2005 年北京交通大学计算机专业考研辅导班笔记

（05 年有好多内容和 04 年一样，04 年有不同我会特别用蓝色注明）

第一章：概论（05 年）

1. 设有两个算法在同一机器上运行，其执行时间分别为 100*n**2 和 2**n，要是前者快于后者，n 至少要多大？

   求不等式 100n**2<2**n, → n>=15

2. 算法的时间复杂度仅与问题的规模相关吗？

   事实上，时间复杂度不仅与问题的规模有关，还与问题的初始状态相关，如起泡排序里时间复杂度就与排序的初始状态有关。

3. 若所需额外空间相对于输入数据量是常数，则称算法为原地工作！（掌握概念）

   有可能出这样的题：给你个算法让你判断它是否是原地工作。 如：简单排序，起泡排序等！

总结：第一章考的内容不多，主要是复杂度问题

概论（04 年）

强调的内容和 05 年差不多，但着重讲了算法复杂度的计算。如下：

1. （1）x=0; y=0; 1 次
   (2) for (k=1;k<=n;k++) n+1 次
   (3) x++; n 次
   (4) for(k=1;k<=n;k++) n+1 次
   (5) for(j=1;j<=n; j++) n(n+1)次
   (6) y++ n**2 次
2. x=1 1 次
   for(k=1;k<=n;k++) n+1 次
   for(j=1;j<=i; j++) Σ(i+1) (求和下限 i=1,上限 n+1)
   for(k=1; k<==j;k++)
   x++; ΣΣj(第一个求和下限 i=1, 上限 n；第二个求和下限 j=1, 上限为 i)
   =Σ(i+1)/2 (求和下限 i=1, 上限 n)
   =(n(n+1)(2n+1))/12+(n(n+1))/4

3. 简单选择排序和起泡排序的比较次数

第二章： 线性表（05 年）

1. 熟悉线性表的逻辑结构及其性质（书上有）
2. 理解插入，删除，定位这三个算法及过程（顺序表，各种链表应熟悉）
3. 循环链表的用法（约瑟夫环，猴子选大王（参看 04 年填程序第二题）自己编一下程序）
4. 双向循环链表判空（head->next=head 或 head->pre=head 带头结点），判满的条件以及它的插入和删除结点的操作。
5. 在顺序表中插入或删除一个结点需平均移动多少个结点？具体的移动次数取决于哪两个因素？

   答：参看书 P25

   取决于顺序表的长度 n，和需要插入和删除的位置 i（i 越接近 n 需要移动的结点越少）

5. 为什么在单循环链表中设尾指针比设头指针好？

   答：用尾指针可以使得查找链表的开始结点和终端结点都很方便。设一带头结点的

单循环链表，其尾指针为 rear 则开始结点和终端结点的位置分别 rear->next->next 和 rear.查找时间都是 O(1). 若用头结点表示则查找终端结点的时间是 O(n)；

6. 在单链表，双链表和单循环链表中，若只知道指针 p 指向某结点，不知道头指针，能否将结点*p 从中删除？

   答： 单链表 不行

   双链表 可以 O(1)

   单循环 可以 O(n) 从 p 开始往后，总可以找到 p 前面的一个结点。

7. 下述算法的功能是什么？

```
Linklist Demo(linklist L)
 {  //L 是头结点
   listNode  *q, *p;
   if (L&&L->next)  //保证有两个结点
     { q=L; L=L->next; p=L;
       while (p->next) p=p->next;
       p->next=q; q->next=Null;
     }   return L;
}//该程序是把第一个结点挪到最后，第二个结点变为第一，返回的 L 为新链表的头指针
```

   答：若 L 指向的单链表至少有两个结点，将第一个结点移到终端结点之后成为新的终端结点。而 L 指向原来的第二个结点，使其成为新的开始结点，并返回新链表的头指针；否则直接返回 L 值不作任何变动

   （老师强调了在做阅读程序的题目时，一定要把其描写得具体些，这样才能保证多拿分）

8. 试分别用顺序表和单链表作为存储结构，写程序对其就地逆置，要求辅助空间为 O(1).
9. 顺序表 L 是递增（或递减）有序表，将 x 插入后，使其仍然有序。
10. 已知 L1,L2 分别指向两个单链表的头结点，试写一算法将两个链表连接在一起，并分析算法的时间复杂度（min(m,n)短的放前面， 把第二个链表的头结点去掉。从短的头结点开始一直找到尾部，并让尾结点指向长链表（last->next=L2->next））
11. 设 A,B 两个单链表，其表中元素递增有序。试写一算法将 A,B 归并成一个递减的 C，要求辅助空间为 O(1),并求时间复杂度 （参看 P21）
12. 约瑟夫环应用

以上题目希望大家能自己动手做做

第三章 栈和队列(05 年)

1. 栈和队列：受限的线性表。

   一般的线性表有： 插入点 n+1 个，删除点 n 个

   栈，队列： 插入点 1 个， 删除点 1 个

2. 入栈，出栈，入队，删除队头的操作均应掌握（包括算法）
3. 掌握循环队列
4. 例题 P48 数制转换
5. 括号匹配 知道是怎么回事就行
6. 迷宫求解 录音里有老师详细讲解
7. 回文游戏 顺读与逆读字符串一样（不含空格）

（1） 读入字符串 （2）去空格 （3） 压入栈 （4）依次出栈与原字符串比较若不等则非回文，若直到栈空都相等则为回文。

考虑另一种方法：若字符串的长度为奇数，则不需比较为非回文。否则可先读入一半字符入栈，然后依次出栈和剩下的字符比较！自己可用这种方法编写一下。

8. 地图四染色问题（未考过）

使地图中相邻的不重色，最少用 4 种颜色可以实现。利用栈回溯。

设一个邻接矩阵 R[][]，主对角线上的元素均为零。其于元素如 R1,3, 如果第一个区域和第三个区域相邻的话则 R1,3 为 1，否则为 0。再使用一个工作数组 S[]用来存放已填色区域的号码。

```
Void  mapcolor (int R[][], int n, int S[])  // n 表示地图共有 n 个区
  { S[1]=1;  //1 号区填 1 号色
    a=2;j=1;  //a 为区号，j 为色号
    while (a<=n)  //a>n 表示填色完成
      { while ((j<=4)&&(a<=n))
            { k=1;      //k 表示已填色的区域
              while ((k<a)&&(s[k]*R[a-1][k-1]!=j))  k=k+1;
                //若不邻，或相邻且不重色，对下一个区进行判断
               if (k<a) j=j+1;  //相邻且重色，色号加 1
               else { s[a]=j; a=a+1; j=1;} //相邻不重色，又从 1 号区着色
            }
         if (j>4) { a=a-1; j=s[a]+1;} //对当前需着色区域 a 来说，1-4 种颜
                                       色都不行，则说明上一个错了，对上
                                       一个进行重填
  }
```

9. 四皇后问题

```
#include<stdio.h>
#define   n   4    // n 是皇后的个数
int m=0, a[n]; //a[i]存放第 i 个皇后放置的行号
int ok(int i, int j)    //检查(i,j)能否放棋子
 { int j1, i1,ok1;
   j1=j; i1=i; ok1=1;
   while ((j1>1)&&ok1)   {j1--;   ok1=a[j1]!i;} //查左边那列该行是否有皇后
   j1=j; i1=i;   //检查对角线上能否放
   while((j1>1)&&(i1>1)&&ok1)    {j1--;   i1--; ok1=a[j1]!i1 ;}
   j1=j; i1=i; //检查另一对角线能否放
   while ((j1>1)&&(i1<n)&&ok1)    {j1--;   i1++; ok1=a[j1]!i1 ;}
   return   ok1;
 }
void   queen(int j) //从第 j 列开始试探
  {   int i;
      if (j>n) //放完了，打印摆法计数
        { m++;   printf("m=%d    ", m);
          for (i=1; i<=n ; i++)    printf("        %d",a[i]);
          printf("\n");
```

```
        }
    else    for(i=1; i<=n; i++)
                if (ok(i,j)) //检查(i,j)上能否放
                {   a[j]=i;   queen(j+1);}//在(i,j)上放，第 j 列的皇后在第 i 行
}
main ()
  {   queen(1);   } //  从第一列开始试探
```

10．队列：　注意循环队列判空判满的条件（增加一个元素的空间）

11.在非循环队列中：

当 Q.front=Q.rear<>0 时能否判空？　　能

当 Q.rear=0 时，能否判空？　　能

Q.front=0 时，能否判空　　不能

13. 循环队列判空判满及求长度

空：　Q.front=Q.rear

满：　(Q.rear+1)%MAXSIZE=Q.front

长度：（Q.rear-Qfront+MAXSIZE）%MAXSIZE

14. k 阶斐波那挈序列

试用循环队列编写求 k 阶斐波那挈序列中的前 n+1 项（f0.,f1,f2,….fn）的算法，要求满足 fn<=max 而 fn+1> max， 其中 max 为某个约定的常数。（注意本题所用的循环队列的容量为 k）

方法一：fi=fi-1+…fi-k（未考过）

而 fk=f0+….f(k-1),用 fk 冲掉 f0,用 f(k+1)冲掉 f1,依次循环求出 fk,直到发 fk>max 为止

```
Void fb(int k; int max)
 { for(i=0;i<=k-2;i++) {f[i]=0; cq.elem[i]=0;}
   cq.elem[k-1]=1; cq.rear]k-1;n=k;
   while (cq.elem[cq.rear]<max)
     {f[n]=0;
      for(j=0;j<k;j++) f[n]=f[n]+cq.elem[j];
      cq.rear=(cq.rear+1)%k;
      cq.elem[cq.rear]=f[n]; n++;
     }
   if(cq.elem[cq.rear]>max) n=n-2;
   else n=n-1;
   if (max=1) {n=k; f[k]=1;}
}
```

方法二：fi=f(i-1)+…f(i-k);　f(i+1)=fi+f(i-1)+….f(i-k+1)

两式相减：f(i+1)=2*fi-f(i-k)

f(k)=2*f(k)-f(0)　(时间复杂度要小)

```
Void   fb (int k;   int max;)
 {   for (i=0;i<=k-2; i++) {f[i]=0; cq.elem[i]=0;}
     cq.elem[k-1]=cq.elem[k]=1; cq.rear=k;   n=k+1;   f[k-1]=f[k]=1;
    while (cq.elem[cq.rear]<max)
      { j=(cq.elem+1)%(k+1);
```

```
            f[n]=cq.elem[cq.rear]*2-cq.elem[j];
            cq.elem[j]=f[n];   cq.rear=j; n++;
          }
        if (cq.elem[cq.rear]>max) n=n-2;   else n=n-1;
        if (max= =1) {n=k; f[k]=1;}
        if (max= =0) n=k_2;
    }
```

第四章：串

1. 重点：KMP 算法

   （要求书上的程序要看懂，比如说 04 年就出了模式匹配的阅读程序的题目）

   会求 next 和 nextval 数组的值。

   本章考试内容基本就是几个形式，要么给你个字符串要求它的 next 或 nextval 数组，或者会给出你主串和子串，要求你写出匹配的过程，还有就是 04 年的那种阅读的形式了！

第五章：数组，广义表

1. 两类矩阵

   1）：特殊矩阵：非 0 元的分布有一定规则。如对称矩阵，三角矩阵，对角矩阵

   2）：稀疏矩阵

2. 对称矩阵（出选择和填空的机会较大）

   对称矩阵最少需要多少个存储单元 n(n+1)/2 个

   关系：k=　　(参看 P95)

   注意在确定对称矩阵和对应一维数组的位置的题目时一定要注意是以行序还是列序，下标是从 0 开始还是从 1 开始

   例：若 1<=i,j<=n 已知 i, j 求 k　　公式 1

   k=(i*(i-1))/2+j-1　　(i>=j)

   k=(j*(j-1))/2+i-1　　(i<j)

   若 0<=i,j<=n-1 已知 i, j 求 k　　公式 2

   k=(i*(i+1))/2+j　　(i>=j)

   k=(j*(j+1))/2+i　　(i<j)

   若 1<=i,j<=n,已知 k, 求 i,j　　（用公式 1）

   已知 k=49 求 i,j

   将 j=i 代入（公式 1）得 k+1<=i(i+1)/2 求使之成立的最小 i .然后再将 i 值代入（公式 1）求出 j

   若 0<=i,j<=n-1 已知 k 求 i,j　　(同上用公式 2)

   以上方法希望大家掌握！

3. 下三角矩阵（类似于对称矩阵）

4. 上三角矩阵

   若 0<=i,j<=n-1 已知 i, j 求 k　（要求自己会推导）

   第 p 行上恰好有 n-p 个元素，在 i 行上 ai,j 前有 j- i 个元素，i 前有 i 个完整行

   所以 第 I 行前共有元素个数为：Σ(n-p) （求和下限为 p=0, 上限为 i-1）= (i*(2n-i+1))/2

   所以 k= (i*(2n-i+1))/2+j-i　　i<=j （上三角）

   若 1<=i,j<=n 已知 i, j 求 k 的情况大家可以自己去推导

   k=((i-1)(2n-i+2))/2+(j-i+1)-1

注意：以上的 k 值均从 0 开始。其实我认为做这种题目，首先看两点，一是矩阵的起始下标，二是数组的开始下标！检验你的公式是否正确你可以取一个比较容易计算的元素比如 a0,2 代入到公式检验下，因为 a0,2 是矩阵的第三个元素！

三角矩阵的存储单元要比对称矩阵多一个元素，用于存储相对三角矩阵的常数值 c，存放在第 n(n+1)/2+1 个单元里

例如：已知一个 9 阶上三角矩阵 A 按行存储在一维数组 B 中，B[0]存放矩阵中第一个元素 a11 则 B[31]存放元素（a5,6），按列序存放，存放元素（a4,8 ）

上面题目在录音中均有详细讲解过程。

以上计算每年至少会有一个填空题！

5. 三对角矩阵

按行序 1<=i,j<=n 。Loc(aij)=Loc(a11)+3*(i-1)-1+j-i+2

若一个 s（素数）对角矩阵满足下述条件：

若 0<=i,j<=n-1，已知 i, j 求 k 。 k=(3*i-1)+(j-i+2)-1=2i+j

若 0<=i,j<=n-1 已知 k 求 i, j i=(k+1)/3, j=k-2*i.

若 1<=i,j<=n 已知 i, j 求 k。 k=3*(i-1)-1+(j-i+2)-1=2*i+j-3

若 0<=i,j<=n-1 已知 k 求 i, j. i=(k+1)/3+1 , j=k-2*i+3

以上公式我认为没有必要去记，可以参考一下人邮出版社的〈数据结构课程辅导与习题解析〉这本书，里面有具体的推导过程。象上面的所有的计算都有具体例题

6 .随机稀疏矩阵存储（只掌握三元组，行逻辑，十字链表不要求掌握）

熟悉带行表的稀疏矩阵的存储结构，会根据行表推导原矩阵（参看 04 年的简答题）

6. 广义表：判断广义表的深度，广度，和表长。会画广义表用书上的第一种存储结构。

有很多朋友画广义表的时候经常会出错，我觉得可以用一个方法去检验你画的图是否正确！你可以根据广义表的表达式用 gethead()和 gettail()函数取得某一个元素（这是一系列的操作组合，参看 04 年的填空第六题的形式），然后看在你画的广义表中从表头开始取得该元素是否也经历了这些操作就行了

第六章：树与二叉树

1．掌握二叉树的 5 个性质，并能灵活运用！第三个性质的证明必须掌握（参看 04 年的简答题 1 ）

2．已知一棵度为 m 的树，有 n1 个度为 1 的结点，有 n2 个度为 2 的结点，有 nm 个度为 m 的结点，问有多少叶子结点？

n0+n1+…+Nm=∑i*ni +1 （求和的下限是 i=1,上限是 m）

n0=∑(i-1)*ni +1

3．两类特殊的二叉树： 满二叉树（结点数为 2**k-1 个），完全二叉树（结点号与满二叉树必须对应）

4．性质 4 的证明 （书上有证明）

5．二叉树的存储结构（书上的定义一定要记住）

	顺序存储结构	链式存储
二叉：	空域	n+1
三叉：	空域	n+2

一般有：在一棵有 n 个结点，度为 k 的树中必有 n*(k-1)+1 个空链域

6．各种遍历的递归和非递归算法均应熟练掌握

中序非递归见 P130

先序非递归参看相关的资料

后序非递归：

```
Void    postorder_fei(BinNode     *t)
{   BinNode     *p;
   printf("post_order_fei :   ");
    top=0; stack[top]=t;
    while (top!=-1)
     {     while (stack[top])
           {     top=top+1;
                 stack[top]=stack[top-1]->lch;
                 top=top-1;
                  if    (top!=-1)
                      {   p=stack[top];
                            if( (!p->rch) || (p->rch->visited==1))
                                 {   stack[top]=null;
                                     printf("%c",p->data);     p->visited=1;
                                 }
                            else
                                 {   top=top+1;   stack[top]=p->rch;   }
                      }
           }
     }
}
```

7．遍历算法的应用举例

(1)求二叉树中叶子结点的个数。（先序（中序，后序）遍历二叉树，设一个全局变量作为计数器，左右指针为空的结点为叶结点）

```
int countleaf(BinTree    T)
  { int n1,n2;
    if (!T) return 0;
    else { if ((!T->lchild)&&(!T->rchild))
             return 1;
           else { n1=countleaf(T->lchild); //n1 存放左子树的叶结点数
                  n2=countleaf(T->rchild);// n2 存放右子树的叶结点数
                  return(n1+n2);
                }
         }
  }
```

(2)求二叉树深度（后序遍历 ）

基本思想：二叉树的深度为左右子树的最大深度加 1

```
int    Depth(BinTree T)
 { int    dep,dep1,dep2;
   if (!T) dep=0;
   else { dep1=Depth(T->lch);
          dep2=Depth(T->rch);
```

```
            dep=1+(dep1>dep2 ? dep1: dep2);
          }
        return dep;
      }
```

(3)复制二叉树（先序遍历）

```
Void copytree(BinTree root, BinTree *newroot) //newroot 为指向指针的指针
  {if (!root) *newroot=Null;
   else
       { *newroot=(BinNode*)malloc(sizeof(BinNOde));
         (*newroot)->data=root->data;
         copytreer(root->lchild, &((*newroot)->lchild));//复制到左子树
         copytree(root->rchild, &((*newroot)->rchild));//复制到右子树
       }
   }
```

(4)交换二叉树的左右子树(类似先序遍历，用后序也可以，但中序不行)

```
Void   exchange(BinNode *T)
 { BinNode *q;
  if (T)
     { q=T->lchild;
       T->lchild=T->rchild;
       T->rchild=q;
       exchange(T->lchild);
       exchange(T->rchild);
     }
 }
```

(5)建立二叉树的存储结构（不同的定义方法相应有不同的存储结构，现以字符串的形式 根 左子树 右子树定义一棵二叉树（即二叉树的扩展序列）如：

以字符串 AB*C**D** (*号表示空字符)

```
status   CreatBitree(BiTree &T) //按先序扩展序列建立二叉树的递归算法
    {   scanf(&ch);
        if (ch= =' ') T=Null;
        else{ if (!(T=(BinNode*)malloc(sizeof(BinNode))));
                 exit(overflow);
              T->data=ch;
              CreatBitree(T->lchild);
              CreatBitree(T->rchild);
            }
      return ok;
     }
```

7. 已知按某规律遍历二叉树的非扩展序列（也即由先序，中序，后序遍历而得的序列），是否可以唯一确定该二叉树的结构？

结论：按某规律遍历二叉树的非扩展序列不能唯一确定该二叉树的结构

8．已知按某规律遍历二叉树的扩展序列，能否唯一确定该二叉树的结构？

结论：先序扩展序列能唯一确定　　中序扩展序列不能唯一确定

后序扩展序列能唯一确定（从后往前推导）

```
//按后序遍历扩展序列建立二叉树结构的递归算法
Void crt_bt_post(Bitreeptr *bt)
 { if (i>=s.len)   //I 是全局变量，初始值为 0，s 存放后序扩展序列字符和长度
    { bt=stack[top]; top=top-1;} //入栈
   else
     { i++; c=s.ch[i];
       if (c=' ') *bt=Null;
       else { *bt=(BiNode*)malloc(sizeof(BiNode));
              (*bt)->data=c;
              (*bt)->rch=stack[top]; top=top-1;
              (*bt)->lch=stack[top]; top=top-1;
            }
       top=top+1; stack[top]=*bt;
       crt_bt_post(&bt);
     }
   }
```

9．层次遍历某二叉树的扩展序列可以唯一确定二叉树的存储结构

按层次遍历的扩展序列建立二叉树非递归算法（pascal）

```
getnode(var bt: bitreptr)
 begin
 i=i+1; e=s.ch[i];
  if (c=' ') then bt=nil;
  else begin
       new(bt); bt->data=c; que.rear=que.rear+1; que.elem[que.rear]=bt;
       end
 end
proceede crt_bt_level( var bt:bitreptr)
 var p:bitreptr;
begin
   que.front=0;   que.rear=0;   getnode(bt);
   while que.rear<>que.front do
      begin
        que.front=que.front+1;
        p=que.elem[que.front];
        getnode(p->lch); getnode(p->rch);
      end
end
```

10．由二叉树的先序和中序建立二叉树（要求会手工做）

```
Int pos(char ch; char pin[]; int start)   //定位：ch 在中序中的位置
  { i=start;     b=false;
    while ((i<=n)&&!b)
      { if (pin[i]= =ch) b=true;
        else   i++;
      }
    if (b= =true)   return i;
  }
Void Creat(Bitree &T, int pp, int pi, int n)
//pre[]为二叉树的先序序列，pp 为子树在先序序列中的起始下标，ino[]为二叉树
  的中序序列，pi 为子树在中序中的起始下标，n 为树中结点数；
  { if (n<=0)   T=Null;
    else
      { T=(BiTNode*)malloc(sizeof(BiTNode));
        T->data=pre[pp];   m=pos(pre[pp],ino,pi);
        Llen=m-pi;   Rlen=n-Llen-1;   //求 T 的左右子树在中序中的长度
        Create(T->lch,pp+1,pi,Llen);   //建立左子树
        Create(T->rch,pp+Llen+1,m+1,Rlen); //建立右子树
      }
  }
```

11．由二叉树的中序和后序序列建立二叉树

能手工作出，相关程序参阅（04 年程序阅读第一题）老师没有写出程序。

12．按给定的表达式建立相应二叉链表

(1)由先缀表达式建树（与普通二叉树的区别是：无度为 1 的结点，只有度为 0 或 2 的结点，因为操作数有 2 个），可唯一确定二叉树

(2)有原表达式建树，要先将其转化为对应的先缀形式

13．线索二叉树的定义

14．先序遍历二叉数（考填空或选择，书上没有，记住）

(1)采用二叉链表

无论是否是叶子结点，找结点 P 的先序前驱都不容易，得从根结点开始；

若 P 非叶子，找 P 的先序后继，容易（有左孩子的就是其左孩子，无左孩子则是右孩子），不用从根结点开始；若 P 是叶子结点，找先序后继也不容易

(2)采用线索二叉链表，找 P 的先序前驱

(a) P 是根，前驱为空；

(b) P 是其双亲的左孩子，前驱为双亲

(c) P 是其双亲的右孩子，且无左兄，则前驱是其双亲

(d) P 是其双亲的右孩子，且有左兄，则前驱是其左兄最右下的叶子

即使是线索二叉链表，找 P 的先序前驱均不容易

（3）采用线索二叉链表，找 P 的先序后继

(a) P 有右孩子

P 有左孩子，则后继为其左孩子；

P 无左孩子，则后继为其右孩子；

(b) P 无右孩子， 则后继为其 rch;

```
BinNode   *findson(BinNode *p)
```

```
{ if (p->ltag= =0)   return   p->lch; //p 有左孩子，后继为左孩
  else return   p->rch;    //p 无左孩，后继为其右指针
  }
```

15. 中序遍历二叉树

(1). 采用二叉链表

若左右子树非空，找 p 的中序前驱和中序后继容易，不用从根开始

若左右子树为空，找 p 的中序后继和前驱，不容易，得从根开始

16. 后序遍历二叉树

(1). 采用二叉链表

若 P 非叶子，则找 P 的后序前驱，容易，不用从根开始；若 P 是叶子，找后序前驱不容易；

找 P 的后序后继，无论是否是叶子，都不容易，得从根开始

(2)采用线索二叉树，找 P 的后序前驱

(a). P 有左孩子

P 有右孩子，则前驱为其右孩

P 无右孩子，则其前驱为其左孩

(b).P 无左孩子， 前驱为其 lch

```
BinNode   *findpre(BiNode   *p)
  { if (p->rtag= =0) return   p->rch;   //P 有右，前驱为右孩子
    else   return p->lch;   //p 无右孩，前驱为其左指针。
```

(3). 采用线索二叉树，找 P 的后序后继

(a). P 是根， 后继为空

(b). P 是双亲结点的右孩子，后继为双亲

(c). P 是双亲的左孩子，且无右兄，后继为双亲

(d).P 是双亲结点的左孩子，有右兄，后继为右兄最左下的叶子。

后序线索二叉链表中，找后继不容易

17. 树的存储结构

树的二叉链表（孩子-兄弟）存储表示法（必须掌握）

18. 森林转换成二叉树：将各科树分别转换成二叉树，将根结点用线相连，以第一棵树根结点作为树的根，把线顺时针转 45 度

19. 二叉树转化成森林：

一抹线： 将二叉树中的根与其右孩子的连线，及右分支搜索到的所有右孩子的连线全部抹掉，使之变成孤立的二叉树。

一还原： 将孤立的二叉树还原成树

20.树的遍历 （先根，后根）

21.森林的遍历

(1)先序遍历：即依次从左至右对森林中的每一棵树进行先根遍历

(2)中序遍历：即依次从左至右对森林中的每棵树进行后跟遍历

22.树，二叉树，森林遍历的对应关系

树	森林	二叉树
先根	先序	先序
后根	中序	中序

23. 求树的深度的算法

递归公式如下：

```
depth(t)=0                          若 t=Null
depth(t)=max{depth(t->firstchilid)+1, depth(t->nextsibling)}
int TreeDepth(CsTree T)
{ if (!T)   return();
 else {   h1=TreeDepth(T->firstchild);
          h2=TreeDepth(T->nextsibling);
          return(max((h1+1),h2));   //填程序时不能直接写 Max，要写成可执行语
                                      句（用 if   else ）
       }
  return ((h1+1)>h2 ? (h1+1): h2);
 }
```

24.归求树中叶子结点的个数 （04 年真题）

树中的叶子结点是二叉树中 firstchild 为空的结点

解 1：

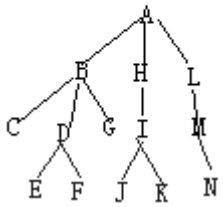
```
int   Countleaf(CsNode *T)
{int leaf; CsNode *T1;
 if (!T)     return 0;
 else {if (!T->fch) return 1;
       else   { leaf=0; T1=T->fch;
                while (T1)
                   {leaf=leaf+Countleaf(T1);
                    T1=T1->nsib;
                   }
                return leaf;
              }
       }
 }
```

解 2：

```
int Countleaf(CsNode *T)
    {   int leaf; CsNode *T1;
        if (!T)   return 0;
        else {   if (!T->fch)
                    { leaf=1+Countleaf(T->nsb);   return(leaf);}
                 else
                    { leaf=Countleaf(T->fch)+Countleaf(T->nsb);   return(leaf)}
                }
    }
```

25. 已知一棵树的层次序列以及各节点的度，建立该树的二叉链表结构。（未考过）

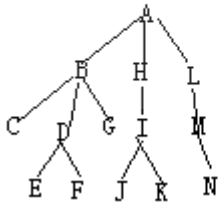
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
数组 a	A	B	H	L	C	D	G	I	M	E	F	J	K	N	//存放结点
数组 b	3	3	1	1	0	2	0	2	1	0	0	0	0	0	//存放结点的度




```
Void   Tree (Cstree   bt,   char   *a, char *b, int n)
    { if (n= =0)   bt=Null;
      else { i=1; qq.front=0; qq.rear=1;   //i 是是数组下标
             p=(CsNode*)malloc(sizeof(CsNode));
             p->data=a[i];    p->du=b[i];   p->snib=Null; //生成第一个结点，右子树必为空
             bt=p;   i++;   qq.elem[qq.rear]=p;
             if (n= =1)    bt->fch=Null;
             else {   while (qq.front!=qq.rear)      //不相等是为空
                        {qq.front++; p=qq.elem[qq.front]; j=p->du;
                         if (j= =0)   p->fch=Null;   //叶子节点时
                         else {while j>0   //兄弟结点加入二叉树，右子树一直往下
                                 { q=(CsNode*)malloc(sizeof(CsNode));
                                   q->data=a[i];   q->du=b[i];
                                   if (j= =p->du)   p->fch=q;
                                   else q1->snib=q;
                                   qq.rear ++;   qq.elem[qq.rear]=q;
                                   q1=q;   i++;   j--;
                                 }
                              }
                      }
             }
    }
```

26. 已知数组 data[1..n]存放一森林的先序序列。在 brother[1..n]中存放每个结点的右兄弟在数组 data[]中的下标，约定 brother[i]中存放 data[i]的右兄弟下标，如不存在右兄弟，其内容为 0，设计算法构造该森林的二叉链表（未考过）

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
data	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
brother	15	8	4	7	6	0	0	12	0	11	0	0	0	0	18	17	0	0



R

```
Void   crt_forest (tlink   &T,   int k1,   int k2)
//T 为第一棵树的树根指针，k1 和 k2 为建森林所对应的数组区域的首尾下标
```

```
{   if (k1>k2)     T=Null;     //对于本例 k1=1,k2=18
    else { T=(csnode*)malloc(sizeof(csnode));
           T->data=data[k1];
           If  (brother[k1]=  =0)        //k1 无右兄弟，则在 data 中从 k+1 到 k2 都是
                                           k1 的子孙
            { crt_forest(T->fch,k1+1, k2); T->nsib=Null;}
           else       //在 data 中从 k+1 到 brother[k1]-1 是 k1 的子孙，从 brother[k1]
                        到 k2 是 k1 的右兄弟分支
            { crt_forest(T->fch, k1+1,brother[k1]-1);
              crt_forest(T->nsib,brother[k1],k2);
            }
         }
  }
```

27. 哈夫曼树（要求会构造）

第七章：图

1．掌握邻接矩阵和邻接表

2．十字链表，邻接多重表不要求编程，但应该能看懂

3．时间复杂度：邻接矩阵 O(n**2)　邻接表 O(n+e)

4．无向连通图：最多有 n(n+1)/2 最少有 n-1 条边

有向连通图：最多有 n(n-1) 最少有 n 条弧

5．最小生成树的两个算法要求掌握

prim 算法书上有；

kruskal 算法

```
//构造无向图的最小生成树，其中用 struct  edges 表示边 bv,  to 表示一条边的两个端
  点，w 表示该边的权值
  #define    Max     100
  struct edges { int bv, tv, w;}
  typedef sturct edges edgeset[Max];
  int seeks(int set[], int v)
   { int i=v;
    while (set[i]>0) i=set[i];
    return(i);
   }
  kruskal(ge,n,e)
{ int n,e; edgeset ge;          //将图中的边按权值由小到大存放在数组 ge 中
 int set[Max], v1,v2,i,j;
 for (i=1;i<=n;i++) set[i]=0; i=1; j=1;
 while (j<n&&i<=e)
   {v1=seek(set, ge[i].bv);
    v2=seek(set, ge[i].tv);
    if (v1!=v2)       //v1=v2 表示这条边的两端在同一个集合中
     { printf("(%d,%d)",ge[i].bv, ge[i].tv);
       set[v1]=v2; j++;
     }
```

```
    i++;
  }
}  //该算法可以自己拿个简单的例子走一遍
```

(1)两种算法的比较：Prim 算法时间复杂度 O(n**2),与边数无关，适合边稠密的图

Kruskal 算法时间复杂度 O(eloge) 适合求顶点稠密的图

(2)最小生成树不唯一，用不同的算法得出的最小生成树不同

6 关节点，重连通图（要求会判断是否是重连通图）

关节点的特性：若生成树根结点有两个子树，其比是关节点

若一棵子树的所有子孙没有指向其祖先的回边

8. 最短路径会手工求就行
9. 拓扑排序会手工排就行

（利用深度优先遍历退栈的逆序列 P183）

若求不出一个图的拓扑排序，则说明该图有回路。

10. 关键路径（会手工做，会求事件和活动的最早最晚时间）
11. Aov:用顶点表示活动，弧表示活动间的优先关系的有向图，趁为顶点表示活动的网

AoE: 顶点表示事件，边表示活动

DaG: 有向无环图，都能进行拓扑排序

第九章：查找

1．顺序查找：既可查顺序表也可查线性表（设监视哨）

2．分析平均查找长度 $ASL=\sum Pi*Ci$ (求和下限 i=1，上限 n)

Pi 为查找表中第 i 个记录的概率 Ci=n-i+1

在等概率的情况下 Pi=1/n ASL=(n+1)/2

在不等概率的情况下要根据公式分别求出概率 Pi 和 Ci;

3．有序查找表—折半查找

有序顺序表才可用折半查找，折半查找的判定树一定要会。

表长确定，平均查找长度确定，判定树的拓扑结构确定

n 个结点的判定树深度为 $\lfloor \log2(n) \rfloor +1$。

折半查找法在查找过程中进行的比较次数最多不超过其判定树的深度。

设表长为 n=(2**h) –1 ,查找成功的概率相等，不成功的概率为 0，采用二分法查找，问 ASL 是多少？判定树是深度为 h 的满二叉树。

ASL=1/n(1*2**(1-1)+2*2**(2-1)+3*2**(3-1)….+h*2**(h-1))

令 S=1/n(1*2**0+2*2**1+3*2**2+….+h*2**(h-1))

S=2S-S=1/n[h*2**h-(2**0+2**1+….2**(h-1)) //其中小括号正好为深度为 h 的满二叉树的总结点数。

=1/n[h*2**h-2**h +1]=1/n[(h-1)*2**h+1]=1/n[(log2(n+1)-1)*(n-1)+1]

=1/n[(n+1)log2(n+1)-n]=(n+1)/nlog2(n+1)-1

//熟悉上述的推导方法

4．分块查找

(1)适用条件：分块有序表

分块查找方法的评价 P226

都采用顺序查找，在等概率的情况下分几块平均查找长度最小(参看教材)

5．设有关键字为 k1….kn,查找成功的概率为 2**-1, 2**-2 ….2**-n，查找不成功的概

率为 2**-n, (1)若从左向右顺序查找，ASL 是多少？
(2)从右向左顺序查找，ASL 是多少？
（3 总的 ASL 是多少？

解：（1）ASL=(∑i*2**-I)+(n+2**-n) // 其中求和的下限是 i=1,上限是 n。第一部分是查找成功的 ASL，第二部分是查找不成功的 ASL

=1*（2**-1）+2*（2**-2）+….+n*(2**-n)+n*(2**-n)

令 S=1* (2**-1)+….+n*(2**-n)

S=2S-S=1+(2**-1)+(2**-2)+……+(2**-(n-1))-n*(2**-n)

=2-(2**-(n-1))-n*(2**-n)

则 ASL=S+n*(2**-n)=2-(2**-(n-1)) 时间复杂度为 O(1),当 n 趋向 0 时 ASL=2

(2)ASL=(∑(n+1-i)*(2**-i) + (n*(2**-n))) 其中求和的下限是 i=1,上限是 n

=n*(2**-1)+(n-1)*(2**-2)+….+(2**-n)+n*(2**-n)

令 S=n*(2**-1)+….(2**-n)

S=2S-S=n-(1-(2**-n))

则 ASL=n+(n+1)*(2**-n)-1 最坏的情况下时间复杂度为 O(n)

(3)总的 ASL 取最小值 ASL=2-(2**-(n-1))

6．动态查找是重点

7．二叉排序树中的插入和删除要会

删除一个结点后，二叉树的中序遍历仍是递增

会求二叉排序树查找的平均 ASL(要分清查找成功的结点和查找失败结点的(外结点)的 ASL)

8．设有关键字 n=(2**h)，查找成功的概率相等，构成二叉排序树，ASL 最大多少？最大树高是多少？

解：高为 n 时，ASL 最大(1=2=3+…+n)/n=(n+1)/2

深度为 h 的满二叉树时，ASL 最小：((n+1)/n)*log2(n+1)-1

9．设有 1023 个关键字的二叉排序树，查找成功的概率相等，使 ASL 最小时树高是多少？

根据第 8 题的公式((n+1)/n)*log2(n+1)-1＝(1024/1023)log2(1024)- 1=10240/1023-1

所以树高 h=10

10．掌握如何构造平衡二叉树（多做几次练习，参看光盘的数据结构课件）

如何选旋转轴？选离插入点最近的不平衡点的儿子结点

11．具有 n 个叶子结点的非满二叉树的完全二叉树深度为（$\lceil \log2(n+1) \rceil$+1）

具有 n 个叶子结点的满二叉树的深度为（log2(n) + 1 ）

具有 n 个结点的完全二叉树的深度（$\lfloor \log2(n) \rfloor$ +1）

具有 n 个结点的折半查找判定树的深度为（$\lfloor \log2(n) \rfloor$ +1）

12．含 n 个关键字的平衡二叉树的最大深度为？

	n=0	n=1	n=2	n=4	n=7
	空树				
最大深	0	1	2	3	4

反过来，深度为 h 的二叉平衡树中所含结点的最小值多少？

h=0, N0=0; h=1,N1=1; h=2,N2=2; h=3,N3=4

一般情况： Nh=N(h-1)+N(h-2)+1

利用归纳法证得：Nh=F(h+2)-1 P 书 238

当平衡二叉树含 20 个结点，高度最高是多少？

N4=2+4+1=7, N5=4+7+1=12, N6=7+12+1=20

所以 h=6

1 3．B 树： 定义，查找过程，插入，删除必须掌握（多做点相关的题目）

高度为 k 的 2-3 树的结点数至少是： 二叉排序树 (2**k)– 1

最多有多少个结点：每个结点含两个关键字最多有：Σ3**(i-1)=[(3**k)-1]/2 求和下限是 i=1,上限是 k

高为 h 的 m 阶 B-树的结点数至少是：t=$\lceil m/2 \rceil$

层	结点	
1	1	
2	2	
3	2t	所有结点相加=1+2*Σ(t**i) 求和下限 i=0, 上限为 k-2
4	2*(t**2)	=1+2*[(t**(h-1)-1)/(t-1)]
..	..	
h	2*(t**(h-2))	

高为 h 的 m 阶 B-树的结点数最多有多少：

层	结点	
1	1	
2	m	结点相加=Σ(m**i) 求和下限 i=0, 上限为 k-1
3	m**2	=[(m**k)-1]/(m-1)
..	..	
n	m**(n-1)	

1 4．高度为 h 的 m 阶 B 树的关键字数最多是((m**h)-1)，至少是(2*[t**(h-1)]-1)，

其中 t=$\lceil m/2 \rceil$

分析：最少，(m-1)* Σ[m**(i-1)] 求和下限 i=1, 上限为 h

=(m-1)*[(m**h) -1]/(m-1)

=(m**h)-1

最多，1+2(t-1) Σ[m**(i-2)] 求和下限 i=2, 上限为 h

=1+2(t-1)*[t**(h-1) -1]/(t-1)

=2*[t**(h-1)]-1

1 5．N 个结点的 m 阶 B-树至少含(($\lceil m/2 \rceil$-1)*(n-1)+1)个关键字 //注意好好听老师的讲解，了解式中的每项是怎么得来的

1 6．B 树中的空指针数总比关键字数多 1

证明：

层次	该层结点	该层关键字	该层结点指针数
1	1	k1	k1+1
2	k1+1	k2	k1+k2+1
3	k1+k2+1	k3	k3+k2+k1+1
…	…	…	…

h k1+….+k(n-1)+1 kn

则第 h 层是叶子结点层，无关键字。空指针=叶子结点数= k1+….+k(n-1)+1

第一层到第(h-1)层关键字总数为 k1+….+k(n-1) 于是命题得证

17．**B**+树掌握概念就行，既适合动态查找也适合顺序查找

18．**哈希表：**概念，构造方法，处理冲突的办法，同义词，查找成功和不成功时的平均查找时间，二次聚集等均要掌握

19．**再哈希：**（参看 04 年作图题）

20．**链地址：**成功 ASLsuc=[(第一列的结点数*1)+（二列结点数*2）….]/表长

不成功 ASL=个单链表表长之和/表长

21．线性探测再散列：

0	1	2	3	4	5	6	7	8	9	10
	4		12	49	38	13	24	32	31	

ASLsuc=(5*1+3*2)/8=11/8

ASLunsuc=(1+2+1+8+7+6+5+4+3+2+1)/11 (可以参看相关的习题中的练习，熟悉其计算方法)

第十章：1.所有的排序算法均要掌握

不稳定的简单排序—简单选择排序

2. 对不稳定算法，要求能举出例子说明为什么不稳定。基数排序应用在整型和字符型的情况下。Shell 排序的复杂度不要求掌握

3. 在堆排序中，想得到一个递增序列用大顶堆，反之用小顶堆

历届真题----大题

1.<99> 图的深度优先遍历

2.<99> 以孩子兄弟链表为存储结构，设计递归和非递归算法求树的深度

3.<01> 设计算法将不带头结点的单链表逆置

4.<01> 设计算法按层次顺序遍历二叉树

应参考历届的算法，注意总结！推荐两本习题均是人邮出版的。〈数据结构课程辅导与习题解析〉和〈数据结构 500 题〉，感觉很适合考北交，书中习题大多数是针对考研来设计的，而且分析很详细。大家应注意理解解题的过程，这是最主要的！碰到没见过的题目，或者经常容易犯错的地方，还有就是经典的算法等最好用专门的本子记下来，这样也可加深记忆，可经常翻阅！我去年基本上就是按这些资料复习的，其于的我也只做了清华大学李春葆的习题和西电的考研习题，感觉一般。这次辅导班老师讲了很多以前考过的和没考过的算法，大家一定要全部认真理解，不管考过没考过都很有可能会出现在 06 年的试题中。还有就是请大家关注 06 的辅导班，这很重要。

由于从没用过 Word，所以编辑有点烂，希望大家能原谅，到时自己去修正了！