数据结构（C 语言版） 期末复习汇总

第一章 绪论

数据结构：是一门研究非数值计算程序设计中的操作对象，以及这些对象之间的关系和操作的学科。

数据结构是一门综合性的专业课程，是一门介于数学、计算机硬件、计算机软件之间的一门核心课程。是设计和实现编译系统、操作系统、数据库系统及其他系统程序和大型应用程序的基础。

数据：是客观事物的符号表示，是所有能输入到计算机中并被计算机程序处理的符号的总称。如数学计算中用到的整数和实数，文本编辑中用到的字符串，多媒体程序处理的图形、图像、声音及动画等通过特殊编码定义后的数据。

数据的逻辑结构划分：**线、树、图**

算法的定义及特性

算法：是为了解决某类问题而规定的一个有限长的操作序列。

五个特性：**有穷性、确定性、可行性、输入、输出**

评价算法优劣的基本标准（4 个）：

正确性、可读性、健壮性、高效性及低存储量

第二章 线性表

线性表的定义和特点：

线性表：由 n(n≥0)个数据特性相同的元素构成的有限序列。线性表中元素个数 n(n≥0)定义为线性表的长度，n=0 时称为空表。

非空线性表或线性结构，其特点：

（1）存在唯一的一个被称作**“第一个”**的数据元素；

（2）存在唯一的一个被称作**“最有一个”**的数据元素；

（3）除第一个之外，结构中的每个数据元素均只有一个**前驱**；

（4）除最后一个之外，结构中的每个数据元素均只有一个**后继**。

顺序表的插入：n 个元素在 i 位插入，应移动**（n-i+1）**位元素。

顺序表存储结构的优缺点：

优点：

逻辑相邻，物理相邻；可随机存取任一元素；存储空间使用紧凑；

缺点：

插入、删除操作需要移动大量的元素；预先分配空间需按最大空间分配，利用不充分；

表容量难以扩充；

线性表的应用：

一般线性表的合并：★★★

算法 2.1：LA=(7,5,3,11)　LB=(2,6,3)

合并后　LA=(7,5,3,11,2,6)

算法思想：扩大线性表 LA，将存在于线性表 LB 中而不存在于线性表 LA 中的数据元素插入到线性表 LA 中去。只要从线性表 LB 中依次取得每个数据元素，并依值在线性表 LA 中进行查访，若不存在，则插入之。

有序表的合并：★★★

算法 2.2：LA=(3,5,8,11)　LB=(2,6,8,9,11,15,20)

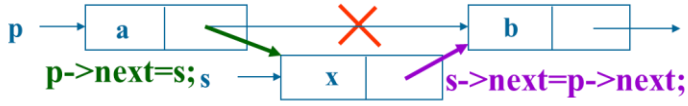
则 LC=(2,3,5,6,8,9,11,11,15,20)

算法思想：首先创建一个空表 LC，通过比较指针 pa 和 pb 所指向的元素的值，依次从 LA

或 LB 中“摘取”元素值较小的结点插入到 LC 表的最后，当其中一个表变空是，说明此表的元素已归并完，则只要将另一个非空表的剩余结点依次插入在 LC 表的最后即可。

线性链表：

线性链表的插入：插入元素师，指针的指向变化：



上述指针修改用语句描述即为：**s->next=p->next; p->next=s;**

单链表的插入：★★★

```
void   insertnode(linklist   head，datetype   x，int   i){
          listnode   * p，*q;
          p=getnode(head，i-1);
          if(p==NULL)
               error(〝position   error〞);
           q=(listnode *)
          malloc(sizeof(listnode));
           q–>data=x;
           q–>next=p–next;
            p–>next=q;
        }
```

课本中：

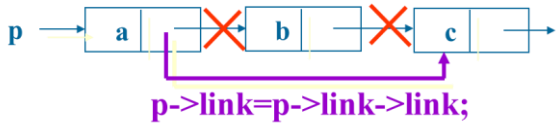
s=new LNode;

s->data =e;

s->next=p->next;

p->next=s;

线性链表的删除：删除元素，指针的指向变化：



上述指针修改用语句描述即为：**p->next=p->next->next;**

单链表的删除：★★★

```
void   deletelist(linklist head，   int i)
    {   listnode * p，   *r;
        p=getnode(head，i-1);
        if(p= =NULL ||
           p–>next= =NULL)
        return ERROR;
        r=p–>next;
        p–>next=r–>next;
         free( r ) ;     }
```

课本中：

q=p->next;

p->next=q->next;

单链表特点：

它是一种动态结构，整个存储空间为多个链表共用；

不需预先分配空间；

指针占用额外存储空间；

不能随机存取，查找速度慢。

第三章 栈和队列

栈的类型定义：

栈(Stack)是限制在表的一端进行插入和删除运算的线性表，通常称插入、删除的这一端为栈顶(Top)，另一端为栈底(Bottom)。当表中没有元素时称为空栈。

假设栈 S=(a1，a2，a3，…an)，则 **a1 称为栈底元素，an 为栈顶元素**。栈中元素按 a1，a2，a3，…an 的**次序进栈**，退栈的第一个元素应为栈顶元素。即，栈的修改是按后进先出的原则进行的。因此，栈称为**后进先出**表（LIFO）。

栈的进栈、出栈顺序：

例：★★★

对于一个栈，给出输入项 A、B、C，如果输入项序列由 ABC 组成，试给出所有可能的输出序列。

A 进 A 出 B 进 B 出 C 进 C 出	ABC
A 进 A 出 B 进 C 进 C 出 B 出	ACB
A 进 B 进 B 出 A 出 C 进 C 出	BAC
A 进 B 进 B 出 C 进 C 出 A 出	BCA
A 进 B 进 C 进 C 出 B 出 A 出	CBA

不可能产生输出序列 CAB

栈的应用：数值转换[大题]

算法思想：首先将按照上述计算过程中得到的八进制树的各位依次进栈，然后将栈中的八进制数依次出栈输出，输出结果就是该是十进制数转换得到的八进制数。

N=(N div d)×d + N mod d （其中：div 为整除运算，mod 为求余运算）

例如 (1348)10=(2504)8，其运算过程如下：

N	N div 8	N mod 8
1348	168	4
168	21	0
21	2	5
2	0	2

算法：【此为课件算法同课本算法一致】★★★

```
void conversion( ) {
        initstack ( s );//构造空栈
        scanf ( " % " , n);
        while( n ){
                push ( s , n % 8 );
                n = n / 8;
            }
        while( ! Stackempty ( s )){//当栈非空时
                pop ( s , e );
                printf ( " % d " , e );} }//conversion
```

队列的类型定义：

定义：队列是限定只能在表的一端进行插入，在表的另一端进行删除的线性表

队尾(rear)：允许插入的一端

队头(front)：允许删除的一端

队列特点：先进先出（FIFO）

第四章 串、数组和广义表

串的类型定义：串是字符串的简称。它是一种在数据元素的组成上具有一定约束条件的线性表，即要求组成线性表的所有数据元素都是字符，所以，人们经常又这样定义串：串是一个有穷字符序列。

串一般记作：

s= “a1a2...an”　　(n≥0)

其中，s 是串的名称，用双引号（“ ”）括起来的字符序列是串的值；ai 可以是字母、数字或其他字符；串中字符的数目 n 被称作串的长度。当 n=0 时，串中没有任何字符，其串的长度为 0，通常被称为空串。

子串、主串：串中任意连续的字符组成的子序列被称为该串的子串。包含子串的串又被称为该子串的主串。

广义表的定义：

广义表是 n(n>=0)个元素 a1,a2,a3,…,an 的有限序列，其中 ai 或者是原子项，或者是一个广义表。通常记作 LS=（a1,a2,a3,…,an)。LS 是广义表的名字，n 为它的长度。若 ai 是广义表，则称它为 LS 的子表。

广义表的结构特点:

1) 广义表的元素可以是子表，而子表的元素还可以是子表……广义表是一个多层次的结构
2) 广义表可为其他广义表所共享
3) 广义表可以是一个递归的表，即广义表也可以是其本身的一个子表

广义表的两个重要运算：取表头 GetHead(LS)，取表尾 GetTail(LS)

任何一个非空广义表

LS = (1, 2, …, n)

均可分解为

表头 Head(LS) = 1

和

表尾 Tail(LS) = (2, …, n)

两部分。

例如：★★★

已知广义表 LS=（a，（b，c，d），c），运用 GetHead 和 GetTail 函数取出原子 d 的运算过程为：GetHead(GetTail(GetTail(GetHead(GetTail(LS)))))

第五章 树和二叉树

数的定义：

树(tree)是 n(n≥0)个结点的有限集 T，其中：

有且仅有一个特定的结点，称为树的根(root)；

当 n>1 时，其余结点可分为 m(m>0)个互不相交的有限集 T1,T2,……Tm，其中每一个集合本身又是一棵树，称为根的子树(subtree)。

特点：

树中至少有一个结点：根

树中各子树是互不相交的集合

线性结构与树形结构的区别

线性结构	树形结构
第一个数据元素（无前驱）	根结点 （无前驱）
最后一个数据元素（无后继）	多个叶子结点（无后继）
其它数据元素(一个前驱,一个后继）	其它数据元素（一个前驱、多个后继）

树的基本术语：

结点(node) ：表示树中的元素，包括数据元素及若干指向其子树的分支

结点的度(degree) ：结点拥有的子树数

树的度 ： 一棵树中最大的结点度数

叶子(leaf) ：度为 0 的结点（终端结点）

孩子(child) ：结点子树的根称为该结点的~~

双亲(parents) ：孩子结点的上层结点叫该结点的~~

兄弟(sibling) ：同一双亲的孩子

子孙 ： 以某结点为根的子树中的所有结点都被称为是该结点的~~

祖先 ：从根结点到该结点路径上的所有结点

堂兄弟 ：双亲在同一层的结点互为~~

结点的层次(level) ： 从根结点算起，根为第一层，它的孩子为第二层……

深度(depth) ： 树中结点的最大层次数

森林(forest) ： m(m 0)棵互不相交的树的集合

有序树、无序树 ： 如果树中每棵子树从左向右的排列拥有一定的顺序，不得互换，则称为有序树，否则称为无序树

对于课本 P96 树形图：★★★

结点 A 的度：3 结点 B 的度：2 结点 M 的度：0	结点 A 的层次：1 结点 M 的层次：4
叶子：K，L，F，G，M，I，J	结点 B，C，D 为兄弟 结点 K，L 为兄弟
结点 A 的孩子：B，C，D 结点 B 的孩子：E，F	结点 I 的双亲：D 结点 L 的双亲：E
结点 F，G 为堂兄弟 结点 A 是结点 F，G 的祖先	树的深度：4 树的度：3

二叉树的定义：

二叉树是 n(n≥0)个结点的有限集，它或为空树(n=0)，或由一个根结点和两棵分别称为左子树和右子树的互不相交的二叉树构成。

特点

每个结点至多有二棵子树(即不存在度大于 2 的结点)；

二叉树的子树有左、右之分，且其次序不能任意颠倒。

二叉树的性质：

性质 1：在二叉树的第 i 层上最多有 2i-1 个结点（i≥1）；

性质 2：深度为 K 的二叉树最多有 2K-1 个结点 （K≥1）；

性质 3：对任何一棵二叉树 T，如果其终端结点数为 n0，度为 2 的结点数为 n2，则 n0=n2+1；

完全二叉树

定义：一棵深度为 h，具有 n 个结点的二叉树，若将它与一棵同深度的满二叉树中的所有结点按从上到下，从左到右的顺序分别进行编号，且该二叉树中的每个结点分别与满二叉树中

编号为 1~n 的结点位置一一对应，则称这棵二叉树为**完全二叉树**。

特点：

叶子结点只可能在层次最大的两层上出现；

对任一结点，若其右分支下子孙的最大层次为 L，则其左分支下子孙的最大层次必为 L 或 L+1。

性质 4：具有 n 个结点的完全二叉树的深度为⌊log2n⌋+1。（其中，⌊log2n⌋ 的结果是不大于 log2n 的最大整数。）

性质 5：对于有 n 个结点的完全二叉树中的所有结点按从上到下，从左到右的顺序进行编号，则对任意一个结点 i （1≤i≤n），都有：

（1）如果 i=1，则结点 i 是这棵完全二叉树的根，没有双亲。否则其双亲结点的编号为 ⌊i/2⌋；

（2）如果 2i>n，则结点 i 没有左孩子。否则其左孩子结点的编号为 2i；

（3）如果 2i+1>n，则结点 i 没有右孩子。否则其右孩子结点的编号为 2i+1。

遍历二叉树：★★★

先序遍历：先访问根结点,然后分别先序遍历左子树、右子树【包络法】

中序遍历：先中序遍历左子树，然后访问根结点，最后中序遍历右子树【垂直映射法】

后序遍历：先后序遍历左、右子树，然后访问根结点

例：（1）

先序遍历：A.B.D.E.F.G.K.M.C.H.J

中序遍历：D.B.F.E.K.A.M.G.H.C.J **P105 贴纸（1）**

后序遍历：D.F.K.M.A.E.B.H.J.C.G

层次遍历：A.B.C.D.E.H.J.F.G.K.M

例：（2）

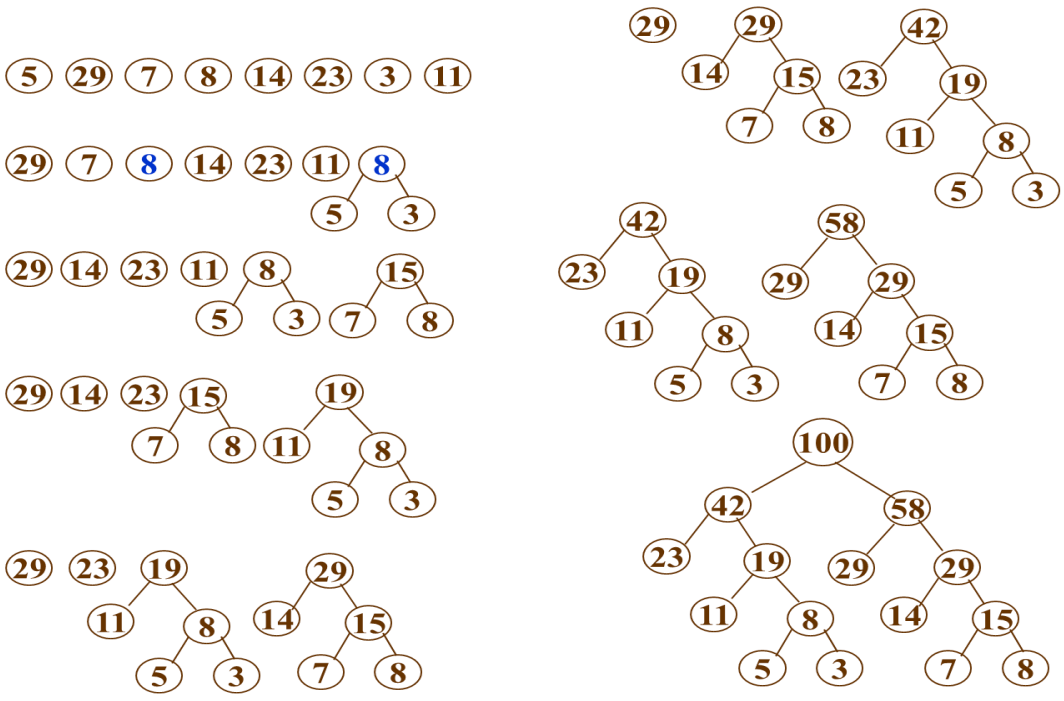
先序遍历：A.B.E.F.I.G.C.D.H.J.K.L.N.O.M

后序遍历：E.I.F.G.B.C.J.K.N.O.L.M.H.D.A **P105 贴纸（2）**

层次遍历：A.B.C.D.E.F.G.H.I.J.K.L.M.N.O

赫夫曼树遍历：★★★

例：W={ 5，29，7，8，14，23，3，11 }



第六章 图

图的定义：图(Graph)：图 G 是由两个集合 V(G)和 E(G)组成的,记为：G=(V,E)

其中：V(G)是顶点的非空有限集；E(G)是边的有限集合，边是顶点的无序对或有序对。

有向图，无向图

顶点的度：无向图中，顶点的度为与每个顶点相连的边数；

有向图中，顶点的度分成入度与出度

入度：以该顶点为头的弧的数目

出度：以该顶点为尾的弧的数目

路径：路径是顶点的序列 V={Vi0,Vi1,……Vin}，满足(Vij-1,Vij) ∈E 或 <Vij-1,Vij>∈E,(1<j≥n)

路径长度：沿路径边的数目或沿路径各边权值之和

回路：第一个顶点和最后一个顶点相同的路径叫~

简单路径：序列中顶点不重复出现的路径叫~

简单回路：除了第一个顶点和最后一个顶点外，其余顶点不重复出现的回路叫~

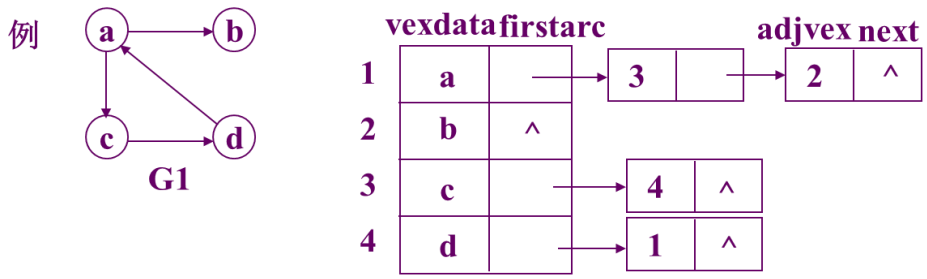
连通：从顶点 V 到顶点 W 有一条路径，则说 V 和 W 是连通的

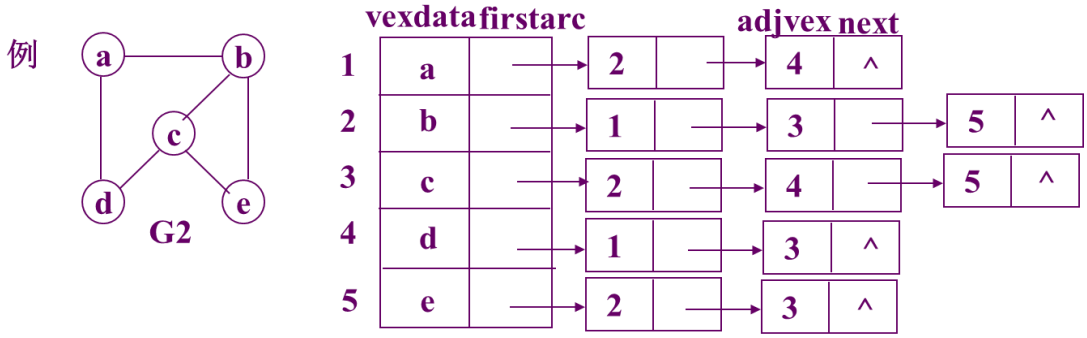
连通图：图中任意两个顶点都是连通的叫~

连通分量：非连通图的每一个连通部分叫~

强连通图：有向图中，如果对每一对 Vi,Vj∈V, Vi∈Vj,从 Vi 到 Vj 和从 Vj 到 Vi 都存在路径，则称 G 是~

邻接表：

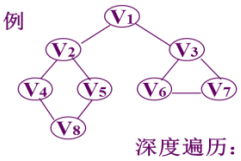




深度遍历：

方法：从图的某一顶点 V0 出发，访问此顶点；然后依次从 V0 的未被访问的邻接点出发，深度优先遍历图，直至图中所有和 V0 相通的顶点都被访问到；若此时图中尚有顶点未被访问，则另选图中一个未被访问的顶点作起点，重复上述过程，直至图中所有顶点都被访问为止。

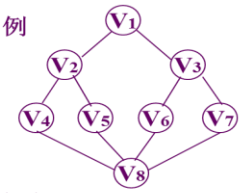
例



深度遍历：V1⇒V2 ⇒V4 ⇒ V8 ⇒V5 ⇒V3 ⇒V6 ⇒V7

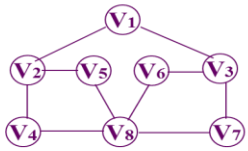
图一

例



深度遍历：V1⇒V2 ⇒V4 ⇒ V8 ⇒V5 ⇒V6 ⇒V3 ⇒V7

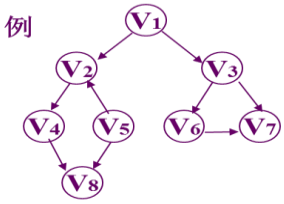
例



深度遍历：V1⇒V2 ⇒V4 ⇒ V8 ⇒V5 ⇒V6 ⇒V3 ⇒V7

图二

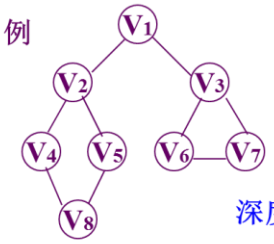
例



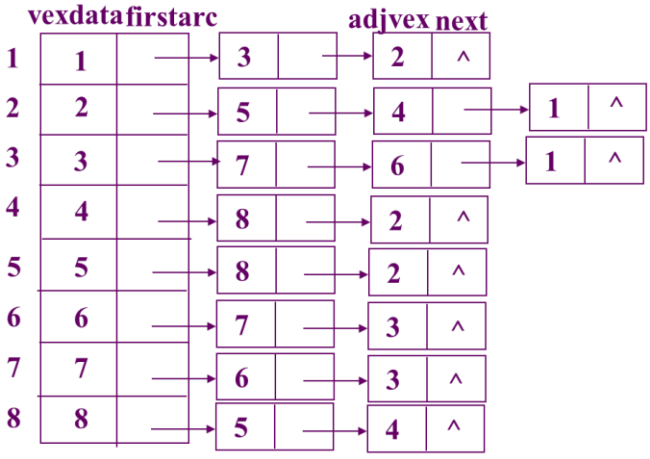
深度遍历：V1⇒V2 ⇒V4 ⇒ V8 ⇒V3 ⇒V6 ⇒V7 ⇒V5

图三

例



深度遍历：V1⇒V3 ⇒V7 ⇒V6 ⇒V2 ⇒V5 ⇒V8 ⇒V4



图四

必须说明，若不给定存储结构，深度优先遍历的结果不唯一。因为哪个顶点是第一邻接点未确定。给定存储结构后，深度优先遍历的结果是唯一的。

广度遍历：【不考，比较深度遍历记忆】

方法：从图的某一顶点 V0 出发，访问此顶点后，依次访问 V0 的各个未曾访问过的邻接点；然后分别从这些邻接点出发，广度优先遍历图，直至图中所有已被访问的顶点的邻接点都被访问到；若此时图中尚有顶点未被访问，则另选图中一个未被访问的顶点作起点，重复上述过程，直至图中所有顶点都被访问为止。

图一中：广度遍历：V1⇒ V2 ⇒V3 ⇒ V4 ⇒V5 ⇒V6 ⇒V7 ⇒V8

第七章 查找

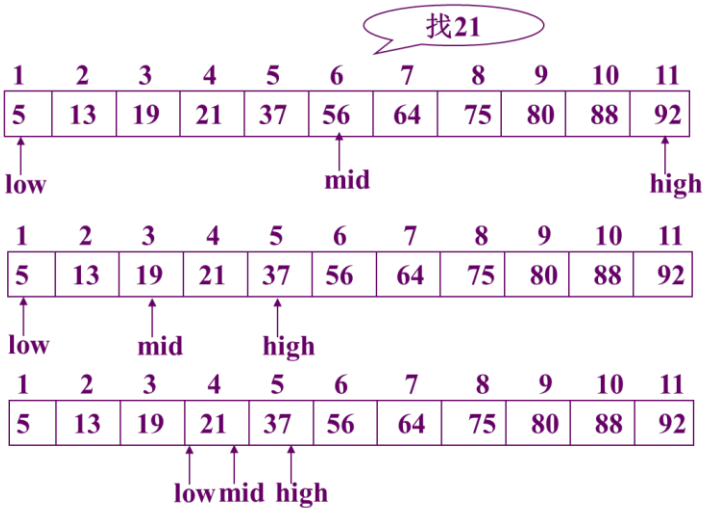
折半查找法（考图形过程）：★★★

（1）有序表：{ 5,16,20,27,30,36,44,55,60,67,71}

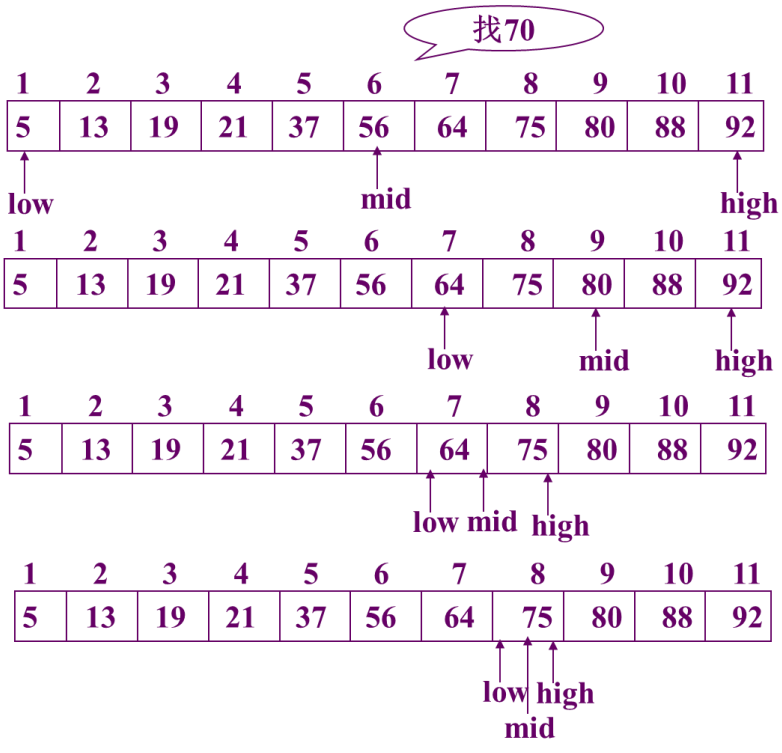
课本 P167，例 7.1 P168 图 7.1

（2）折半查找过程示例：有序表{5,13,19,21,37,56,64,75,80,88,92}

例



例

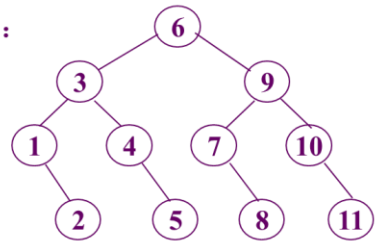


1	2	3	4	5	6	7	8	9	10	11
5	13	19	21	37	56	64	75	80	88	92

high low

1	2	3	4	5	6	7	8	9	10	11
5	13	19	21	37	56	64	75	80	88	92

判定树：



第八章 排序

★★★ **冒泡排序：**

排序过程：

将第一个记录的关键字与第二个记录的关键字进行比较，若为逆序 r[1].key>r[2].key，则交换；然后比较第二个记录与第三个记录；依次类推，直至第 n-1 个记录和第 n 个记录比较为止——第一趟冒泡排序，结果关键字最大的记录被安置在最后一个记录上

对前 n-1 个记录进行第二趟冒泡排序，结果使关键字次大的记录被安置在第 n-1 个记录位置

重复上述过程，直到“在一趟排序过程中没有进行过交换记录的操作”为止。

例

初始关键字	第一趟	第二趟	第三趟	第四趟	第五趟	第六趟
38	38	38	38	13	13	13
49	49	49	13	27	27	27
65	65	13	27	30	30	30
76	13	27	30	38	38	
13	27	30	49	49		
27	30	65	65			
30	76	76				
97	97					

课件上的算法：

```
void BubbleSort(int a[], int n)
{
```

```
    int i, j, exchange;
    int tmp;
    for (i = 0; i < n-1; i++)
{   exchange=0;
    for (j = n-1; j>i; j--)    //比较，找出最小关键字的记录
    if (a[j] < a[j -1])
    {
        tmp = a[j];                    //a[j]与 a[j-1]进行交换，将最小关键字记录前移
        a[j] = a[j - 1];
        a[j -1] = tmp;
        exchange=1;
    }
    if (exchange==0)       //本趟未发生交换时结束算法
        return;   }}
```

课本上的伪码算法：

```
void BubbleSort(SqList &L){
//对顺序表 L 做冒泡排序
m=L.length-1;flag=1;
while((m>0)&&(flag==1)){
flag=0;//flag 置为 0，如果本趟排序没有发生交换，则不会执行下一趟排序
for(j=1;j<=m;j++)
  if(L.r[j].key>L.r[j+1].key){
flag=1;//flag 置为 1，表示本趟排序发生了交换
t=L.r[j];L.r[j]=L.r[j+1];L.r[j+1]=t;//交换前后两个记录
}//if
--m;
}//while
}//BubbleSort
```

C 语言冒泡排序法：

```
#include<stdio.h>
void main()
{
    int i,j,m, a[10];
    printf("用冒泡排序法对 a 数组 10 个数组元素从小到大进行排序。\n");
    printf("Enter the numbers:\n");
    for(i=0;i<10;i++)
        scanf("%d",&a[i]);
    for(i=0;i<9;i++)
        for(j=0;j<10-i;j++)
            if(a[j+1]<a[j])
            {m=a[j];a[j]=a[j+1];a[j+1]=m;}
            for(i=0;i<10;i++)
                printf("%5d",a[i]);
}
```