 5 6 2 1 3

3、某通讯系统只可能有 A、B、C、D、E、F 6 种字符，其出现的概率分别是 0.1、0.4、0.04、0.16、0.19、 0.11，试画出相应的哈夫曼树，并设计哈夫曼编码。

解：



编码：A:1011　B:0　C:1010　D:110　E:111　F:100

4、在二叉平衡检索树（AVL 树）的调整中，将最靠近新插入点的不平衡结点调整平衡后，树中是否还会有不平衡结点？为什么？

解：

不会再有不平衡点。因为插入结点发生不平衡现象后，会改变
以“靠近新插入点的不平衡结点”为根的子树（即最小不平横树）的高度加 1，
经过调整后使最小不平衡树的整体高度又恢复到原来的值，所以不会对原平衡树
的其他部分造成危害，因此不会再有不平衡点。

5、指定 Hash 函数 H(k)=3*k mod 11 及线性探测开地址法处理冲突，试在 0~10 的散列空间中对关键字序列（22，41，53，46，30，13，01，67）构造 Hash 表，并求在等查找概率下查找成功的平均查找长度。

解：

插入元素后的分布情况：

0	1	2	3	4	5	6	7	8	9	10
22		41	30	01	53	46	13	67		

ASL = (1+1+1+1+2+2+2+6)/8=2.0

四、（10 分）下图是带权的有向图 G 的邻接矩阵表示，请给出：

1、其邻接表存储结构

2、按 Floyd 算法求所有顶点对之间最短距离的矩阵变化过程。

	V1	V2	V3	V4
V1|	0	1	∞	4
V2|	∞	0	9	2
V3|	3	5	0	8
V4|	∞	∞	6	0

解：Floyd 算法执行过程中矩阵的变化情况为（从左到右）：

0 1 ∞ 4	0 1 10 3	0 1 10 3	0 1 9 3
∞ 0 9 2	∞ 0 9 2	12 0 9 2	11 0 8 2
3 4 0 7	3 4 0 6	3 4 0 6	3 4 0 6
∞ ∞ 6 0	∞ ∞ 6 0	9 10 6 0	9 10 6 0

五、（12 分） 设双链表结点结构为 llink data rlink，请设计算法将其中 P 所指结点与其 rlink 所指结点位置互换的算法。

解：

```
typedef struct DLNode{
    ElemType data;
    struct DLNode *llink,*rlink;
}DLNode,*DLinkList;
//思想：将 P->rlink 先从链表中删除掉，然后再插入到 P 前
Status SwapANode(DLNode *&P)
{
    //结点存在吗?
    if(!P || !(P->rlink))return ERROR;
    q = P->rlink;
    //删除 q 结点
    if(!q->rlink)
        P->rlink = NULL;
    else {
        P->rlink = q->rlink;
        q->rlink->llink = P;
    }
    //将 q 结点插入到 P 结点前面
    if(!P->llink)
    {
        q->llink = NULL;
```

```
        q->rlink = P;
        P->llink = q;
    } else {
        q->llink = P->llink;
        q->rlink = P;
        P->llink->rlink = q;
        P->llink = q;
    }
    return OK;
}
```

六、（13 分） 若有一棵二叉树的存储结构为二叉链表，T 指向根结点，请写出一个非递归算法判定其是否为二叉排序数。

解：

解法一：

```
#define TRUE  1
#define FALSE 0
typedef int BOOL;
typedef struct BTreeNode {
    ElemType data;
    struct BTreeNode *lchild,*rchild;
}BTreeNode,*BTree;

//我是将教材 P130 的中序非递归遍历方法改的
//注释没写，大家看书上的吧
BOOL IsBST(BTree T)
{
    if(!T) return TRUE;
    InitStack(S);
    x = T->data;
    Push(S,T);

    while(!StackEmpty(S))
    {
        while(GetTop(S,p) && p) Push(S,p->lchild);
        Pop(S,p);
        if(!StackEmpty(S))
        {
            Pop(S,p);
            if(x > p->data) return FALSE;
            x = p->data;
            Push(S,p->rchild);
        } // if
    // while
    return TRUE;
```

```
}
```

另解：

```
//我的另外一个解法,根据 longeli 的思想
//Author: Ritchie
//  Date: 2002-10-12
typedef int ElemType;
typedef struct BiTreeNode {
    ElemType data;
    struct BTreeNode *lchild,*rchild;
}BiTreeNode,*BiTree;
//功能：判断二叉树 T 是否为二叉排序树,如果是返回 TRUE,否则返回 FALSE
//思想：判断 T 中的结点是否符合要求,按层序进行判断,一旦不符合就返回
Status IsBST(BiTree T)
{
    if (!T) return TRUE; //空树是 BST
    InitQueue(Q);
    EnQueue(Q,T);
    while(!QueueEmpty(Q))
    {
        DeQueue(Q,t);
        //左右孩子先入队
        if(t->lchild) EnQueue(Q,t->lchild);
        if(t->rchild) EnQueue(Q,t->rchild);
        if(t->lchild && t->lchild){
            // 左右子树不为空且不满足 BST 的条件，返回 FALSE
            if((t->lchild->data>=t->data)||(t->rchild->data< t->data))
                return FALSE;
        } else if (t->lchild && (t->lchild->data >= t->data)) {
            //右子树为空的情况
            return FALSE;
        } else if(t->rchild && (t->rchild->data < t->data)) {
            //左子树为空的情况
            return FALSE;
        }
    }
    DestroyQueue(Q);
    return TRUE;
}
```

用递归的方式做也有两三种做法，这儿就不列举了。

七、（15 分） 下表列出了某工序之间的优先关系和各工序所需时间，要求：

（1）画出 AOE 网

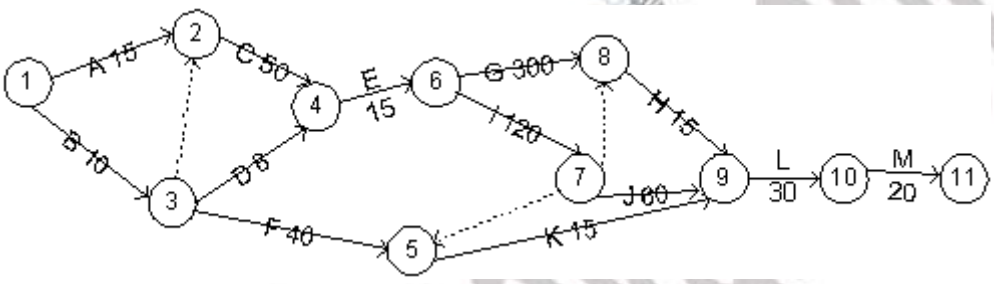
（2）列出各事件的最早、最晚发生时间

（3）找出该 AOE 网中的关键路径，并回答完成该工程所需要的最短时间。

工序代号	所需时间	先驱工序		工序代号	所需时间	先驱工序
A	15	无		H	15	G、I
B	10	无		I	120	E
C	50	A、B		J	60	I
D	8	B		K	15	F、I
E	15	C、D		L	30	H、J、K
F	40	B		M	20	L
G	300	E				

解：

(1) AOE 图如下(需要添加虚线才可以画出图，我就是在这儿被迷惑住的，第一次看到这样的题目)



（2）各事件的最早最迟发生时间:

事件编号	1	2	3	4	5	6	7	8	9	10	11
ve(i)	0	15	10	65	50	80	200	380	395	425	445
vl(i)	0	15	57	65	380	80	335	380	395	425	445

(3)通过上表不用求出活动的最早最迟开始时间就可以看出关键路径为：

1，2，4，6，8，9，10，11

完成工程所需的最短时间为：445