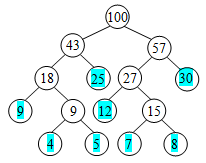
**说明：题目只是复习参考，一定要把题目的相关知识和方法弄懂。**

图的遍历：（教材例题，图7.15、图7.16）

假定用于通信的电文由8个字符A、B、C、D、E、F、G、H组成，各字母在电文中出现的概率为5％、25％、4％、7％、9％、12％、30％、8％，试为这8个字母设计哈夫曼编码。

解： 根据题意，设这8个字母对应的权值分别为（5，25，4，7，9，12，30，8）



规定左分支用0表示，右分支用1表示，字母A、B、C、D、E、F、G、H的哈夫曼编码如下表示：

A:0011 B:01 C:0010 D:1010

E:000 F:100 G:11 H:1011

假设一棵二叉树的先序序列为 ABCDEFGHIJK，中序序列为 CBDEAGFIKJH。

（1）画出该树；

（2）写出后序序列；

（3）写出对应该二叉树的森林；

设关键字输入序列为{30,25,46,22,27,32,40,28,29,70}，依据关键字序列构造平衡二叉排序树。

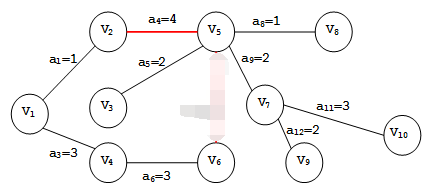
在下图所示的AOE网中，顶点V1,V2,V3,V4,V5,V6,V7,V8,V9,V10表示事件，弧a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14表示活动，请回答以下问题：

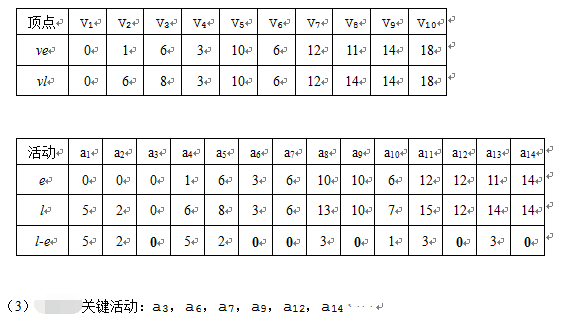


（1）去掉边的方向后，画出最小生成树。

（2）求出所有事件的最早发生和最迟发生时间、所有活动的最早开始和最晚开始时间；

（3）列出所有关键活动。





**二叉树：**

已知二叉树结点定义如下：

typedef struct bitnode{

int data;

bitnode \*lchild,\*rchild;

}bitnode;

写出二叉树叶子计数的递归算法。函数原型为void countleaf(bitnode \*t)。

void countleaf(bitnode \*t){

if(t != null){

int m = countleaf(t->lchild);

int n = countleaf(t->rchild);

if(m+n == 0)

return 1;

else

return m+n;

}

return 0;

}

已知二叉树结点定义如下：

typedef struct bitnode{

int data;

bitnode \*lchild,\*rchild;

}bitnode;

构造一个给每个结点的值加上的递归算法，函数原型为void add(bitnode \*t, int n)。

void add(bitnode \*t, int n){

if(t != null){

add(t->lchild);

t->data += n;

add(t->rchild);

}

}

已知二叉树结点定义如下：

typedef struct Bitnode{

int data;

Bitnode \*lchild,\*rchild;

}Bitnode;

写出计算二叉树高度的递归算法。函数原型为 int height(Bitnode \*p) 。

int height(Bitnode \*p){

int m, n;

if( p != null){

m = 1 + height(p->lchild);

n = 1 + height(p->rchild);

if( m >= n )

return m;

else

return n;

}

return 0;

}

**单链表：**

已知一带表头结点的单链表，结点类型为（data,next）。以 head 为头指针，每个结点的 data 域存放的是一个整数，试构造一个删除链表中值最大的结点的算法。函数原型为void deletemax(LNode \*head)。

void DeleteMax(lnode \*&head){

LNode \*pre = head, \*p = pre->next;

LNode \*maxp = p, \*maxpre = pre;

while( p != null ){ /\*遍历单链表，maxp指向值最大的结点，maxpre指向其前驱\*/

if (p->data > maxp->data)

{

maxp = p;

maxpre = pre;

}

pre = p;

p = p->next;

}

maxpre->next = maxp->next; /\* 将maxp所指结点从单链表中摘除 \*/

free maxp;

}

（与求链表长度类似）

已知带头结点单链表中结点的定义为（data,next），试写出返回链表中结点的值为奇数的元素的个数。函数原型为Count(Lnode \*head)。

int Count(Lnode \*head){

int sum = 0;

Lnode \*p = head->next;

while( p != null){

if( p->data%2 == 1 )

sum++;

p = p->next;

}

return sum;

}

**栈和队列：**

栈和队列参考教材示例代码

**查找、排序：**

假定一组记录为(46,79,56,25,76,38,40,80)，对其进行快速排序的第一次划分后，右区间内元素的个数为\_\_\_\_\_\_\_\_\_\_。

从具有n个结点的二叉排序树中查找一个元素时，在最坏情况下的时间复杂度为(　　)。

A. O(n)

B. O(1)

C. O(log2n)

D. O(n2)

解析：最坏的情况是O(n)，只有二叉排序树同时也具有平衡树特点是才达到O(log2n)

对于顺序存储的有序表(5,12,20,26,37,42,46,50,64)，若采用折半查找，则查找元素26的比较次数为(　　)。

解析：区间[0, 8]

low=0 high=8

计算 mid=4，比较 26 < a[mid]，更新区间为low=0 high=mid-1=3，即[0, 3]

计算 mid = 1 ... 比较并更新区间为[2, 3]

计算 mid = 2 ... 比较并更新区间为[3,3]

计算 mid =3 比较，查找成功。