蜂考速成课

《数据结构》

C 语言版

课时一　数据结构和算法

考点	重要程度	占分	题型
1. 数据结构	★★★	2～4	选择、填空、判断
2. 算法	★★★		

1. 数据结构

（1）数据是对客观事物的符号表示，如图像、声音等。

（2）数据元素是数据的基本单位。

（3）数据项是构成数据元素的不可分割的最小单位。

一个数据元素可由若干个数据项组成，例如，一位学生的信息记录为一个数据元素，它是由学号、姓名、性别等数据项组成。

（4）数据对象是具有相同性质的数据元素的集合。

（5）数据结构是相互之间存在一种或多种特定关系的数据元素的集合。

数据结构包括三方面的内容：逻辑结构、存储结构和数据的运算。

数据结构的形式定义为：数据结构是一个二元组

$$Data\_Structure=(D,S)$$

其中：D 是数据元素的有限集，S 是 D 上关系的有限集。

题 1. 以下说法正确的是（　　）。

A. 数据项是数据的基本单位　　*B*. 数据元素是数据的最小单位

C. 数据结构是带结构的数据项的集合　　*D*. 数据元素可由若干数据项组成

答案：*D*

题 2. 数据的逻辑结构从形式上可用二元组 (D,R) 表示，其中 R 是（　　）的有限集。

A. 算法　　*B*. 数据元素　　*C*. 数据项　　*D*. 数据关系

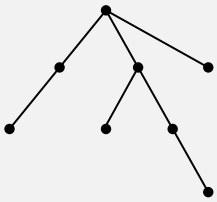
答案：*D*

逻辑结构是指数据元素之间的逻辑关系，与数据的存储无关，独立于计算机。

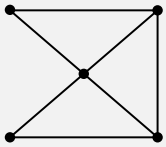
- 逻辑结构
  - 线性结构：　线性表（一般线性表、栈、队列、串、数组）　　一对一
  - 非线性结构
    - 集合
    - 树形结构　　一对多
    - 图形结构（网状结构）　　多对多

集合

线性结构



树形结构



图形结构

题 1. 以下数据结构中，不是线性结构的是（　　）。

A. 栈　　*B.* 队列　　*C.* 串　　*D.* 二叉树

答案：*D*

题 2. 根据数据元素之间关系的不同特性，通常由四类基本结构，即：集合、线性结构、树形结构和_____结构。

答案：图形或网状

存储结构（物理结构）是指数据结构在计算机中的表示，它包括数据元素的表示和关系的表示。

- 存储结构
  - 顺序存储
  - 链式存储
  - 索引存储
  - 散列存储

顺序存储：把逻辑上相邻的元素存储在物理位置上也相邻的存储单元中，元素之间的关系由存储单元的邻接关系来体现。
链式存储：借助指示元素存储地址的指针来表示元素之间的逻辑关系。
索引存储：在存储元素信息的同时，还建立附加的索引表。
散列存储：根据元素的关键字直接计算出该元素的存储地址，又称哈希（*hash*）存储。

题 1. 在顺序存储、链式存储、索引存储和散列存储这 4 种存储方式中，最基本、最常用的两种存储结构是_____和_____。

答案：顺序存储，链式存储

题 2. 以下与数据的存储结构无关的术语是（　　）。

A. 有序表　　*B*. 链表　　*C*. 顺序队列　　*D*. 哈希表

答案：*A*

2. 算法

算法是对特定问题求解步骤的一种描述，它是指令的有限序列，其中的每条指令表示一个或多个操作。此外，一个算法还具有下列 5 个重要特性：

（1）有穷性。一个算法必须总是在执行有穷步后结束，且每一步都是在有穷时间内完成。
（2）确定性。算法中每条指令必须有确切的含义，且相同的输入只能得到相同的输出。
（3）可行性。算法中描述的操作都可以通过已经实现的基本运算执行有限次来实现。
（4）输入。一个算法有零个或多个输入。
（5）输出。一个算法有一个或多个输出。

题 1. 以下不属于算法特性的是（　　）。

A. 可行性　　*B*. 输入　　*C*. 确定性　　*D*. 健壮性

答案：*D*

通常设计一个“好”的算法应考虑达到以下目标：
（1）正确性。算法应能够正确地求解问题。
（2）可读性。算法能具有良好的可读性，以帮助人们理解。
（3）健壮性。输入非法数据时，算法能适当地做出反应或进行处理，而不会产生莫名其妙的输出结果。
（4）效率与低存储量需求。效率是指算法执行的时间，存储量需求是指算法执行过程中所需的最大存储空间。

算法效率的度量是通过时间复杂度和空间复杂度来描述的。
一个语句的频度是指该语句在算法中被重复执行的次数。
算法中所有语句的频度之和记为$T(n)$，它是该算法问题规模n的函数，时间复杂度主要分析$T(n)$的数量级。

题 1. 算法分析的目的是（　　）。

A. 找出数据结构的合理性　　*B*. 研究算法中的输入和输出

C. 分析算法的效率以求改进　　*D*. 分析算法的易懂性

答案：*C*

题 2. 若一个算法中的语句频度之和为$T(n)=n+4n\log_2 n$，则算法的时间复杂度为_____。

答案：$O(n\log_2 n)$

加法规则：$T(n)=T_1(n)+T_2(n)=O(f(n))+O(g(n))=O(\max(f(n),g(n)))$

乘法规则：$T(n)=T_1(n)\times T_2(n)=O(f(n))\times O(g(n))=O(f(n)\times g(n))$

常见的渐近时间复杂度为

$$O(1)<O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<O(n^3)<O(2^n)<O(n!)<O(n^n)$$

题 3. 下面两段程序的时间复杂度是（　　）。

(1)
```
for(i=2;i<=n;++i)
    ++x;
    for(j=1;j<=i-1;++j)
        a[i][j]=x;
```

(2)
```
i=1;
while(i<=n)
    i=i*2;
```

A. $O(n^2),O(\log_2 n)$　　*B*. $O(n^2),O(n^2)$

C. $O(n),O(n)$　　*D*. $O(n),O(\log_2 n)$

答案：*A*

课时一　练习题

1. 数据结构被形式的定义为(K,R)，其中K是（　　）的有限集合，R是K上关系的有限集合。

   A.算法　　B.数据元素　　C.数据操作　　D.逻辑结构

2. 数据元素是数据的最小单位。　　（　　）

3. 存储数据时，通常不仅需要存储各数据元素的值，而且还要存储（　　）。

   A.数据的处理方法　　B.数据元素的类型

   C.数据元素之间的关系　　D.数据的存储方法

4. 逻辑上可以将数据结构分为（　　）。

   A.动态结构和静态结构　　B.线性结构和非线性结构

   C.顺序结构和链式结构　　D.初等结构和组合结构

5. 按数据元素的逻辑关系来说，数据结构可分为四种：线性表、集合、树和图，其中树形结构中的数据元素之间存在“_____”的关系。

6. 有向图是一种非线性结构。　　（　　）

7. 以下属于逻辑结构的是（　　）。

   A.顺序表　　B.哈希表　　C.有序表　　D.单链表

8. 以下是4个算法的时间复杂度函数表达式，其中时间复杂度最小的是（　　）。

   A. $T(n)=2n^3+3n^2+1000$　　B. $T(n)=n^3-2000$

   C. $T(n)=n^2\log_2 n+n^2$　　D. $T(n)=n^2+1000$

9. 算法是对特定问题求解步骤的一种描述，它具有_____、_____、可行性、输入和输出五个重要的特性。

10. 求下列程序段的时间复杂度。

```
(1) for(i=0;i<=m;i++)
        for(j=0;j<n;j++)
            A[i][j]=0;
```

(2) y=0;
```
while((y+1)*(y+1)<=n)
    y=y+1;
```
(3) i=1;
```
while(i<=n)
    i=i*3;
```

课时二　顺序表

考点	重要程度	占分	题型
1. 线性表的定义	★★★	0～3	选择、填空
2. 顺序表的结构	★★★★★	4～6	
3. 顺序表的实现	必考	6～8	算法

1. 线性表的类型定义

线性表是具有相同数据类型的$n(n \geq 0)$个数据元素的有限序列。

用L命名线性表，则其一般表示为

$$L=(a_1, a_2, \cdots, a_i, a_{i+1}, \cdots, a_n)$$

除第一个元素外，每个元素有且仅有一个直接前驱。

除最后一个元素外，每个元素有且仅有一个直接后继。

题 1. 线性表是具有n个（　　）的有限序列。

A. 表元素　　*B*. 字符　　*C*. 数据元素　　*D*. 数据项

答案：*C*

线性表的基本操作：

（1）*InitList*(&*L*)：初始化表。构造一个空的线性表。

（2）*Length*(*L*)：求表长。返回线性表*L*的长度，即*L*中数据元素的个数。

（3）*LocateElem*(*L*,*e*)：按值查找操作。获取表*L*中查找具有给定关键字值的元素。

（4）*GetElem*(*L*,*i*)：按位查找操作。获取表*L*中第*i*个位置的元素的值。

（5）*ListInsert*(&*L*,*i*,*e*)：插入操作。在表*L*中的第*i*个位置上插入指定元素*e*。

（6）*ListDelete*(&*L*,*i*,&*e*)：删除操作。删除表*L*中第*i*个位置的元素，并用*e*返回删除元素的值。

（7）*PrintList*(*L*)：输出操作。按前后顺序输出线性表*L*的元素值。

（8）*Empty*(*L*)：判空操作。

（9）*DestroyList*(&*L*)：销毁操作。

2. 顺序表的结构

线性表的顺序存储称为顺序表，它是由一组地址连续的存储单元依次存储线性表中的数据元素，从而使得逻辑上相邻的两个元素在物理位置上也相邻。

数组下标	顺序表	内存地址
0	a_1	LOC(A)
1	a_2	LOC(A)+sizeof(ElemType)
	⋮	
i-1	a_i	LOC(A)+sizeof(ElemType)*(i-1)

题 1. 线性表采用顺序存储结构表示时，必须占用一片连续的存储单元。　（　）

答案：正确

题 2. 如果一个顺序表中第一个元素的存储地址为1000，每个元素占4个地址单元，那么第6个元素的存储地址应是（　）。

A.1020　　*B*.1010　　*C*.1016　　*D*.1024

答案：*A*　　$1000+(6-1)*4=1020$

假定线性表的元素类型为 ElemType，则线性表的顺序存储类型描述为

```
#define MaxSize 50                    //定义线性表的最大长度
typedef struct{
   ElemType data[MaxSize];            //顺序表的元素
   int length;                        //顺序表的当前长度
}SqList;                              //顺序表的类型定义
```

上述一维数组是属于静态分配，由于数组的大小和空间事先已经固定，一旦空间占满，再加入新的数据将会产生溢出，进而导致程序崩溃。

而在动态分配时，存储数组的空间是在程序执行过程中通过动态存储分配语句分配的，一旦数据空间占满，就另外开辟一块更大的存储空间，用以替换原来的存储空间，从而达到扩充存储数据空间的目的，而不需要为线性表一次性地划分所有空间。

```
#define InitSize 50                    //表长度的初始定义
typedef struct{
   ElemType *data;                     //指示动态分配数组的指针
   int MaxSize,length;                 //数组的最大容量和当前个数
}SeqList;                              //动态分配数组顺序表的类型定义
```

C 语言的初始动态分配语句为

```
L.data=(ElemType*)malloc(sizeof(ElemType)*InitSize);
```

顺序表的特点：

顺序表最主要的特点是随机存取，即通过首地址和元素序号可在时间 $O(1)$ 内找到指定的元素。

顺序表的存储密度高，每个结点只存储数据元素。

顺序表逻辑上相邻的元素物理上也相邻，所以插入和删除操作需要移动大量元素。

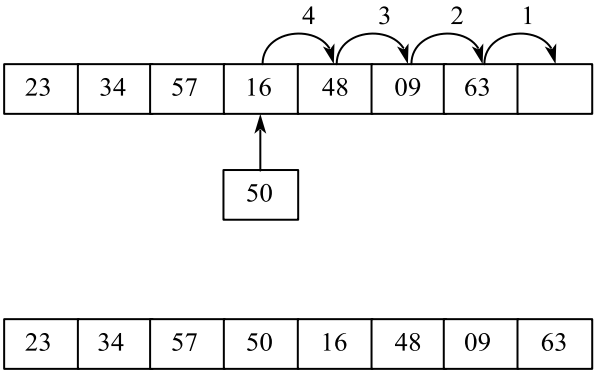
题 3. 当需要随机查找线性表的元素时，宜采用（　　）作存储结构。

A. 双向链表　　*B*. 循环链表　　*C*. 顺序表　　*D*. 单链表

答案：*C*

3. 顺序表的实现

(1)插入操作



在顺序表 L 的第 $i(1 \leqslant i \leqslant L.length+1)$ 个位置插入新元素 e。若 i 的输入不合法，则返回 $false$，表示插入失败；否则，将顺序表的第 i 个元素及其后的所有元素右移一个位置，腾出一个空位置插入新元素 e，顺序表长度增加1，插入成功，返回 $true$。

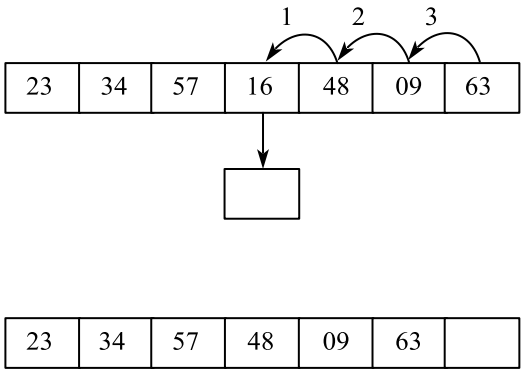
```
bool ListInsert(SqList &L,int i,ElemType e){
    if(i<1||i>L.length+1)                    //判断i的范围是否有效
        return false;
    for(int j=L.length;j>=i;j--)             //将第i个元素及之后的元素后移
        L.data[j]=L.data[j-1];
    L.data[i-1]=e;                           //在位置i处放入e
    L.length++;                              //线性表长度增加1
    return true;
}
```

题 1. 在长度为 n 的顺序表的第 i 个位置上插入一个元素 $(1 \leqslant i \leqslant n+1)$，元素的移动次数为（　　）。

$A. n-i+1$　　$B. n-i$　　$C. i$　　$D. i-1$

答案：A

(2)删除操作



删除线性表L中第$i(1\leqslant i\leqslant L.length)$个位置的元素，若成功则返回$true$，并将被删除的元素用引用变量$e$返回，否则返回$false$。

```
bool ListDelete(SqList &L,int i,ElemType &e){
    if(i<1||i>L.length)                    //判断i的范围是否有效
        return false;
    e=L.data[i-1];                         //将被删除的元素赋给e
    for(int j=i;j<L.length;j++)            //将第i个位置后的元素后移
        L.data[j-1]=L.data[j];
    L.length--;                            //线性表长度减1
    return true;
}
```

题 2. 设顺序线性表中有n个数据元素，则删除表中第i个元素需要移动（　　）个元素。

$A.\ n-i$　　$B.\ n-i-1$　　$C.\ n-i+1$　　$D.\ i$

答案：A

题 3. 对于顺序存储的线性表，访问结点和增加、删除结点的时间复杂度为（　　）。

$A.\ O(n),O(n)$　　$B.\ O(n),O(1)$　　$C.\ O(1),O(n)$　　$D.\ O(1),O(1)$

答案：C

(3)按值查找操作

在顺序表L中查找第一个元素值等于e的元素，并返回其位序。

```
int LocateElem(SqList L,ElemType e){
    int i;
    for(i=0;i<L.length;i++)
        if(L.data[i]==e)
            return i+1;                    //下标为i的元素值等于e，返回其位序
    return 0;                              //退出循环，说明查找失败
}
```

(4)按位查找操作

```
int GetElem(SqList L,int i){
    if(i<1||i>L.length)                    //判断i的范围是否有效
      return 0;
    return L.data[i-1];
}
```

题 4.将两个有序顺序表A, B合并为一个新的有序顺序表C，并用函数返回结果顺序表。

算法思想:首先,按顺序不断取下两个顺序表表头较小的结点存入新的顺序表中。然后，看哪个表还有剩余，将剩下的部分添加到新的顺序表后面。

```
bool Merge(SqList A,SqList B,SqList &C){
    if(A.length+B.length>C.MaxSize)
        return false;
    int i=0,j=0,k=0;
    while(i<A.length&&j<B.length){
        if(A.data[i]<=B.data[j])
            C.data[k++]=A.data[i++];
        else
            C.data[k++]=B.data[j++];
    }
    while(i<A.length)
```

```
            C.data[k++]=A.data[i++];
        while(j<B.length)
            C.data[k++]=B.data[j++];
        C.length=k;
        return true;
    }
```

课时二　练习题

1. 线性表的特点是每个元素都有一个前驱和一个后继。　（　）

2. 线性表的顺序存储结构是一种（　）的存储结构。

A. 随机存取　　*B*. 顺序存取　　*C*. 索引存取　　*D*. 散列存取

3. 下述（　）是顺序存储结构的优点。

A. 存储密度大　　*B*. 插入运算方便

C. 删除运算方便　　*D*. 无需大片连续存储空间

4. 两个有序顺序表分别是具有n个元素与m个元素且$n \leqslant m$，现将其归并成一个有序表，其最少的比较次数是（　）。

A. n　　*B*. m　　*C*. $n-1$　　*D*. $m+n$

5. 删除顺序表中第1个数据元素a_0的时间复杂度是$O(n)$。　（　）

6. 在n个结点的顺序表中，算法的时间复杂度是$O(1)$的操作是（　）。

A. 访问第i个结点$(1 \leqslant i \leqslant n)$和求第$i$个结点的直接前驱$(2 \leqslant i \leqslant n)$

B. 在第i个结点后插入一个新结点$(1 \leqslant i \leqslant n)$

C. 删除第i个结点$(1 \leqslant i \leqslant n)$

D. 将n个结点从小到大排序

7. 下面（）不是线性表的特性。

A. 除第一个元素外，每一个元素都有前驱

B. 除最后一个元素外，每一个元素都有后继

C. 线性表是数据元素的有限序列

D. 线性表的长度等于n，并且n不等于0

8. 二维数组$A[10][20]$按行优先顺序存储，每个元素占4个存储单元，$A[1][1]$的存储地址是1000，$A[5][6]$的存储地址是 _____。

9. 已知长度为n的线性表采用顺序存储结构。写一算法，删除线性表中所有值为x的元素。

10. 设计一个高效算法，将顺序表L的所有元素逆置。

课时三　链表

考点	重要程度	占分	题型
1. 单链表的结构	必考	6～10	选择、算法
2. 单链表的实现	必考	4～10	
3. 双链表	★★	0～2	选择、填空、判断
4. 循环链表	★★★★	2～4	

1. 单链表的结构

线性表的链式存储，又称单链表，它是指通过一组任意的存储单元来存储线性表中的数据元素。

为了建立数据元素之间的线性关系，对每个链表结点，除存放元素自身的信息外，还需要存放一个指向其后继的指针。

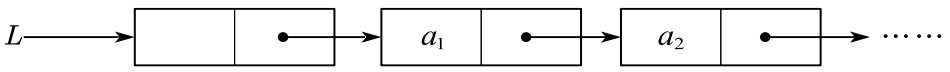
data	*next*

其中 *data* 为数据域，存放数据元素；*next* 为指针域，存放其后继结点的地址。

单链表结点类型的描述如下：

```
typedef struct LNode{                    //定义单链表结点类型
    ElemType data;                       //数据域
    struct LNode *next;                  //指针域
}LNode,*LinkList;
```

通常用头指针来标识一个单链表，如单链表 L，头指针为 *NULL* 时表示一个空表。此外，为了操作上的方便，在单链表第一个结点之前附加一个结点，称为头结点。头结点的数据域可以不设任何信息。头结点的指针域指向线性表的第一个元素结点。



头结点和头指针的区分：不管带不带头结点，头指针始终指向链表的第一个结点，而头结点是带头结点的链表中第一个结点，结点内通常不存储信息。

题 1. 若线性表采用链式存储，则表中各元素的存储地址（　　）。

A. 必须是连续的　　B. 部分地址是连续的

C. 一定是不连续的　　D. 不一定是连续的

答案：D

题 2. 单链表中，增加一个头结点的目的是为了（　　）。

A. 使单链表至少有一个结点　　B. 标识表结点中首结点的位置

C. 方便运算的实现　　D. 说明单链表是线性表的链式存储

答案：C

引入头结点后，可以带来两个优点：

①由于第一个数据结点的位置被存放在头结点的指针域中，所以在链表的第一个位置上的操作和在表的其他位置上的操作一致，无需进行特殊处理。

②无论链表是否为空，其头指针都指向头结点的非空指针（空表中头结点的指针域为空）。

2. 单链表的实现

（1） 按序号查找结点值

在单链表中从第一个结点出发，顺指针 *next* 域逐个往下搜索，直到找到第 *i* 个结点为止，否则返回最后一个指针域 *NULL*。

按序号查找结点值的算法如下：

```
LNode *GetElem(LinkList L,int i){
    int j=1;                    //计数，初始为1
    LNode *p=L->next;           //头结点指针赋给p
    if(i==0)
        return L;               //若i等于0，则返回头结点
    if(i<0)
        return NULL;            //若i无效，则返回NULL
    while(p&&j<i){              //从第1个结点开始找，查找第i个结点
        p=p->next;
```

```
            j++;
        }
        return p;          //返回第i个结点的指针，若i大于表长则返回NULL
    }
```

按序号查找操作的时间复杂度为$O(n)$。

（2） 按值查找表结点

从单链表的第一个结点开始，由前往后一次比较表中各结点数据域的值，若某结点数据域的值等于给定值e，则返回该结点的指针，若整个单链表中没有这样的结点，则返回$NULL$。

按值查找表结点的算法如下：

```
LNode *LocateElem(LinkList L,ElemType e){
    LNode *p=L->next;
    while(p!=NULL&&p->data!=e) //从第一个结点开始查找data域为e的结点
        p=p->next;
    return p;                  //找到后返回该结点指针，否则返回NULL
}
```

按值查找操作的时间复杂度为$O(n)$。

题 1.从一个具有n个结点的单链表中查找其值等于x的结点时，在查找成功的情况下，平均需比较（　　）个元素结点。

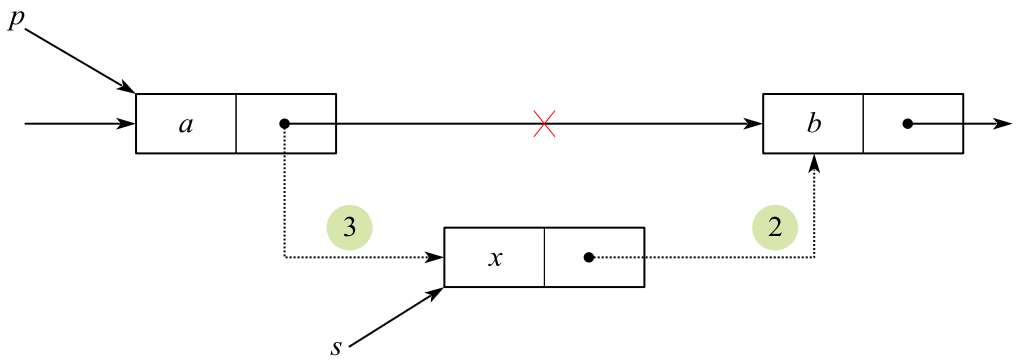
$A.\ n/2$　　$B.\ n$　　$C.\ (n+1)/2$　　$D.\ (n-1)/2$

答案：C

（3） 插入结点操作

插入结点操作将值为x的新结点插入到单链表的第i个位置上。先检查插入位置的合法性，然后找到待插入位置的前驱结点，即第$i-1$个结点，再在其后插入新结点。

算法首先调用按序号查找算法$GetElem(L,i-1)$，查找$i-1$个结点。假设返回的第$i-1$个结点为$*p$，然后令新结点$*s$的指针域指向$*p$的后继结点，再令结点$*p$的指针域指向新插入的结点$*s$。



实现插入结点的代码如下：

① $p=GetElem(L,i-1);$　　//查找插入位置的前驱结点

② $s->next=p->next;$

③ $p->next=s;$

注意：②③语句的顺序不能颠倒。

前插操作是指在某结点的前面插入一个新结点。

我们仍然$*s$插入到$*p$的后面，然后将$p->data$和$s->data$交换，这样实现了将新结点$*s$插入到了结点$*p$的前面。

算法的代码片段如下：

$s->next=p->next;$

$p->next=s;$

$temp=p->data;$

$p->data=s->data;$

$s->data=temp;$

题 2. 在单链表中，已知p,q,s是指向结点的指针，且q是p的直接前驱，若在q和p之间插入s，则需执行（　　）。

A. $s->next=p->next;\ p->next=s;$

B. $q->next=s;\ s->next=p;$

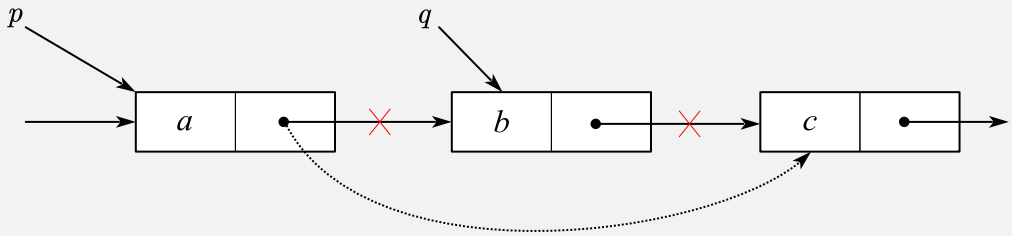
C. $p->next=s->next;\ s->next=p;$

D. $p->next=s;\ s->next=q;$

答案：B

（4） 删除结点操作

删除结点操作是将单链表的第i个结点删除。先检查删除位置的合法性，后查找表中第$i-1$个结点，即被删除结点的前驱结点，再将其删除。



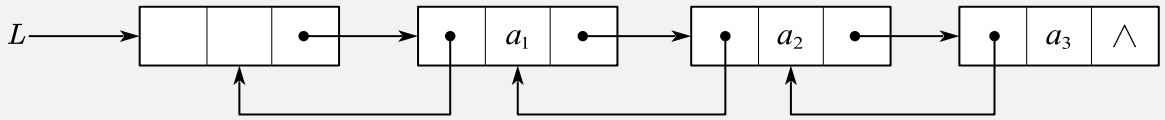
实现删除结点的代码片段如下：

```
p=GetElem(L,i-1);          //查找删除位置的前驱结点
q=p->next;                 //令q指向被删除的结点
p->next=q->next;           //将*q结点从链中断开
free(q);                   //释放结点的内存空间
```

3. 双链表、循环链表

（1） 双链表

双链表结点中有两个指针$prior$和$next$，分别指向其前驱结点和后继结点。



双链表中结点类型描述如下：

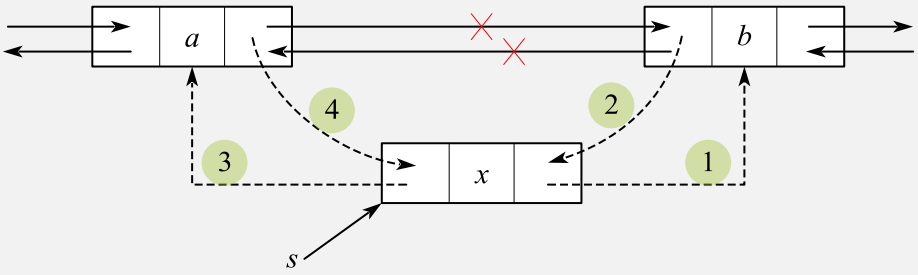
```
typedef struct DNode{              //定义双链表结点类型
        ElemType data;             //数据域
        struct DNode *prior,*next; //前驱和后继指针
}DNode,*DLinkList;
```

题 1. 在双向链表中，每个结点含有两个指针域，一个指向_____结点，另一个指向_____结点。

答案：前驱，后继

① 双链表的插入操作

在双链表中 p 所指的结点之后插入结点 $*s$。



插入操作的代码片段如下：

① $s->next=p->next;$

② $p->next->prior=s;$

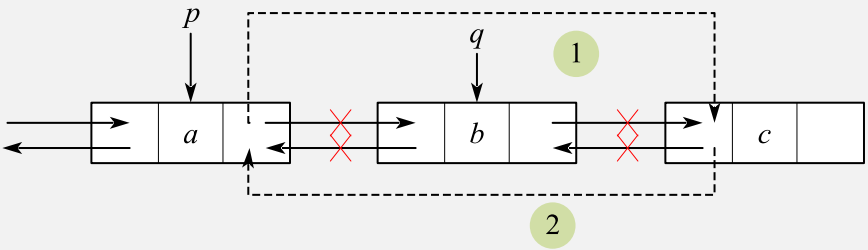
③ $s->prior=p;$

④ $p->next=s;$

其中，①②语句必须在④语句之前。

② 双链表的删除操作

删除双链表中结点 $*p$ 的后继结点 $*q$。



删除操作的代码片段如下：

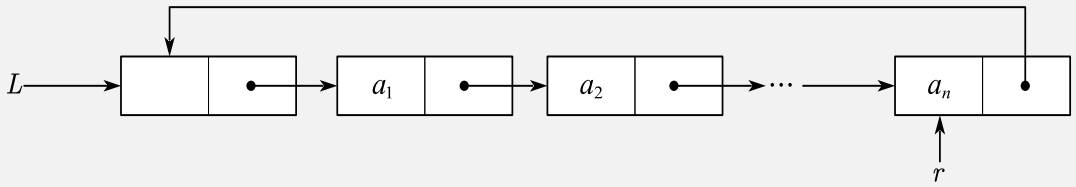
$p->next=q->next;$

$q->next->prior=p;$

$free(q);$

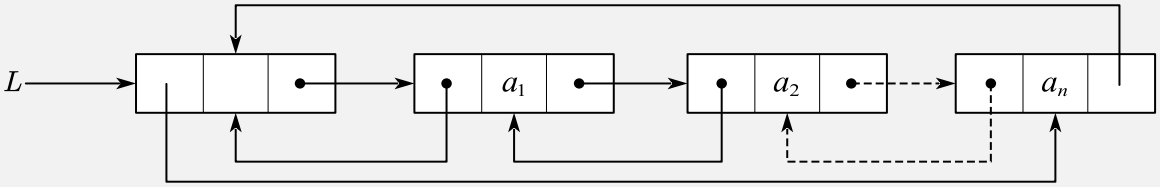
（2） 循环链表

① 循环单链表



在循环单链表中，表为结点*r 的next 域指向L，故表中没有指针域为NULL 的结点，因此，循环单链表的判空条件不是头结点的指针是否为空，而是它是否等于头指针。

② 循环双链表



在循环双链表L 中，某结点*p 为尾结点时，$p->next=L$；

当循环双链表为空表时，其头结点的 prior 域和next 域都等于L。

题 2. 循环链表中每一个元素都有后继。　　（　　）

答案：正确

题 3. 有一个含头结点的双向循环链表，头指针为 head，则其为空的条件是（　　）。

A. $head->prior==NULL$

B. $head->next==NULL$

C. $head->next==head$

D. $head->next->prior==NULL$

答案：C

题 4. 在以下的叙述中，正确的是（　　）。

A. 线性表的顺序存储结构优于链式存储结构

B. 线性表的顺序存储结构适用于频繁插入或删除数据元素的情况

C. 线性表的链式存储结构适用于频繁插入或删除数据元素的情况

D. 线性表的链式存储结构优于顺序存储结构

答案：C

课时三　练习题

1. 线性表采用链式存储时，其地址（　　）。

   A. 必须是连续的　　B. 一定是不连续的

   C. 部分地址必须是连续的　　D. 连续与否都可以

2. 顺序存储方式插入和删除数据元素效率太低，因此它不如链式存储方式好。（　　）

3. 在双向链表中，若要求在p指针所指的结点之前插入指针为s所指的结点，则需执行下列语句：$s->next=p;\ s->prior=$_____;\ _____$=s;\ p->prior=s;$

4. 链表不具有的特点是（　　）。

   A. 插入不需要移动元素　　B. 可随机访问任一元素

   C. 不必事先估计存储空间　　D. 删除不需要移动元素

5. 在一个单链表中，已知q结点是p结点的前驱结点，若在q和p之间插入s结点，则执行（　　）。

   A. $s->next=p->next;\ p->next=s;$

   B. $p->next=s->next;\ s->next=p;$

   C. $q->next=s;\ s->next=p;$

   D. $p->next=s;\ s->next=q;$

6. 线性表的每个结点只能是一个简单类型，而链表的每个结点可以是一个复杂类型。（　　）

7. 将长度为n的单链表A链接在长度为m的单链表B之后的算法时间复杂度为_____。

8. 对于在表的首、尾两端进行插入操作的线性表，宜采用的存储结构是（　　）。

   A. 顺序表　　B. 用头指针表示的单循环链表

   C. 用尾指针表示的单循环链表　　D. 单链表

9. 链表的删除算法很简单，因为当删除链中某个结点后，计算机会自动地将后续的各个单元向前移动。　　（　　）

10. 阅读下列算法，并补充所缺语句。

从头指针为*la*的带表头结点的有序顺序表中删除所有值相同的多余元素，并释放被删除结点的空间。

```
void purge_linkst(ListLink &la){
    ListNode *p,*q,*t;
    ElemType temp;
    p=la->link;
    while(p!=NULL){
        q=p;
        temp=p->data;
        p=p->link;
        if(p!=NULL&&__________)
              p=p->link;
        else{
             while(p!=NULL&&__________){
                  t=p;
                  p=p->link;
                  free(t);
             }
             p=q->link;
         }
     }
 }
```

课时四　栈和队列

考点	重要程度	占分	题型
1. 栈的基本概念	★★★★	2～4	选择、填空、判断
2. 栈的存储结构	★★★★★	4～8	
3. 队列的基本概念	★★★★	2～4	
4. 队列的存储结构	★★★★★	4～8	

1. 栈的基本概念

栈(*Stack*)是只允许在一端进行插入或删除操作的线性表。

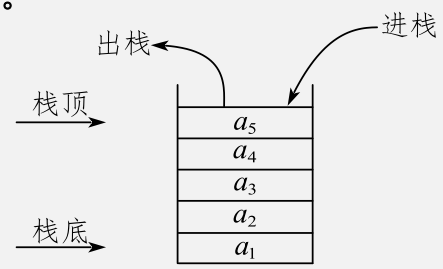
限定这种线性表只能在某一端进行插入和删除操作。

栈顶：线性表允许进行插入删除的那一端。

栈底：不允许进行插入和删除的另一端。

空栈：不含任何元素的空表。

栈操作的特性是先进后出(*First In LAST Out,FILO*)。



题 1. 限定仅在表尾进行插入和删除操作的线性表称为_____，它的修改是按_____的原则进行的。

答案：栈，先进后出

题 2. 若进栈序列为a,b,c，则通过入栈操作可能得到的a,b,c出栈的不同排列个数（　　）。

A. 4　　*B*. 5　　*C*. 6　　*D*. 7

答案：*B*

n不同元素进栈，出栈元素不同排列的个数为$\frac{1}{n+1}C_{2n}^{n}$。

栈的基本操作：

InitStack(&*S*)：初始化一个空栈S。

StackEmpty(*S*)：判断一个栈是否为空，若栈S为空则返回*true*，否则返回*false*。

Push(&*S*,*x*)：进栈，若栈 S 未满，则将x加入使之成为新栈顶。

Pop(&*S*,&*x*)：出栈，若栈S非空，则弹出栈顶元素，并用x返回。

GetTop(*S*,&*x*)：读栈顶元素，若栈S非空，则用x返回栈顶元素。

DestroyStack(&*S*)：销毁栈，并释放栈S占用的存储空间（"&"表示引用调用）。

2. 栈的存储结构

栈是一种操作受限的线性表，类似于线性表，它也有对应的两种存储方式。

（1） 顺序栈的实现

采用顺序存储的栈称为顺序栈，它利用一组地址连续的存储单元存放自栈底到栈顶的数据元素，同时附设一个指针（*top*）指向当前栈顶元素的位置。

栈的顺序存储类型可描述为

```
#define MaxSize 50              //定义栈中元素的最大个数
typedef struct{
   ElemType data[MaxSize];      //存放栈中元素
   int top;                     //栈顶指针
}SqStack;
```

栈顶指针：$S.top$，初始时设置$S.top=-1$;栈顶元素：$S.data[S.top]$。

进栈操作：栈不满时，栈顶指针先加1，再送值到栈顶元素。

出栈操作：栈非空时，先取栈顶元素值，再将栈顶指针减1。

栈空条件：$S.top==-1$;栈满条件：$S.top==MaxSize-1$；栈长：$S.top+1$。

（2） 顺序栈的基本运算

①初始化

```
void InitStack(SqStack &S){
    S.top=-1;                       //初始化栈顶指针
}
```

②判栈空

```
bool StackEmpty(SqStack S){
   if(S.top==-1)                    //栈空
      return true;
   else                             //不空
      return  false;
}
```

③进栈

```
bool Push(SqStack &S,ElemType x){
    if(S.top==MaxSize-1)              //栈满，报错
        return false;
    S.data[++S.top]=x;                //指针先加1，再入栈
    return true;
}
```

题 1. 向顺序栈中压入新元素，应当（　　）。

A. 先移动栈顶指针，再存入元素　　*B*. 先存入元素，再移动栈顶指针

C. 先后次序无关紧要　　*D*. 同时进行

答案：*A*

④出栈

```
bool Pop(SqStack &S,ElemType &x){
    if(S.top==-1)                     //栈空，报错
        return false;
    x=S.data[S.top--];                //先出栈，指针再减1
    return true;
}
```

⑤读栈顶元素

```
bool GetTop(SqStack S,ElemType &x){
    if(S.top==-1)                     //栈空，报错
        return false;
    x=S.data[S.top];                  //x 记录栈顶元素
    return  true;
}
```

题 2. 一个栈的入栈序列是1,2,3,4,5，则栈的不可能输出序列是（　　）。

A. 43512　　*B*. 54321　　*C*. 45321　　*D*. 12345

答案：*A*

题 3. 对于声明：ListStack myStack;

myStack.push("*A*");

myStack.push("*K*");

myStack.push("*D*");

myStack.pop( );

myStack.push("*H*");

myStack.push("*K*");

myStack.pop( );

myStack.pop( );

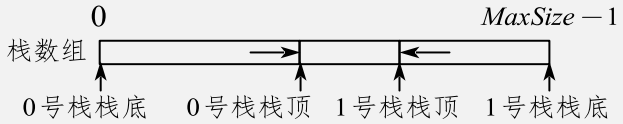
myStack.push("*D*");

此时，栈内元素从栈底向栈顶依次为_____。

答案：*AKD*

（3） 共享栈

利用栈底位置相对不变的特性，可让两个顺序栈共享一个一维数组空间，将两个栈的栈底分别设置在共享空间的两端，两个栈顶向共享空间的中间延伸。



两个栈的栈顶指针都指向栈顶元素，$top0=-1$时0号栈为空，$top1=MaxSize$时1号栈为空；仅当两个栈顶指针相邻($top1-top0=1$)时，判断为栈满. 当0号栈进栈时$top0$先加1再赋值，1号栈进栈时$top1$先减1再赋值；出栈时则刚好相反。

共享栈是为了更有效的利用存储空间，两个栈的空间相互调节，只有在整个存储空间被占满时才发生上溢。

题 1. 若栈采用顺序存储方式存储，现两栈共享空间$V[1\cdots m]$,$top1$,$top2$分别代表第一和第二个栈的栈顶，栈1的底在$V[1]$，栈2的底在$V[m]$，则栈满的条件是（　　）。

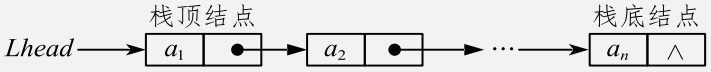
A. $|top2-top1|=0$　　*B*. $top1+1=top2$

C. $top1+top2=m$　　*D*. $top1=top2$

答案：*B*

（4） 栈的链式结构

采用链式存储的栈称为链栈，链栈的优点是便于多个栈共享存储空间和提高其效率，且不存在栈满上溢的情况。通常采用单链表实现，并规定所有操作都是在单链表的表头进行的. 这里规定链栈没有头结点，*Lhead* 指向栈顶元素。



栈的链式存储类型可描述为

```
typedef struct LinkNode{
    ElemType data;                  //数据域
    struct LinkNode *next;          //指针域
}*LiStack;                          //栈类型定义
```

题 1. 链栈对比顺序栈主要优点在于（　　）。

A. 通常不会出现栈满的情况　　*B*. 通常不会出现栈空的情况

C. 插入操作更加方便　　*D*. 删除操作更加方便

答案：*A*

题 2. 将一个递归算法改为对应的非递归算法时，通常需要使用（　　）.

A. 栈　　*B*. 队列　　*C*. 循环队列　　*D*. 优先队列

答案：*A*　　栈能适用于递归算法，表达式求值以及括号匹配等问题中。

3. 队列的基本概念

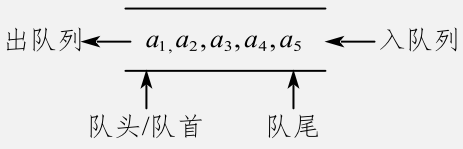
队列(*Queue*)简称队，也是一种操作受限的线性表，只允许在表的一端进行插入，而在表的另一端进行删除。

向队列中插入元素称为入队或进队；删除元素称为出队或离队。

队列操作的特性是先进先出(*First In First Out,FIFO*)。

队头：允许删除的一端，又称队首。

队尾：允许插入的一端。



题 1. 队列是一种操作受限的线性表，它与栈不同的是，队列中所有的插入操作均限制在表的一段进行，而所有的删除操作都限制在表的另一端进行，允许插入的一端称为_____，允许删除的一端称为_____，队列具有_____的特点。

答案：队尾，队首，先进先出

题 2. 一个队列的入队序列是 a,b,c,d，则队列的输出序列是（　　）。

A. a,b,c,d　　*B.* a,d,c,b　　*C.* c,b,d,a　　*D.* d,c,b,a

答案：*A*

队列的基本操作：

InitQueue(&*Q*)：初始化队列，构造一个空队列 Q。

QueueEmpty(Q)：判队列空，若队列 Q 为空返回 *true*，否则返回 *false*。

EnQueue(&Q,x)：入队，若队列 Q 未满，将 x 加入，使之成为新的队尾。

DeQueue(&Q,&x)：出队，若队列 Q 非空，删除队头元素，并用 x 返回。

GetHead(Q,&x)：读队头元素，若队列 Q 非空，则将队头元素赋值给 x。

需要注意的是，栈和队列是操作受限的线性表，因此不是任何对线性表的操作都可以作为栈和队列的操作。比如，不可以随便读取栈或队列中间的某个数据。

4. 队列的存储结构

（1） 队列的顺序存储

队列的顺序实现是指分配一块连续的存储单元存放队列中的元素，并附设两个指针；队头指针 *front* 指向队头元素，队尾指针 *rear* 指向队尾元素的下一个位置。

队列的顺序存储类型可描述为

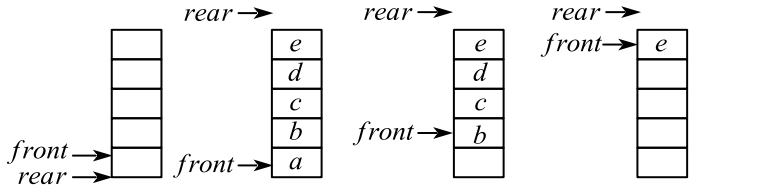
```
#define MaxSize 50                  //定义队列中元素的最大个数
typedef struct{
   ElemType data[MaxSize];          //存放队列元素
   int front,rear;                  //队头指针和队尾指针
 } SqQueue;
```

初始状态（队空条件）：$Q.front == Q.rear == 0$

进队操作：队不满时，先送值到队尾元素，再将队尾指针加1

出队操作：队不空时，先取队头元素值，再将队头指针加1

不能用$Q.rear == MaxSize$作为队列满的条件。图(d)中，队列中仅有一个元素，但仍满足该条件。这时入队出现“上溢出”，但这种溢出并不是真正的溢出，在$data$数组中依然存在可以存放元素的空位置，所以是一种假溢出。



(a)空队　(b)5个元素入队　(c)出队1次　(d)出队3次

（2） 循环队列

把存储队列元素的表从逻辑上视为一个环，称为循环队列。

当队首指针$Q.front = MaxSize - 1$后再前进一个位置就自动到0，这可以利用除法取余运算(%)来实现。

初始时：$Q.front = Q.rear = 0$.

队首指针进1：$Q.front = (Q.front + 1)\%MaxSize$

队尾指针进1：$Q.rear = (Q.rear + 1)\%MaxSize$

队列长度：$(Q.rear + MaxSize - Q.front)\%MaxSize$

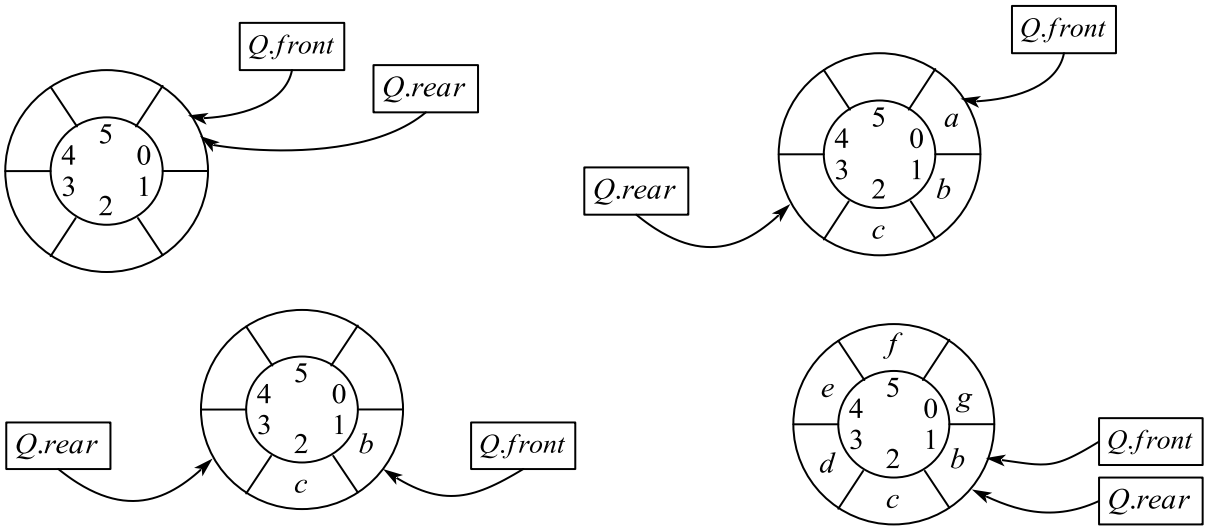
出队入队时：指针都按顺时针方向进1。

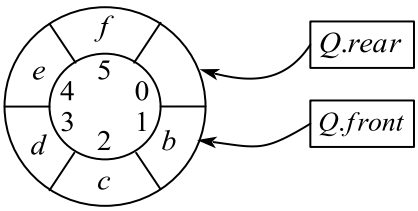
牺牲一个单元来区分队空和队满，入队时少用一个队列单元。

队满条件：$(Q.rear + 1)\%MaxSize == Q.front$

队空条件：$Q.front == Q.rear$

队列中元素的个数：$(Q.rear - Q.front + MaxSize)\%MaxSize$





题 1. 循环队列用数组 $A[0...m-1]$ 存放其元素值，已知其头尾指针分别是 $front$ 和 $rear$，则当前队列的元素个数是_____。

答案：$(rear+m-front)\%m$

题 2. 循环队列 Q 的数据元素值存放在长度为 m 的数组中，且此数组最多只能存放 $m-1$ 个数据元素. 已知头尾指针分别是 $Q.front$ 和 $Q.rear$，则判断 Q 为满队列的条件是（　　）。

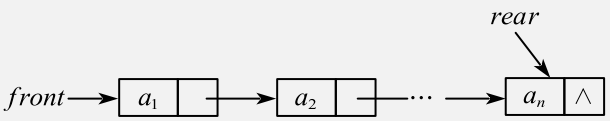
A. $Q.front==Q.rear$　　B. $Q.front==(Q.rear+1)\%m$

C. $Q.front!=Q.rear$　　D. $Q.front!=(Q.rear+1)\%m$

答案：B

（3） 队列的链式存储结构

队列的链式表示称为链队列，它实际上是一个同时带有队头指针和队尾指针的单链表。头指针指向队头结点，尾指针指向队尾结点，即单链表的最后一个结点。



不带头结点的链式队列

题 1. 在一个链队列中，假定 $front$ 和 $rear$ 分别为队首指针和队尾指针，则删除一个结点的操作为（　　）。

A. $front=front->next$　　B. $rear=rear->next$

C. $rear=front->next$　　D. $front=rear->next$

答案：A

课时四　练习题

1. 栈的*push*和*pop*操作均在（　）进行。

A.栈顶位置　　*B*.栈底位置　　*C*.任意位置　　*D*.中间位置

2. 栈又称先进先出的线性表。　（　）

3. 若已知一个栈的入栈序列是$1,2,3,...,n$，其输出序列为$P_1,P_2,P_3,...P_n$，若$P_1=n$，则P_i为（　）。

A. i　　*B*. $n-i$　　*C*. $n-i+1$　　*D*.不确定

4. 由两个栈共享一个向量空间的好处是（　）。

A.减少存取时间，降低下溢发生的概率

B.节省存储空间，降低上溢发生的概率

C.减少存取时间，降低上溢发生的概率

D.节省存储空间，降低下溢发生的概率

5. 在循环顺序队列中，假设以设置一个计数变量*num*的方法来区分队列判满和判空的条件，*front*和*rear*分别为队首和队尾指针，它们分别指向队首元素和队尾元素的下一个存储单元。队列的最大存储容量为*MaxSize*，则下面不是队列判满或判空条件是（　）。

A. $front==rear$　　*B*. $front==rear\ \&\&num==0$

C. $front==rear\ \ \&\&num>0$　　*D*. $num==MaxSize$

6. 设栈S和队列Q的初始状态为空，元素$e1,e2,e3,e4,e5,e6$依次通过栈S，一个元素进栈后即进队列Q，若六个元素出栈的序列是$e2,e4,e3,e6,e5,e1$，则栈S的容量至少应该是 _____。

7. 在具有n个单元的循环队列中，队满时共有_____个元素。

8. 栈和队列的共同点是（　）。

A.都是先进先出　　*B*.都是先进后出

C.只允许在端点处插入和删除元素　　*D*.没有共同点

9. 解决括号匹配问题，最适合使用（　）数据结构。

A.堆　　*B*.栈　　*C*.队列　　*D*.二叉树

课时五　串

考点	重要程度	占分	题型
1. 串的基本概念	★★★★	2～4	选择、填空、判断
2. 串的简单模式匹配算法	★★★	0～2	
3. 矩阵的压缩存储	★★★★★	2～4	

1. 串的基本概念

串是由零个或多个字符组成的有限序列。一般记为

$$S='a_1a_2\cdots a_n'$$

其中，S 是串名，单引号括起来的字符序列是串的值；a_i 可以是字母、数字或其他字符；串中字符的个数 n 称为串的长度。$n=0$ 时的串称为空串($\varnothing$)。

串中任意个连续的字符组成的子序列称为该串的子串，包含子串的串相应地称为主串。某个字符在串中的序号称为该字符在串中的位置。子串在主串中的位置以子串的第一个字符在主串中的位置来表示。当两个串的长度相等且每个对应位置的字符都相等时，称这两个串是相等的。

由一个或多个空格组成的串称为空格串，其长度为串中空格字符的个数。

例如，$A='China\ Beijing'$, $B='Beijing'$, $C='China'$，它们的长度分别是13, 7, 5。B,C 都是 A 的子串，B 在 A 中的位置是7，C 在 A 的位置是1。

串的基本操作：

① $StrCopy(\&T,S)$：复制操作。由串 S 复制得到 T。

② $StrEmpty(S)$：判空操作。

③ $StrCompare(T,S)$：比较操作。

④ $StrLength(S)$：求串长。

⑤ $SubString(\&Sub,S,pos,len)$：求子串。用 Sub 返回 S 的第 pos 个字符起长度为 len 的子串。

⑥ $Concat(\&T,S1,S2)$：串联接。用 T 返回由 $S1$ 和 $S2$ 联接而成的新串。

⑦ $Index(S,T)$：定位操作。若主串 S 中存在与串 T 值相同的子串，则返回它在主串 S 中第一次出现的位置；否则返回函数值为0。

题 1. 串中任意个字符组成的子序列称为该串的子串。　　（　　）

答案：错误

题 2. 从数据结构来看，串是一种特殊的线性表，其特殊性体现在（　　）。

A. 可以顺序存储　　B. 数据元素可以是一个字符

C. 可以链式存储　　D. 数据元素可以是多个字符

答案：B

题 3. 字符串 $t='child'$, $s='cake'$，请写出下列函数的结果：$StrLength(t)=$_____；

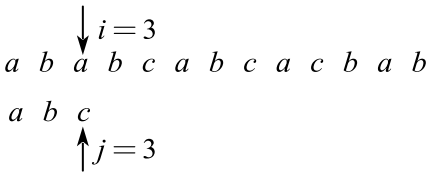
$Concat(\&T,SubString(\&l1,s,3,1),SubString(\&l2,t,2,2))=$_____。

答案：5，'khi'

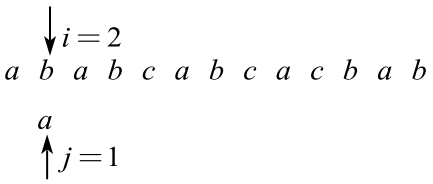
2. 串的简单模式匹配算法

子串的定位操作通常称为串的模式匹配，它求的是子串在主串中的位置。

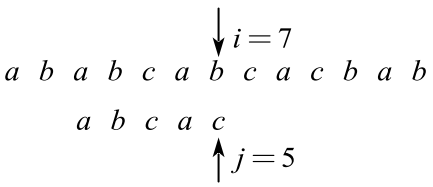
第一趟匹配



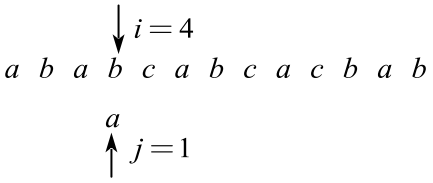
第二趟匹配



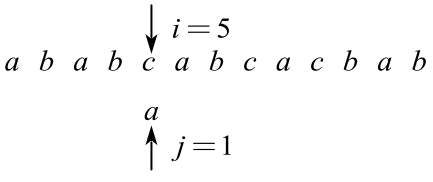
第三趟匹配



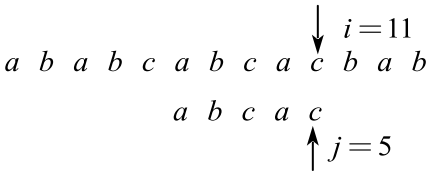
第四趟匹配



第五趟匹配



第六趟匹配



题 1. 设有两个串 p 和 q，求 q 在 p 中首次出现的位置的运算称作（　　）。

A. 连接　　*B*. 模式匹配　　*C*. 求串长　　*D*. 求子串

答案：*B*

3. 矩阵的压缩存储

压缩存储：指为多个值相同的元素只分配一个存储空间，对零元素不分配存储空间。其目的是为了节省存储空间。

特殊矩阵：指具有许多相同矩阵元素或零元素，并且这些相同矩阵元素或零元素的分布有一定规律性的矩阵。常见的特殊矩阵有对称矩阵、上（下）三角矩阵、稀疏矩阵等。

（1） 对称矩阵

若对一个 n 阶方阵 $A[1\cdots n][1\cdots n]$ 中的任意一个元素 $a_{ij}=a_{ji}(1\leqslant i,j\leqslant n)$，则其称为对称矩阵。

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

B

0	1	2				$n(n+1)/2-1$
a_{11}	a_{21}	a_{22}	$\cdots$	$\cdots$	$\cdots$	a_{nn}

将对称矩阵 $A[1\cdots n][1\cdots n]$ 存放在一维数组 $B[n(n+1)/2]$ 中，B 数组下标从 0 开始。

因此，元素 a_{ij} 在数组 B 中的下标 $k=1+2+\cdots+(i-1)+j-1=i(i-1)/2+j-1$。元素下标之间的对应关系如下：

$$k=\begin{cases} \dfrac{i(i-1)}{2}+j-1, & i\geqslant j(\text{下三角区和主对角线元素}) \\ \dfrac{j(j-1)}{2}+i-1, & i<j(\text{上三角区元素}\ a_{ij}=a_{ji}) \end{cases}$$

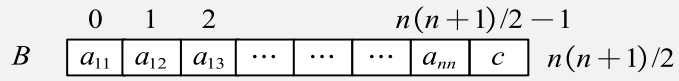
题 1. 设有一个10 阶的对称矩阵 A，采用压缩存储方式，以行序为主存储，a_{11} 为第一个元素，其存储地址为1，每个元素占1个地址空间，存储下三角区元素，则 a_{75} 的地址为_____。

答案：26　　$\dfrac{7\times(7-1)}{2}+5=26$

（2） 三角矩阵

上三角矩阵，下三角区的所有元素均为同一常量。其存储思想与对称矩阵类似，不同之处在于存储完上三角区和主对角线上的元素之后，还要存储对角线下方的元素一次，因此将这个上三角区矩阵 $A[1\cdots n][1\cdots n]$ 压缩存储在 $B[n(n+1)/2+1]$ 中。

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ & a_{22} & \cdots & a_{2n} \\ & & \ddots & \vdots \\ & & & a_{nn} \end{bmatrix}$$

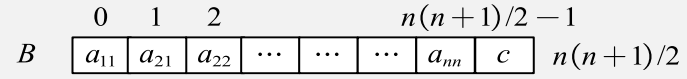


因此，元素 a_{ij} 在数组中的下标 $k=n+(n-1)+\cdots+(n-i+2)+(j-i+1)-1=(i-1)(2n-i+2)/2+(j-i)$。

元素下标的对应关系如下：

$$k=\begin{cases} \dfrac{(i-1)(2n-i+2)}{2}+(j-i), & i\leqslant j(\text{上三角区和主对角线元素}) \\ \dfrac{n(n+1)}{2}, & i>j(\text{下三角区元素}) \end{cases}$$

$$\begin{bmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & \vdots & \ddots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$



因此，元素 a_{ij} 在数组中的下标 $k=1+2+3+\cdots+(i-1)+j-1=i(i-1)/2+j-1$

元素下标的对应关系如下：

$$k=\begin{cases} \dfrac{i(i-1)}{2}+j-1, & i\geqslant j(\text{下三角区和主对角线元素}) \\ \dfrac{n(n+1)}{2}, & i<j(\text{上三角区元素}) \end{cases}$$

题 2. 已知 $n*n$ 的下三角矩阵 A，矩阵元素的行、列下标 $1,2,\cdots,n$，将其下三角区元素存储在一维数组 s 中（从 0 号单元开始存储），则矩阵元素 a_{ii} 在数组 s 中对应元素的下标为_____。

答案：$i(i+1)/2-1$　　$1+2+\cdots+i-1$

（3） 稀疏矩阵

矩阵中非零元素的个数t，相对于矩阵元素的个数s非常少，这样的矩阵称为稀疏矩阵。

将非零元素及其行和列构成一个三元组（行标、列标、值）。稀疏矩阵压缩存储后就失去了随机存取特性。

$$M=\begin{bmatrix}4 & 0 & 0 & 0\\ 0 & 0 & 6 & 0\\ 0 & 9 & 0 & 0\\ 0 & 23 & 0 & 0\end{bmatrix}$$

i	j	v
0	0	4
1	2	6
2	1	9
3	1	23

稀疏矩阵的三元组既可以采用数组存储，也可以采用十字链表法存储。

题 3. 稀疏矩阵一般的压缩存储方法有（　　）。

A. 二维数组和三维数组　　*B*. 三元组顺序表和散列表

C. 三元组顺序表和十字链表　　*D*. 散列表和十字链表

答案：*C*

课时五　练习题

1. 下面关于串的叙述中，（　　）是不正确的。

A. 空串是由空格构成的串　　B. 串是字符的有限序列

C. 模式匹配是串的一种重要运算

D. 串既可以采用顺序存储，也可以采用链式存储

2. 串的长度指的是（　　）。

A. 串中所含不同字母的个数　　B. 串中所含字符的个数

C. 串中所含不同字符的个数　　D. 串中所含非空格字符的个数

3. 两个串相等必有串长度相等且（　　）。

A. 串的各位置字符任意　　B. 串中各位置字符均对应相等

C. 两个串含有不同的字符　　D. 两个所含字符任意

4. 字符串$t='a\ very\ cute\ child', s='I\ like\ coffee\ and\ cake'$，请写出下列函数的结果：

$StrLength(s)=$_____；

$Concat(SubString(\&l1,s,3,15),SubString(\&l2,t,12,6))=$_____。

5. 与线性表相比，串的插入和删除操作的特点是（　　）。

A. 通常以串整体作为操作对象　　B. 需要更多的辅助空间

C. 算法的时间复杂度较高　　D. 涉及移动的元素更多

6. 设有两个串p和q，其中q是p的子串，求q在p中首次出现的位置的算法称为

_____。

7. 若将n阶上三角矩阵A按列优先顺序压缩存放在一维数组B中，第一个非零元素a_{11}存放在$B[0]$中，则应放在$B[k]$中的非零元素a_{ij}的下标i,j与k的对应关系为（　　）。

A. $i(i+1)/2+j$　　B. $i(i-1)/2+j-1$

C. $j(j+1)/2+i$　　D. $j(j-1)/2+i-1$

8. 设有一个10阶的对称矩阵$A[10][10]$，采用压缩存储方式按行将矩阵中下三角部分的元素存入一维数组中，$A[0][0]$存入$B[0]$中，则$A[8][5]$在B中（　）位置。

A. 32　　*B*. 33　　*C*. 41　　*D*. 65

9. 采用带辅助向量的三元组形式实现稀疏矩阵的转置运算，主要是为了（　）。

A. 增加时间复杂度　　*B*. 提高算法效率

C. 降低空间复杂度　　*D*. 节省存储空间

10. 下列（　）是稀疏矩阵的一种压缩存储方法。

A. 顺序表　　*B*. 单链表

C. 双向链表　　*D*. 三元组的顺序表

课时六　树和二叉树（1）

考点	重要程度	占分	题型
1．树的基本概念	★★★★	0～2	选择、填空
2．二叉树	★★★★★	2～4	
3．二叉树的遍历	必考	4～8	解答

1. 树的基本概念

树是$n(n\geq 0)$个结点的有限集。当$n=0$时，称为空树。在任意一颗非空树上应满足：

（1）有且仅有一个特定的根的结点。

（2）当$n>1$时，其余结点可分为$m(m>0)$个互不相交的有限集$T_1,T_2,\ldots T_m$，其中每个集合本身又是一棵树，并且称为根的子树。

树作为一种逻辑结构，同时也是一种分层结构，具有以下两个特点：

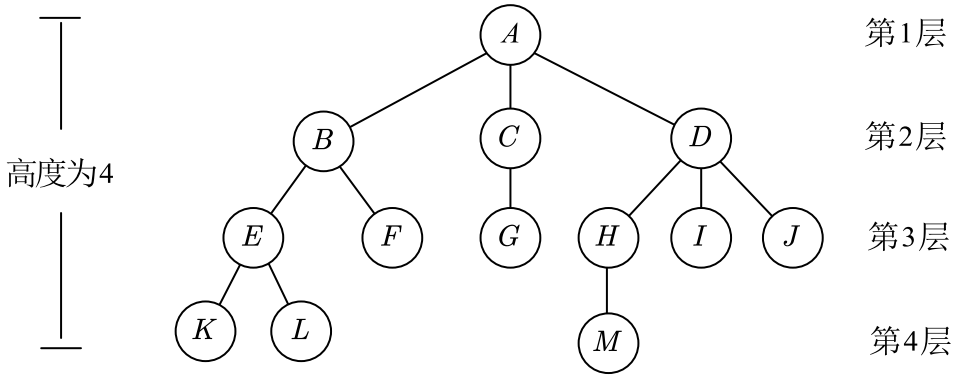
（1）树的根结点没有前驱，除根结点外的所有结点有且只有一个前驱。

（2）树中所有结点可以有零个或多个后继。

题 1. 如果在数据结构中每个数据元素只可能有一个直接前驱，但可有多个直接后继，则该结构是（　　）。

A. 栈　　*B*. 队列　　*C*. 树　　*D*. 图

答案：*C*



基本术语：

（1） 根 A 到结点 K 的唯一路径上的任意结点，称为结点 K 的祖先。

路径上最接近结点 K 的结点 E 称为 K 的双亲，而 K 为结点 E 的孩子。

（2） 树中一个结点的孩子个数称为该结点的度，树中结点的最大度数称为树的度。

（3） 度大于 0 的结点称为分支结点（又称非终端结点）；度为 0（没有孩子结点）的结点称为叶子结点（又称终端结点）。

有相同双亲的结点称为兄弟。

（4） 结点的层次从树根开始定义，根结点为第 1 层，它的子结点为第 2 层。

结点的深度是从根结点开始自顶向下逐层累加。

结点的高度是从叶结点开始自底向上逐层累加。

树的高度（或深度）是树中结点的最大层数。

（5） 有序树和无序树。树中结点的各子树从左到右是有次序的，不能互换，称该树为有序树，否则称为无序树。假设图为有序树，若将子结点位置互换，则变成一颗不同的树。

（6） 路径和路径长度。树中两个结点之间的路径是由这两个结点之间所经过的结点序列构成的，而路径长度是路径上所经过的边的个数。

树的性质：

（1） 树中的结点数等于所有结点的度数加 1。

（2） 度为 m 的树中第 i 层上至多有 m^{i-1} 个结点 $(i \geq 1)$。

（3） 高度为 h 的 m 叉树至多有 $(m^h-1)/(m-1)$ 个结点。

（4） 具有 n 个结点的 m 叉树的最小高度为 $\lceil \log_m(n(m-1)+1) \rceil$。

题 2. 对一棵 n 个结点的树，则该树中结点的度数之和为（　　）。

A. n　　*B*. $n-1$　　*C*. $n+1$　　*D*. 不确定

答案：*B*

题 3. 设一棵度为 3 的树中有 2 个度数为 1 的结点，2 个度数为 2 的结点，2 个度数为 3 的结点，则该树中有（　　）个度数为 0 的结点。

A. 5　　*B*. 6　　*C*. 7　　*D*. 8

答案：*C*

设树中有x个度数为0的结点

$2+2+2+x=2\times1+2\times2+2\times3+1$

2. 二叉树

二叉树的定义：

二叉树的特点是每个结点至多只能有两棵子树（即二叉树中不存在度大于2的结点），并且二叉树的子树有左右之分，其次序不能任意颠倒。二叉树也已递归的形式定义。

二叉树是$n(n\geq0)$个结点的有限集合：

（1） 或者为空二叉树；

（2） 或者由一个根结点和两个互不相交的被称为根的左子树和右子树组成，左右子树又分别是一棵二叉树。

几个特殊的二叉树：

（1）满二叉树。一颗高度为h，且含有2^h-1个结点的二叉树称为满二叉树，即树中的每层都含有最多的结点，如下图（a）所示。满二叉树的叶子结点都集中在二叉树的最下一层，并且除叶子结点之外的每个结点度数均为2。

对满二叉树按层序编号：约定编号从根结点（根结点编号为1）起，自上而下，自左向右。这样，每个结点对应一个编号，对于编号为i的结点，若有双亲，则其双亲为$\lfloor i/2\rfloor$，若有左孩子，则左孩子为$2i$；若有右孩子，则右孩子为$2i+1$。

（2）完全二叉树。高度为h、有n个结点的二叉树，当且仅当其每个结点都与高度为h的满叉树中编号为$1\sim n$的结点一一对应时，称为完全二叉树，如图（b）所示。其特点如下：

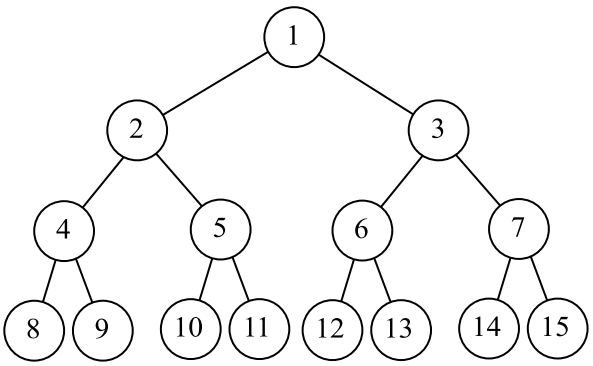
①若$i\leq\lfloor n/2\rfloor$，则结点i为分支结点，否则为叶子结点。

②叶子结点只可能在层次最大的两层上出现。对于最大层次中的叶子结点，都依次排列在该层最左边的位置上。

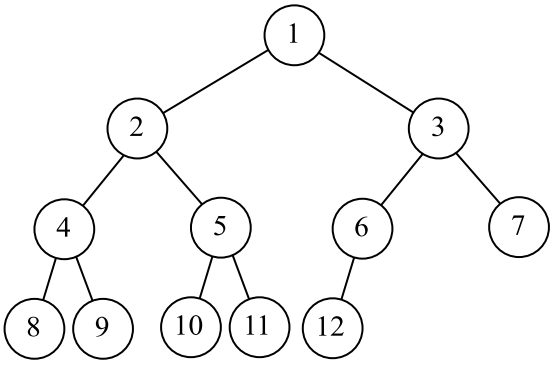
③若有度为1的结点，则只可能有一个，且该结点只有左孩子而无右孩子。

④按层序编号后，一旦出现某结点（编号为i）为叶子结点或只有左孩子，则编号大于i的结点均为叶子结点。

⑤若n为奇数，则每个分支结点都有左孩子和右孩子；若n为偶数，则编号最大的分支结点（编号为$n/2$）只有左孩子，没有右孩子，其余分支结点左、右孩子都有。



(*a*)满二叉树



(*b*)完全二叉树

（3）二叉排序树。左子树上所有结点的关键字均小于根结点的关键字；右子树上的所有结点的关键字均大于根结点的关键字；左子树和右子树又各是一颗二叉排序树。

（4）平衡二叉树。树上任一结点的左子树和右子树的深度之差绝对值不超过1。

二叉树的性质：

（1） 非空二叉树上的叶子结点数等于度为 2 的结点数加1，即 $n_0 = n_2 + 1$。

（2） 非空二叉树上第 k 层上至多有 2^{k-1} 个结点($k \geq 1$)。

（3） 高度为 h 的二叉树至多有 $2^h - 1$ 个结点($h \geq 1$)。

（4） 对完全二叉树按从上到下、从左到右的顺序依次编号 $1, 2, \cdots, n$，则有以下关系：

①当 $i > 1$ 时，结点 i 的双亲的编号为 $\lfloor i/2 \rfloor$，即当 i 为偶数时，其双亲的编号为 $i/2$，它是双亲的左孩子；当 i 为奇数时，其双亲的编号为 $(i-1)/2$，它是双亲的右孩子。

②当 $2i \leq n$ 时，结点 i 的左孩子编号为 $2i$，否则无左孩子。

③当 $2i + 1 \leq n$ 时，结点 i 的右孩子编号为 $2i + 1$，否则无右孩子。

（5） 具有 $n(n > 0)$ 个结点的完全二叉树的高度为 $\lceil \log_2(n+1) \rceil$ 或 $\lfloor \log_2 n \rfloor + 1$。

题 1. 一棵二叉树有67个结点，这些结点的度要么是0，要么是2。这棵二叉树中度为2的结点有_____个。

答案：33　　$(67-1)/2 = 33$

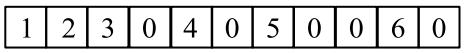
题 2. 将一棵有100个结点的完全二叉树，从根这一层开始，每一层从左到右依次对结点编号，根结点的编号为1，则编号为49的结点的双亲编号为（　　）。

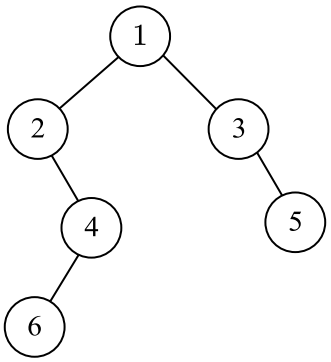
A. 23　　*B*. 25　　*C*. 24　　*D*. 无法确定

答案：*C*　　$\lfloor 49/2 \rfloor = 24$

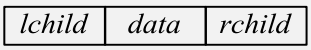
二叉树的存储结构：

（1）顺序存储结构：指用一组地址连续的存储单元依次自上而下、自左至右存储完全二叉树的结点元素，即将完全二叉树上编号为i的结点元素存储在一维数组下标$i-1$的分量中。



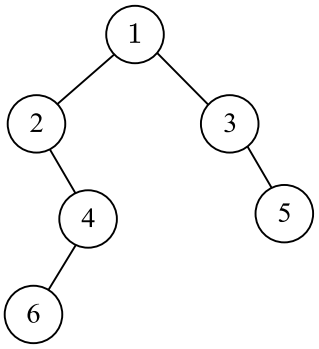


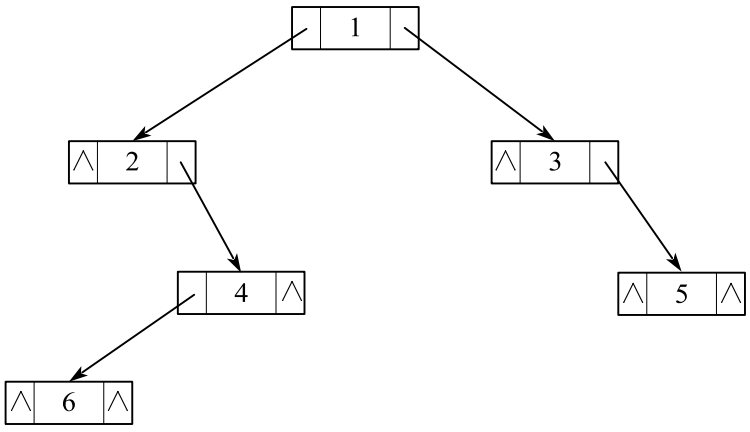
（2）链式存储结构：用链表结点来存储二叉树中的每个结点。在二叉树中，结点结构通常包括若干数据域和若干指针域，二叉链表至少包含3个域：数据域*data*，左指针域*lchild*和右指针域*rchild*。



二叉树的链式存储结构描述如下：

```
typedef struct BiTNode{
   ElemType data;                          //数据域
   struct BiTNode    *lchild,*rchild;      //左、右孩子指针
}BiTNode,*BiTree;
```





题 4. 若二叉树用二叉链表作存储结构，则在n个结点的二叉树链表中只有$n-1$个非空指针域。　　（　）

答案：正确

在含有n个结点的二叉链表中，含有$n+1$个空链域，含有$n-1$个非空链域。

3. 二叉树的遍历

二叉树的遍历是指按某条搜索路径访问树中每个结点，使得每个结点均被访问一次，而且仅被访问一次。

由二叉树的递归定义可知，遍历一颗二叉树便要决定对根结点N，左子树L，右子树R的访问顺序。按照先遍历左子树再遍历右子树的原则，常见的遍历次序有先序(NLR)、中序(LNR)和后序(LRN)三种遍历算法，其中“序”指的是根结点在何时被访问。

（1）先序遍历

先序遍历($PreOrder$)的操作过程如下：

若二叉树为空，则什么也不做；否则，

1）访问根结点；

2）先序遍历左子树；

3）先序遍历右子树；

对应的递归算法如下：

```
void PreOrder(BiTree T){
    if(T!=NULL){
        visit(T);                   //访问根结点
        PreOrder(T->lchild);        //递归遍历左子树
        PreOrder(T->rchild);        //递归遍历右子树
    }
}
```

（2）中序遍历

中序遍历($InOrder$)的操作过程如下，

若二叉树为空，则什么也不做，否则：

1）中序遍历左子树；

2）访问根结点；

3）中序遍历右子树；

对应的递归算法如下：

```
void InOrder(BiTree T){
    if(T!=NULL){
        InOrder(T->lchild);        //递归遍历左子树
        visit(T);                  //访问根结点
        InOrder(T->rchild);        //递归遍历右子树
    }
}
```

（3）后序遍历

后序遍历(*PostOrder*)的操作过程如下。

若二叉树为空，则什么也不做；否则，

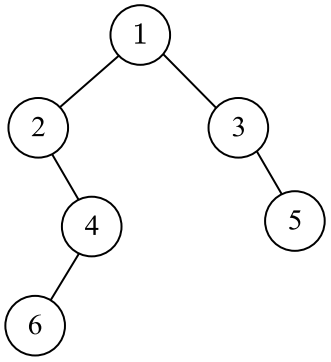
1）后序遍历左子树；

2）后序遍历右子树；

3）访问根结点。

对应的递归算法如下：

```
void PostOrder(BiTree T){
    if(T!=NULL){
        PostOrder(T->lchild);      //递归遍历左子树
        PostOrder(T->rchild);      //递归遍历右子树
        visit(T);                  //访问根结点
    }
}
```



先序遍历所得到的结点序列为1 2 4 6 3 5。
中序遍历所得到的结点序列为2 6 4 1 3 5。
后序遍历所得到的结点序列为6 4 2 5 3 1。

（4）层次遍历

要进行层次遍历，需要借助一个队列。先将二叉树根结点入队，然后出队，访问出队结点，若它有左子树，则将左子树根结点入队；若它有右子树，则将右子树根结点入队。然后出队，访问出队结点……如此反复，直至队列为空。

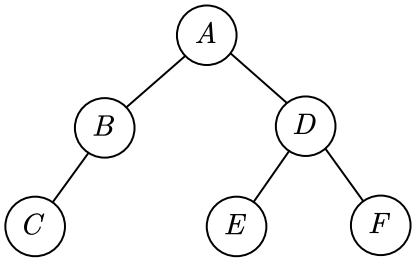
二叉树的层次遍历算法如下：

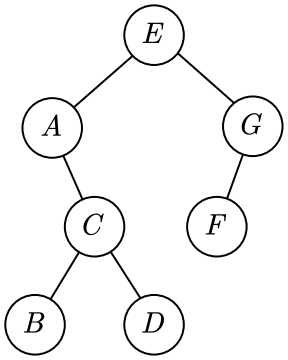
```
void LevelOrder(BiTree T){
    InitQueue(Q);                       //初始化辅助队列
    BiTNode *p=T;
    EnQueue(Q,p);                       //将根结点入队
    while(!IsEmpty(Q)){                 //队列不空则循环
        DeQueue(Q,p);                   //队头元素出队
        visit(p);                       //访问出队结点
        if(p->lchild!=NULL)
            EnQueue(Q,p->lchild);       //左子树不空，左子树根结点入队
        if(p->rchild!=NULL)
            EnQueue(Q,p->rchild);       //右子树不空，右子树根结点入队
    }
}
```

题 1. 已知一棵二叉树的先序遍历结果为 *ABCDEF*，中序遍历结果为 *CBAEDF*，则后序遍历的结果为（　　）。

A. *CBEFDA*　　*B*. *FEDCBA*　　*C*. *CBEDFA*　　*D*. 不定

答案：*A*





题 2. 某二叉树结点的中序序列为：*ABCDEFG*，后序序列为：*BDCAFGE*，则其左子树中结点数目为（　　）。

A. 3　　*B*. 2　　*C*. 4　　*D*. 5

答案：*C*

课时六　练习题

1. 设一棵树的度是4，其中度为0,1,2,3,4的结点个数分别是8,4,2,1和（　　）。

   A. 4　　*B*. 3　　*C*. 2　　*D*. 1

2. 设一棵m叉树中有N_1个度数为1的结点，N_2个度数为2的结点，……，N_m个度数为m的结点，则该树中共有（　　）个叶子结点。

   $A.\sum_{i=1}^{m}(i-1)N_i$　　$B.\sum_{i=1}^{m}N_i$　　$C.\sum_{i=2}^{m}N_i$　　$D.1+\sum_{i=2}^{m}(i-1)N_i$

3. 树最适合用来表示（　　）。

   A.有序数据元素　　*B*.无序数据元素

   C.元素之间具有分支层次关系的数据　　*D*.元素之间无联系的数据

4. 设T是一棵二叉树，T中有n个叶子结点，且非叶子结点都是具有两个孩子的结点，那么T中共有（　　）个结点。

   $A.2n-1$　　$B.2n$　　$C.2n+1$　　$D.2(n+1)$

5. 深度为h的完全二叉树至少有_____个结点，至多有_____个结点。

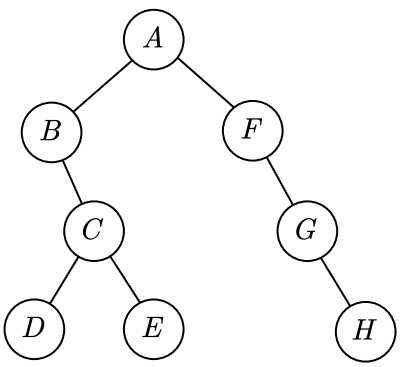
6. 一个具有1025个结点的二叉树的高h为（　　）。

   A.11　　*B*.10　　*C*.11～1025　　*D*.12～1025

7. 若10个结点的完全二叉树采用顺序表存储，下标范围0～9，根结点下标为0，则下标为3的结点的左孩子的下标是_____。

   A. 5　　*B*. 6　　*C*. 7　　*D*. 8

8. 一棵二叉树为下图，则后序序列为_____，中序序列为_____，先序序列为_____。



9. 已知某二叉树的后序遍历序列是*dabec*，中序遍历序列是*debac*，它的前序遍历序列为（　　）。

A. acbed　　*B. decab*　　*C. deabc*　　*D. cedba*

10. 已知一棵二叉树的先序遍历为：*ABDCEF*，中序遍历为：*DBAECF*，要求：

（1）画出这棵二叉树；

（2）写出这棵二叉树的后序遍历序列。

课时七　树和二叉树（2）

考点	重要程度	占分	题型
1. 树和森林	★★★★★	2～4	解答
2. 二叉排序树	★★★★★	4～6	
3. 哈夫曼树	必考	6～10	

1. 树和森林

树的存储结构：双亲表示法、孩子表示法、孩子兄弟表示法。

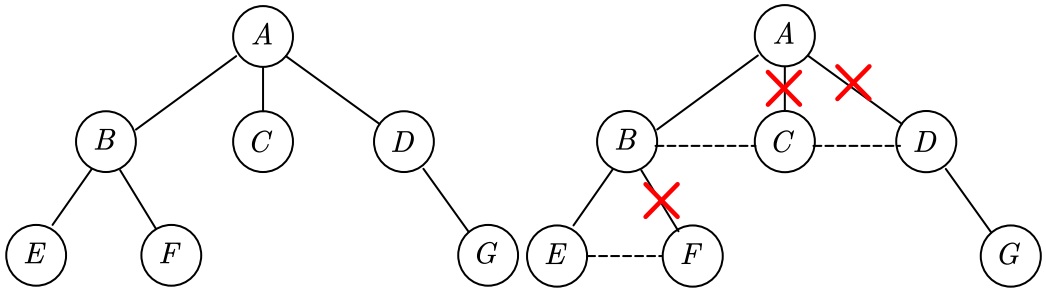
树转换成二叉树的画法：

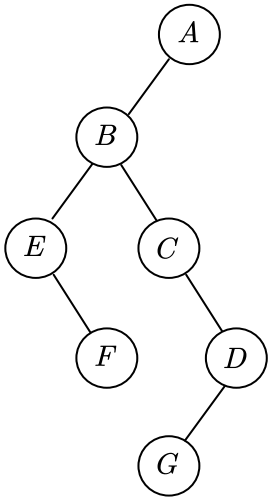
① 在兄弟结点之间加一条线；

② 对每个结点，只保留它与第一个孩子的连线，抹去与其他孩子的连线；

③ 以树根为轴心，顺时针旋转45°。

题 1. 将树转换成二叉树。





题 2. 由树转换成二叉树，其根结点的右子树总是空的。　　（　　）

答案：正确

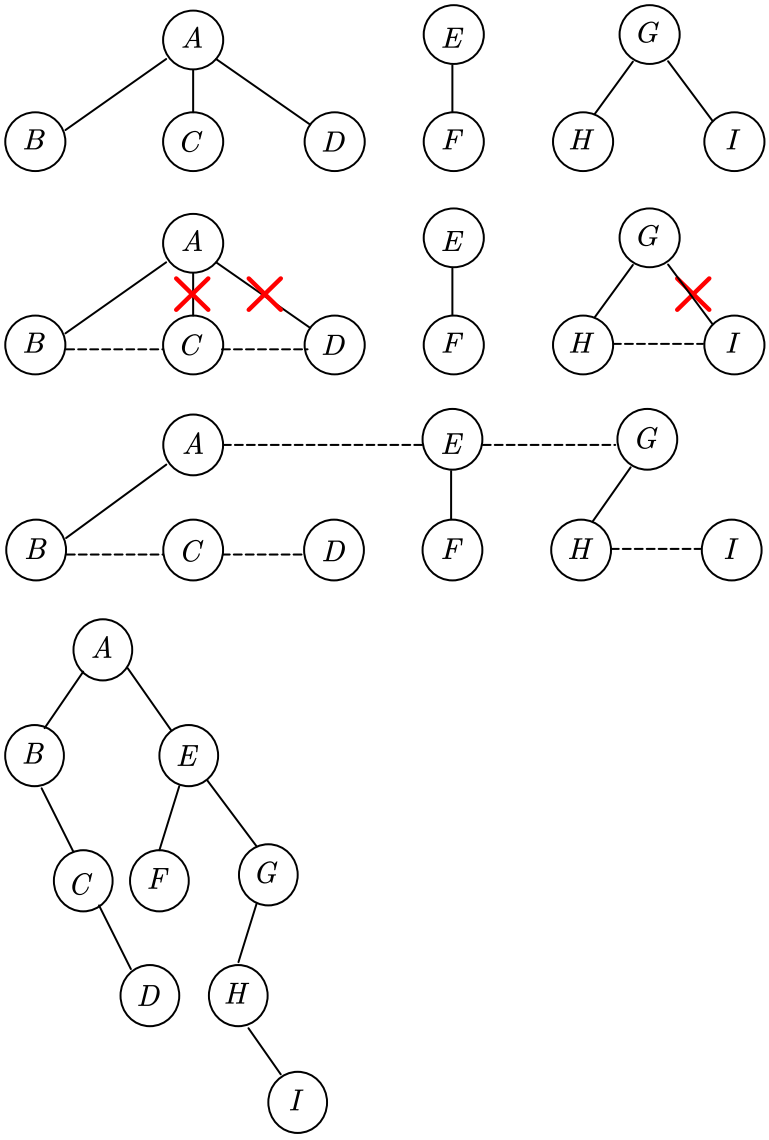
森林转换成二叉树的画法：

① 将森林中的每棵树转换成相应的二叉树；

② 每棵树的根也可视为兄弟结点，在每棵树之间加一根连线；

③ 以第一棵树的根为轴心顺时针旋转 45° 。

题 3. 将森林转换成二叉树。

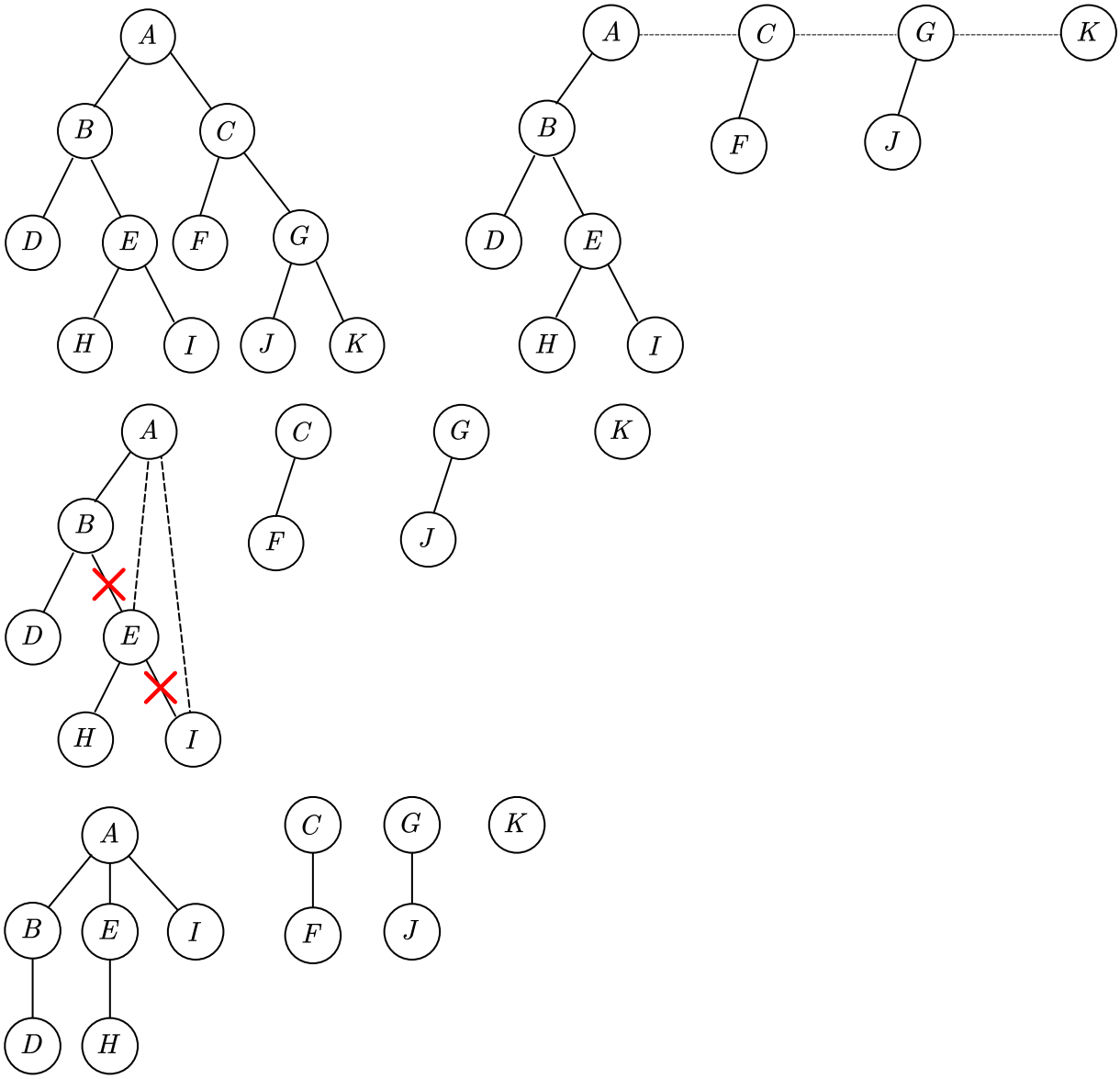


题 4. 设森林 F 中有三棵树，第一、第二、第三棵树上的结点个数分别为 M_1, M_2, M_3，则与森林 F 对应的二叉树根结点的右子树上的结点个数为（　　）。

$A.\ M_1$　　$B.\ M_1+M_2$　　$C.\ M_3$　　$D.\ M_2+M_3$

答案：D

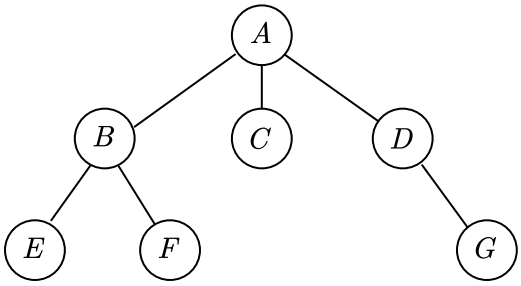
题 5. 画出下图中的二叉树对应的森林。



树的遍历是指用某种方式访问树中的每个结点，且仅访问一次。主要有两种方式：

（1） 先根遍历。若树非空，先访问根结点，再依次遍历根结点的每棵子树，遍历子树时仍遵循先根后子树的规则。其遍历次序与这棵树对应二叉树的先序序列相同。

（2） 后根遍历。若树非空，先依次遍历根结点的每棵子树，再访问根结点，遍历子树时仍遵循先子树后根的规则。其遍历次序与这棵树对应二叉树的中序序列相同。

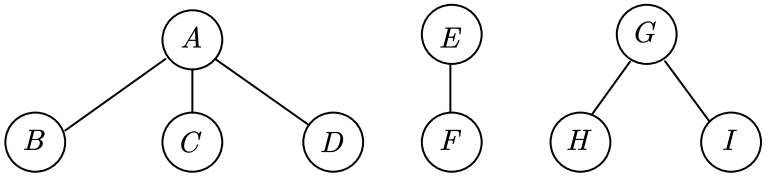


先根遍历序列为 $A\ B\ E\ F\ C\ D\ G$，后根遍历序列为 $E\ F\ B\ C\ G\ D\ A$。

森林的两种遍历方法：

（1） 先序遍历。若森林非空，先访问森林中第一棵子树的根结点，再先序遍历第一棵树中根结点的子树森林，再先序遍历除去第一棵树之后剩余的树构成的森林。

（2） 中序遍历。若森林非空，先中序遍历森林中第一棵树的根结点的子树森林，再访问第一棵树的根结点，再中序遍历除去第一棵树之后剩余的树构成的森林。



先序遍历序列为 $A\ B\ C\ D\ E\ F\ G\ H\ I$，中序遍历序列为 $B\ C\ D\ A\ F\ E\ H\ I\ G$。

树	森林	二叉树
先根遍历	先序遍历	先序遍历
后根遍历	中序遍历	中序遍历

2. 二叉排序树

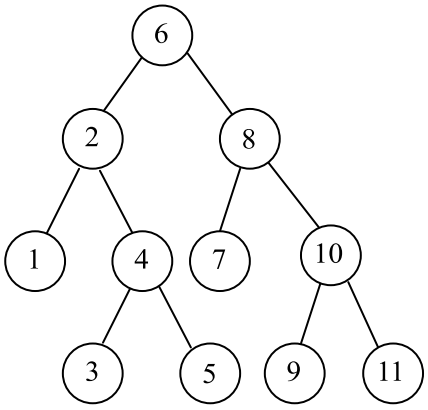
二叉排序树（二叉查找树）或者是一颗空树，或者是具有下列特性的二叉树。

（1）若左子树非空，则左子树上所有结点的值均小于根结点的值；

（2）若右子树非空，则右子树上所有结点的值均大于根结点的值；

（3）左、右子树也分别是一棵二叉排序树。

根据二叉排序树的定义，左子树结点值＜根结点值＜右子树结点值，所以对二叉排序树进行中序遍历，可以得到一个递增的有序序列。



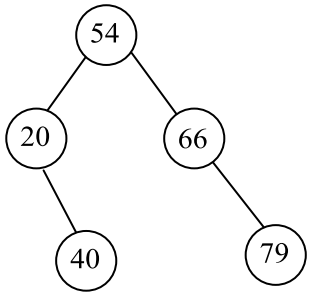
二叉排序树的中序遍历为1 2 3 4 5 6 7 8 9 10 11 。

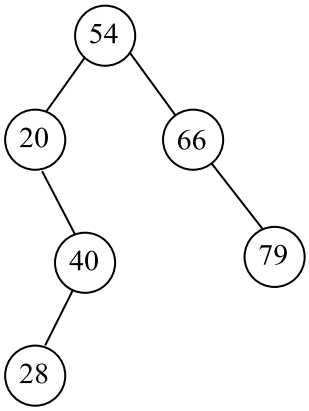
题 1. 若对一棵二叉排序树构成的序列采用中序遍历，可得到（　　）结果。

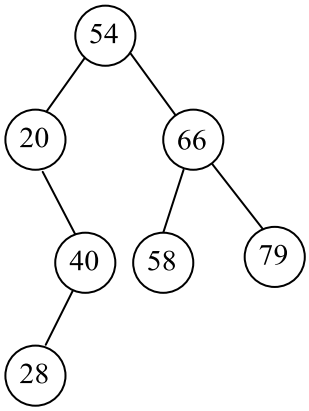
A. 降序　　*B*. 升序　　*C*. 无序　　*D*. 不确定

答案：*B*

二叉树的插入：
若原二叉排序树为空，则直接插入结点；否则，若关键字 k 小于根结点值，则插入到左子树，若关键字 k 大于根结点值，则插入到右子树。





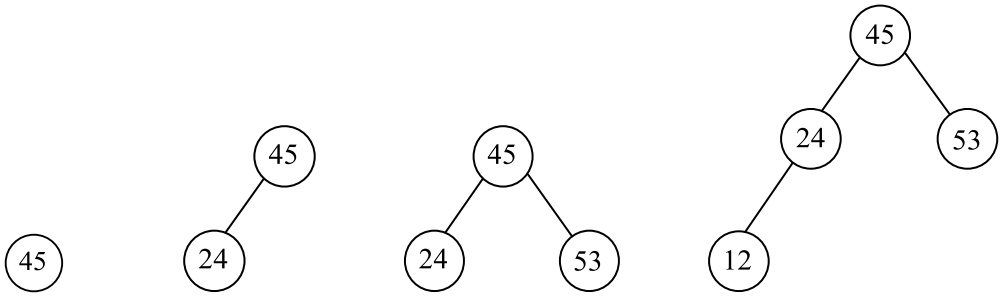


题 2. 向二叉排序树插入一个新结点时，新结点一定会成为二叉排序树的一个叶子结点。（　　）

答案：正确

二叉排序树的构造：从一棵空树出发，依次输入元素，将它们插入二叉排序树的合适位置。

题 3. 设查找的关键字序列是{45, 24, 53, 12}，构造一棵二叉排序树。

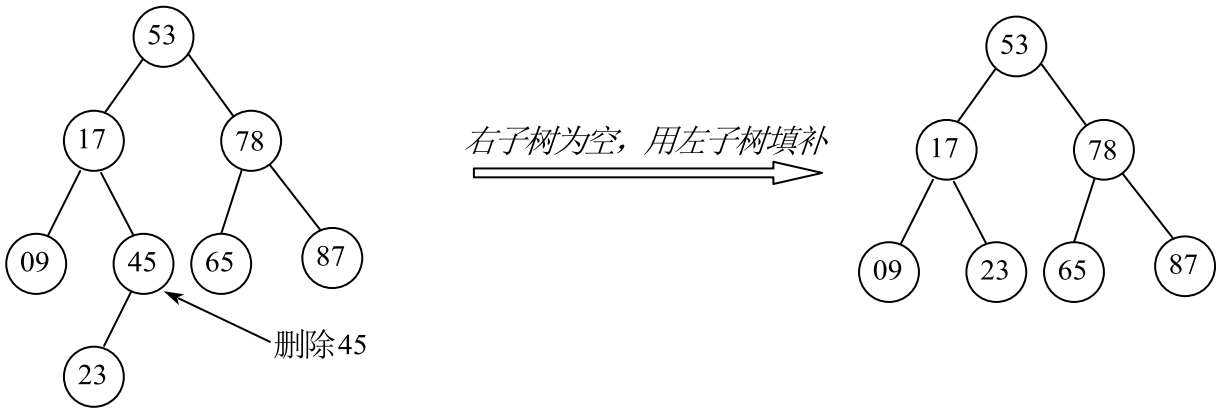


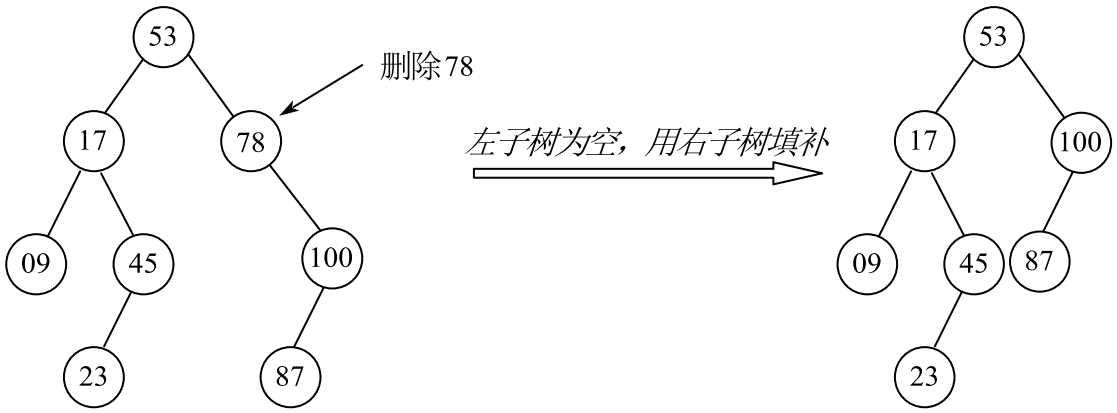
二叉排序树的删除：

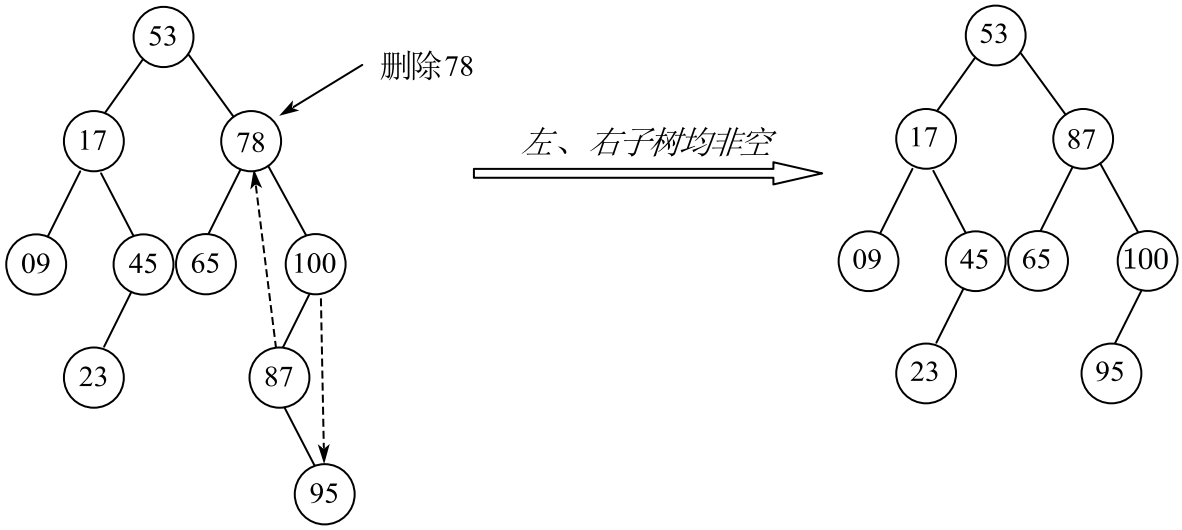
（1）若被删除的结点是叶子结点，则直接删除；

（2）若结点只有一棵左子树或者右子树，则让该结点的子树成为其父结点的子树；

（3）若结点有左、右两棵子树，则令该结点的直接后继（直接前驱）代替该结点，然后从二叉排序树中删去这个直接后继（直接前驱），这样就转换成了前两种情况。







3. 哈夫曼树

树中结点常被赋予一个代表某种意义的数值，那个数值称为该结点的权。

从树的根到任意结点的路径长度（经过的边数）与该结点上权值的乘积，称为该结点的带权路径长度。

树中所有叶结点的带权路径长度之和称为该树的带权路径长度，记为 $WPL=\sum_{i=1}^{n} w_i l_i$。

带权路径长度最小的二叉树称为哈夫曼树，也称最优二叉树。

给定 n 个权值分别为 $w_1, w_2, \cdots, w_n$ 的结点，构造哈夫曼树的算法描述如下：

（1）将这 n 个结点分别作为 n 棵仅含一个结点的二叉树，构成森林 F。

（2）构造一个新结点，从 F 中选取两棵根结点权值最小的树作为新结点的左、右子树，并且将新结点的权值置为左、右子树上根结点的权值之和。

（3）从 F 中删除刚才选出的两棵树，同时将新得到的树加入 F 中。

（4）重复步骤（2）和（3），直至 F 中只剩下一棵树为止。

题 1. 有一电文使用五种字符 a,b,c,d,e，其出现频率依次为 $4,7,5,2,9$。

（1）试画出对应的哈夫曼树（要求左子树根结点的权小于等于右子树根结点的权）。

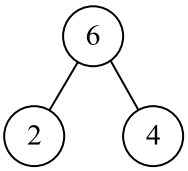
（2）求出每个字符的哈夫曼编码。

（3）译出编码序列 11000111000101011 的相应电文。

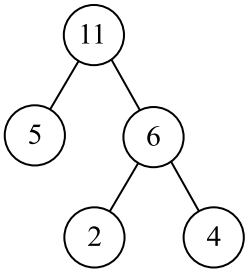
（4）求带权路径长度。

答案：（1）

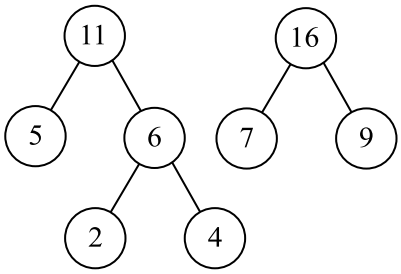
4, 7, 5, 2, 9



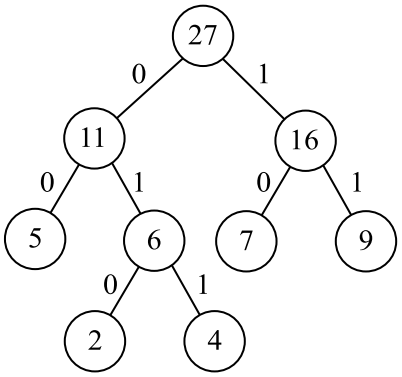
6, 7, 5, 9



11, 7, 9



11, 16



27

（2）a 对应的哈夫曼编码为 011；

b 对应的哈夫曼编码为 10；

c 对应的哈夫曼编码为 00；

d 对应的哈夫曼编码为 010；

e 对应的哈夫曼编码为 11；

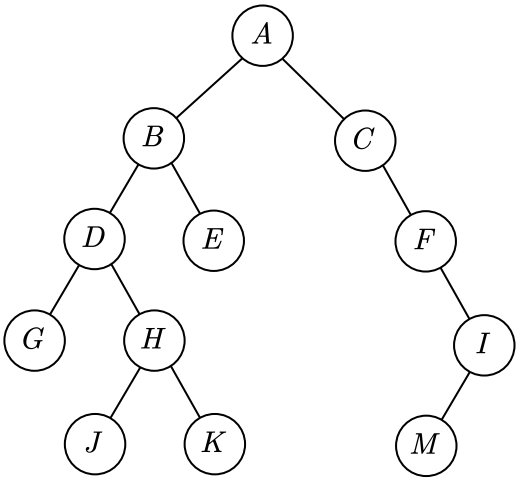
（3）编码序列 11000111000101011 的相应电文为 $ecabcbbe$。

（4）$WPL = 5\times2 + 2\times3 + 4\times3 + 7\times2 + 9\times2 = 60$

课时七　练习题

1. 由树转换成二叉树，其根的右孩子指针总是空的。（　　）

2. 画出下列二叉树对应的森林。



3. 设森林F对应的二叉树为B，它有m个结点，B的根为P，P的右子树的结点个数为n，森林F中第一棵树的结点的个数是（　　）。

$A.\ m-n$　　$B.\ m-n-1$　　$C.\ n+1$　　$D.$不能确定

4. 在下列存储形式中，（　　）不是树的存储形式。

$A.$双亲表示法　　$B.$孩子链表表示法　　$C.$孩子兄弟表示法　　$D.$顺序存储表示法

5. 如果F是由有序树T转换而来的二叉树，那么T中结点的后根序列就是F中结点的（　　）序列。

$A.$前序　　$B.$中序　　$C.$后序　　$D.$层次

6. 假设一棵二叉树的层次序列（按层次递增顺序排列，同一层次自左向右）为$ABECFGDHI$，中序序列为$BCDAFEHIG$。请画出该二叉树，并将其转换成对应的森林。

7. 设用于通信的电文仅由7个字母a, b, c, d, e, f, g组成，字母在电文中出现的频率分别是$0.07, 0.19, 0.06, 0.34, 0.03, 0.21, 0.10$

（1）请为这7个字母构建$Huffman$树，写出每个字母的$Huffman$编码。

（2）求该哈夫曼树的带权路径长度WPL。（要求左子树根结点的权小于等于右子树根结

点的权）

8. 依次输入一个关键字序列{60, 27, 76, 66, 80, 22, 70, 62}

（1）按输入次序构造并画出此二叉排序树；

（2）画出该树在删除关键字“76”后的二叉排序树。

9. 如下图所示是一棵二叉排序树，现对它做如下插入和删除操作。

（1）插入2；

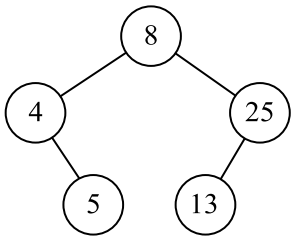
（2）删除13；

（3）插入34；

（4）插入26；

（5）删除4；

（6）删除8；



课时八　图

考点	重要程度	占分	题型
1. 图的基本概念	★★★★	2～4	选择、填空、判断
2. 图的存储结构	★★★★		
3. 图的遍历	★★★★★	4～6	解答、填空

1. 图的基本结构

（1）有向图。若E是有向边（弧）的有限集合时，则图G为有向图。弧是顶点的有序对，记为$<v,w>$，其中v,w是顶点，v称为弧尾，w称为弧头，$<v,w>$称为从顶点v到顶点w的弧。

（2）无向图。若E是无向边（边）的有限集合时，则图G为无向图。边是顶点的无序对，记为(v,w)或(w,v)，因为两者相等，其中v,w是顶点，顶点v,w互为邻接点。边(v,w)依附于顶点v,w，或者边(v,w)和顶点v,w相关联。

（3）简单图。一个图G若满足：①不存在重复边；②不存在顶点到自身的边，则称图G为简单图。

（4）多重图。若图G中某两个结点之间的边数多于一条，又允许顶点通过一条边和自己关联，则图G为多重图。

（5）完全图（简单完全图）。对于无向图，任意两个顶点之间都存在边。

对于有向图，任意两个顶点之间都存在方向相反的两条弧。

（6）子图。设有两个图$G=(V,E)$和$G'=(V',E')$，若V'是V的子集，且E'是E的子集，则称G'是G的子图。若有满足$V(G')=V(G)$的子图G'，则称其为G的生成子图。

（7）连通、连通图。

在无向图中，若从顶点v到顶点w有路径存在，则称v和w是连通的。

若图G中任意两个顶点都是连通的，则称图G是连通图。

极小连通子图是既要保持图连通又要使得边数最少的子图。

（8）生成树。连通图的生成树是包含图中全部顶点的一个极小连通子图。

（9）顶点的度、入度和出度。

对于无向图，顶点v是依附于该顶点的边的条数。无向图的全部顶点的度的和等于边数的两倍。

对于有向图，入度是以顶点v为终点的有向边的数目。出度是以顶点v为起点的有向边的数目。顶点的度等于入度和出度之和。有向图的全部顶点的入度之和与出度之和相等，并且等于边数。

（10）边的权和网。在一个图中，每条边都可以标上具有某种意义的数值，该数值称为该边的权值。这种边上带有权值的图称为带权图，也称网。

（11）路径、路径长度和回路。

顶点v_p到顶点v_q之间的一条路径是指顶点序列$v_p,v_{i1},v_{i2},\cdots,v_{im},v_q$。

路径上边的数目称为路径长度。第一个顶点和最后一个顶点相同的路径称为回路或环。

题 1. 设有 6 个结点的无向图，该图至少应有（　　）条边才能确保是一个连通图。

A. 5　　*B*. 6　　*C*. 7　　*D*. 8

答案：*A*

题 2. 在一个图中，所有顶点的度数之和等于边数的（　　）倍。

A. 1/2　　*B*. 1　　*C*. 2　　*D*. 4

答案：*C*

2. 图的存储结构

图的存储结构有四种：邻接矩阵法、邻接表法、十字链表法、邻接多重表法。

（1）邻接矩阵法

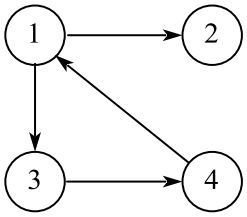
邻接矩阵存储，是指用一个一维数组存储图中顶点的信息，用一个二维数组存储图中边的信息（即各顶点之间的邻接关系），存储顶点之间邻接关系的二维数组称为邻接矩阵。

结点数为n的图$G(V,E)$的邻接矩阵A是$n\times n$的。将G的顶点编号为$v_1,v_2,\cdots,v_n$。若$(v_i,v_j)\in E$，则$A[i][j]=1$，否则$A[i][j]=0$。

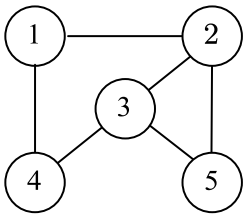
$$A[i][j]=\begin{cases}1, & \text{若}(v_i,v_j)\text{或}\langle v_i,v_j\rangle\text{是}E(G)\text{中的边}\\ 0, & \text{若}(v_i,v_j)\text{或}\langle v_i,v_j\rangle\text{不是}E(G)\text{中的边}\end{cases}$$

对于带权图而言，若顶点v_i和v_j之间有边相连，则邻接矩阵中对应项存放着该边对应的权值，若顶点v_i和v_j不相连，则用∞来表示这两个顶点之间不存在边。

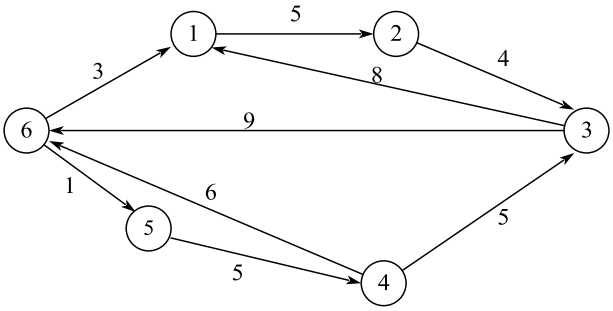
$$A[i][j]=\begin{cases}w_{ij}, & \text{若}(v_i,v_j)\text{或}\langle v_i,v_j\rangle\text{是}E(G)\text{中的边}\\ 0\text{或}\infty, & \text{若}(v_i,v_j)\text{或}\langle v_i,v_j\rangle\text{不是}E(G)\text{中的边}\end{cases}$$



$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$



$$\begin{bmatrix} \infty & 5 & \infty & \infty & \infty & \infty \\ \infty & \infty & 4 & \infty & \infty & \infty \\ 8 & \infty & \infty & \infty & \infty & 9 \\ \infty & \infty & 5 & \infty & \infty & 6 \\ \infty & \infty & \infty & 5 & \infty & \infty \\ 3 & \infty & \infty & \infty & 1 & \infty \end{bmatrix}$$

图的邻接矩阵存储表示法具有以下特点：

（1）无向图的邻接矩阵一定是个对称矩阵。

（2）对于无向图，邻接矩阵的第 i 行（或第 i 列）非零元素（或非 ∞ 元素）的个数正好是第 i 个顶点的度。

（3）对于有向图，邻接矩阵的第 i 行（或第 i 列）非零元素（或非 ∞ 元素）的个数正好是第 i 个顶点的出度（或入度）。

题 1. n 个顶点的无向连通图用邻接矩阵表示时，该矩阵至少有_____个非零元素。

答案：$2(n-1)$

题 2. 带权有向图 G 用邻接矩阵 A 存储，则顶点 i 的入度等于 A 中（　　）。

A. 第 i 行非无穷元素之和　　*B.* 第 i 列非无穷元素之和

C. 第 i 行非零非无穷元素之和　　*D.* 第 i 列非零非无穷元素之和

答案：*D*

题 3. 用邻接矩阵存储一个图时，在不考虑压缩存储的情况下，所占用的存储空间大小只与图中顶点的个数有关，而与图的边数无关。（　　）

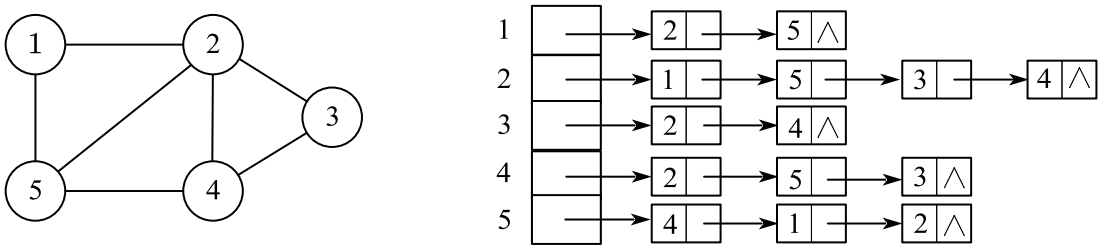
答案：正确

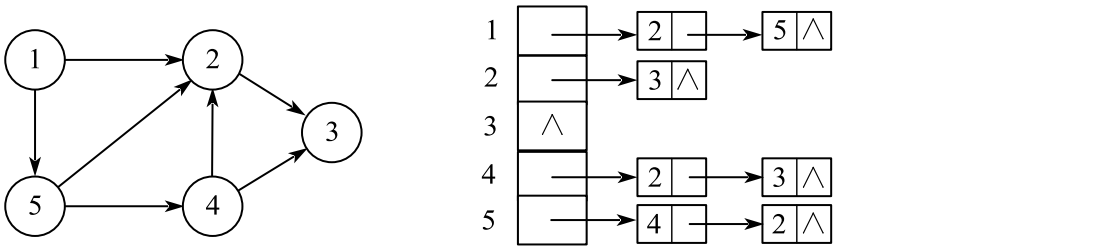
（2）邻接表法

邻接表，是指对图G中的每个顶点v_i建立一个单链表，第i个单链表中的结点表示依附于顶点v_i的边，这个单链表就称为顶点v_i的边表。

边表的头指针和顶点的数据信息采用顺序存储（称为顶点表），所以在邻接表中存在两种结点：顶点表结点和边表结点。

顶点域	边表头指针	邻接点域	指针域
data	*firstarc*	*adjvex*	*nextarc*





图的邻接表存储表示法具有以下特点：

（1）若G为无向图，则所需的存储空间为$O(|V|+2|E|)$；若G为有向图，则所需的存储空间为$O(|V|+|E|)$。

（2）对于稀疏图，采用邻接表表示能极大地节省存储空间。

（3）图的邻接表表示不唯一。

（4）在有向图的邻接表表示中，求一个给定顶点的出度只需计算其邻接表中的结点个数。

题 4. n个结点，e条边的无向图邻接表中，有_____个头结点和_____个表结点。

答案：n，$2e$

题 5. 在有向图的邻接表表示中，下面（　　）最费时间。

A. 求某顶点的出度　　*B*. 求某顶点的入度

C. 求图中顶点的个数　　*D*. 求从某顶点出发的弧

答案：*B*

3. 图的遍历

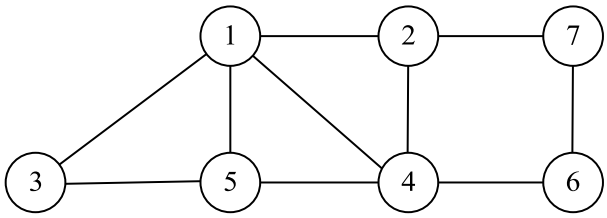
图的遍历是指从图中的某一顶点出发，按照某种搜索方法沿着图中的边对图中的所有顶点访问一次且仅访问一次。

图的遍历算法主要有两种：广度优先搜索和深度优先搜索。

（1） 广度优先搜索

广度优先搜索类似于二叉树的层序遍历算法。其基本思想是：首先访问起始顶点 v，接着由 v 出发，依次访问 v 的各个未访问过的邻接顶点 $w_1, w_2, \cdots, w_i$，然后依次访问 $w_1, w_2, \cdots, w_i$ 的所有未被访问过的邻接顶点；再从这些访问过的顶点出发，访问它们所有未被访问过的邻接顶点，直至图中的所有顶点都被访问过为止。若此时图中尚有顶点未被访问，则另选图中的一个未被访问的顶点作为始点，重复上述过程，直至图中所有顶点都被访问到为止。

题 1. 如下图，从顶点 1 出发，按照广度优先规则遍历，可能得到的序列是（　　）。



A. 1352467　　*B*. 1423756　　*C*. 1234576　　*D*. 1354672

答案：*C*

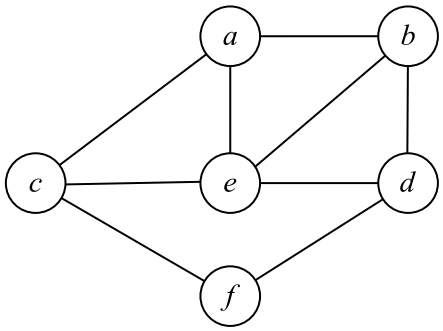
（2） 深度优先搜索

广度优先搜索类似于树的先序遍历。其基本思想是：首先访问起始顶点v，然后由v出发，访问与v邻接且未被访问的任一顶点w_1，再访问与w_1邻接且未被访问的任一顶点w_2……重复上述操作。当不能再继续向下访问时，依次退回到最近被访问的顶点，若它还有邻接顶点未被访问过，则从该点开始继续上述搜索过程，直至图中所有顶点均被访问过为止。

题 2. 若无向图G中的边的集合$E=\{(a,b),(a,e),(a,c),(b,e),(e,d),(d,f),(f,c)\}$，则从顶点$a$出发进行深度优先遍历，可以得到的一种顶点序列为（　　）。

A. aedfcb　　*B. acfebd*　　*C. aebcfd*　　*D. aedfbc*

答案：*A*



题 3. 深度优先搜索遍历类似于树的_____遍历，广度优先搜索遍历类似于树的_____遍历，它们可以用_____、_____两种数据结构来实现。

答案：先序，层次，栈，队列

题 4. 如果从一个无向图的任意顶点出发进行一次深度优先搜索即可访问所有顶点，则该图一定是_____。

答案：连通图

课时八　练习题

1. $n(n>0)$个顶点的无向图最多有_____条边，最少有_____条边。

2. 在一个有向图中，所有顶点的入度之和等于所有顶点的出度之和的_____倍。

3. 下列关于无向连通图特性的叙述中，正确的是（　　）。

（1） 所有顶点的度数之和为偶数

（2） 边数小于顶点个数减1

（3） 至多有一个顶点的度数为1

A.（1）　　*B.*（2）　　*C.*（1）（2）　　*D.*（1）（3）

4. 下面关于图的存储结构中正确的是（　　）。

*A.*用邻接表法存储图，占用的存储空间大小只与图中边数有关，而与结点个数无关

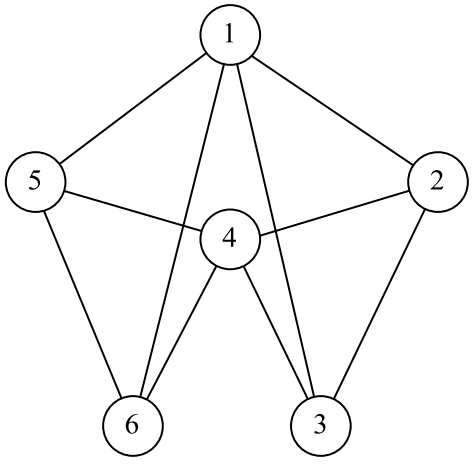
*B.*用邻接表法存储图，占用的存储空间大小与图中边数和结点个数都有关

*C.*用邻接矩阵法存储图，占用的存储空间大小与图中边数和结点个数都有关

*D.*用邻接矩阵法存储图，占用的存储空间大小只与图中边数有关，而与结点个数无关

5. 邻接矩阵适用于稀疏图的存储，邻接表适用于稠密图的存储。（　　）

6. 如下图所示一个无向图，试分别给出它的邻接矩阵和邻接表。

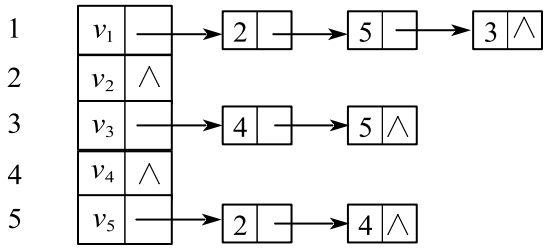


7. 如图所示是一个无向连通网的邻接矩阵，试画出该网。

$$\begin{bmatrix} 0 & 6 & 1 & 5 & \infty & \infty \\ 6 & 0 & 5 & \infty & 3 & \infty \\ 1 & 5 & 0 & 5 & 6 & 4 \\ 5 & \infty & 5 & 0 & \infty & 2 \\ \infty & 3 & 6 & \infty & 0 & 6 \\ \infty & \infty & 4 & 2 & 6 & 0 \end{bmatrix}$$

8. 对任意一个图，从它的某个顶点出发进行一次深度优先或广度优先搜索遍历可访问到该图的各个顶点。（　　）

9. 已知有向图的邻接表存储结构如下图所示，则按深度优先遍历算法从顶点 v_1 出发，所得到的顶点序列为（　　）。



A. v_1, v_5, v_3, v_4, v_2　　B. v_1, v_3, v_2, v_5, v_4

C. v_1, v_3, v_4, v_5, v_2　　D. v_1, v_3, v_2, v_4, v_5

10. 已知图的邻接矩阵如下：试给出邻接表结构，并给出从顶点1出发进行深度优先和广度优先的搜索结果。

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

课时九　图的应用

考点	重要程度	占分	题型
1. 最小生成树	必考	6～10	选择、填空、判断、解答
2. 最短路径	★★★★★	4～8	
3. 拓扑排序	★★★★★	4～8	
4. 关键路径	★★★★★	4～8	

1. 最小生成树

对一个带权连通无向图，所有生成树中权值之和最小的生成树称为最小生成树。

最小生成树具有如下性质：

（1）最小生成树不一定是唯一的，即最小生成树的树形不唯一。

当带权连通无向图中的各边权值互不相等时，这时最小生成树才是唯一的。

（2）最小生成树的边的权值之和总是唯一的。

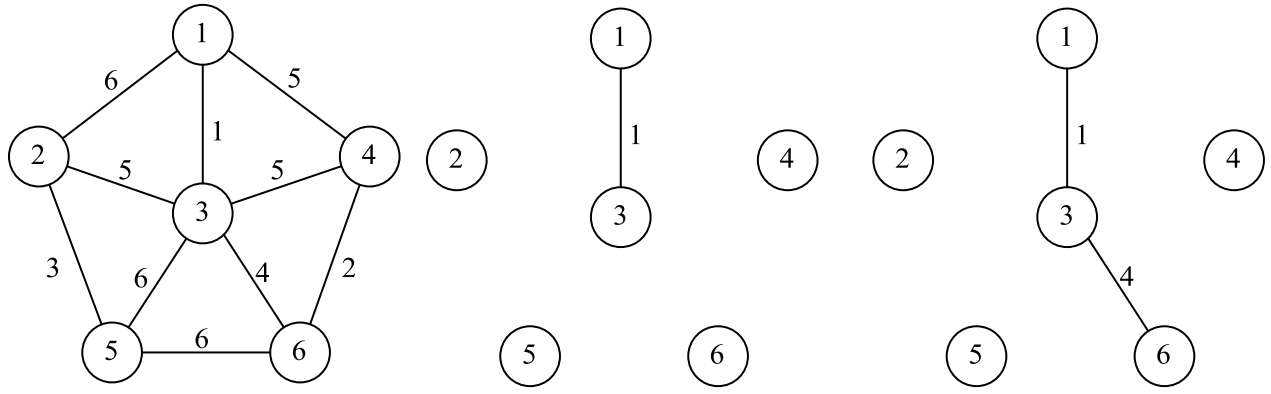
（3）最小生成树的边数为顶点数减1。

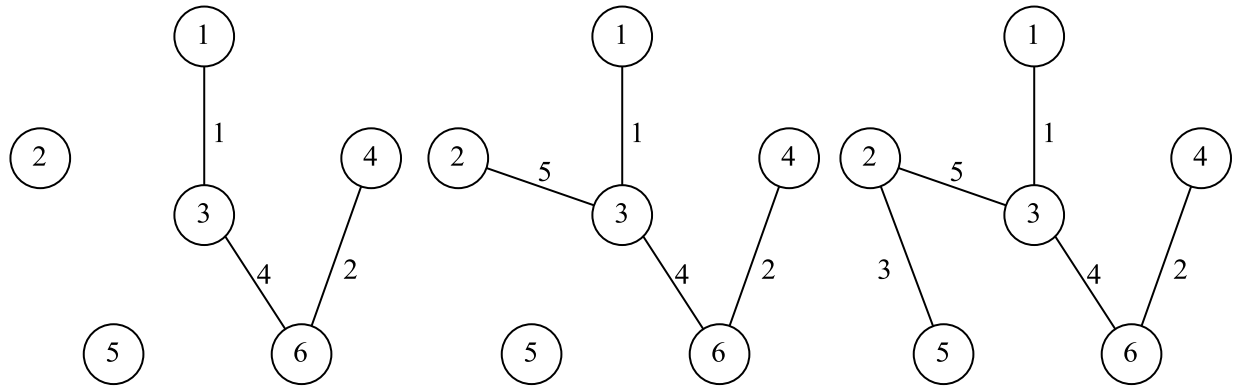
Prim 算法：

①任取一顶点（或题中已告知顶点），去掉所有边

②选择一个与当前顶点集合距离最近的顶点，并将该顶点和相应的边加入进来，同时不形成回路

③重复②，直至图中所有顶点都并入。



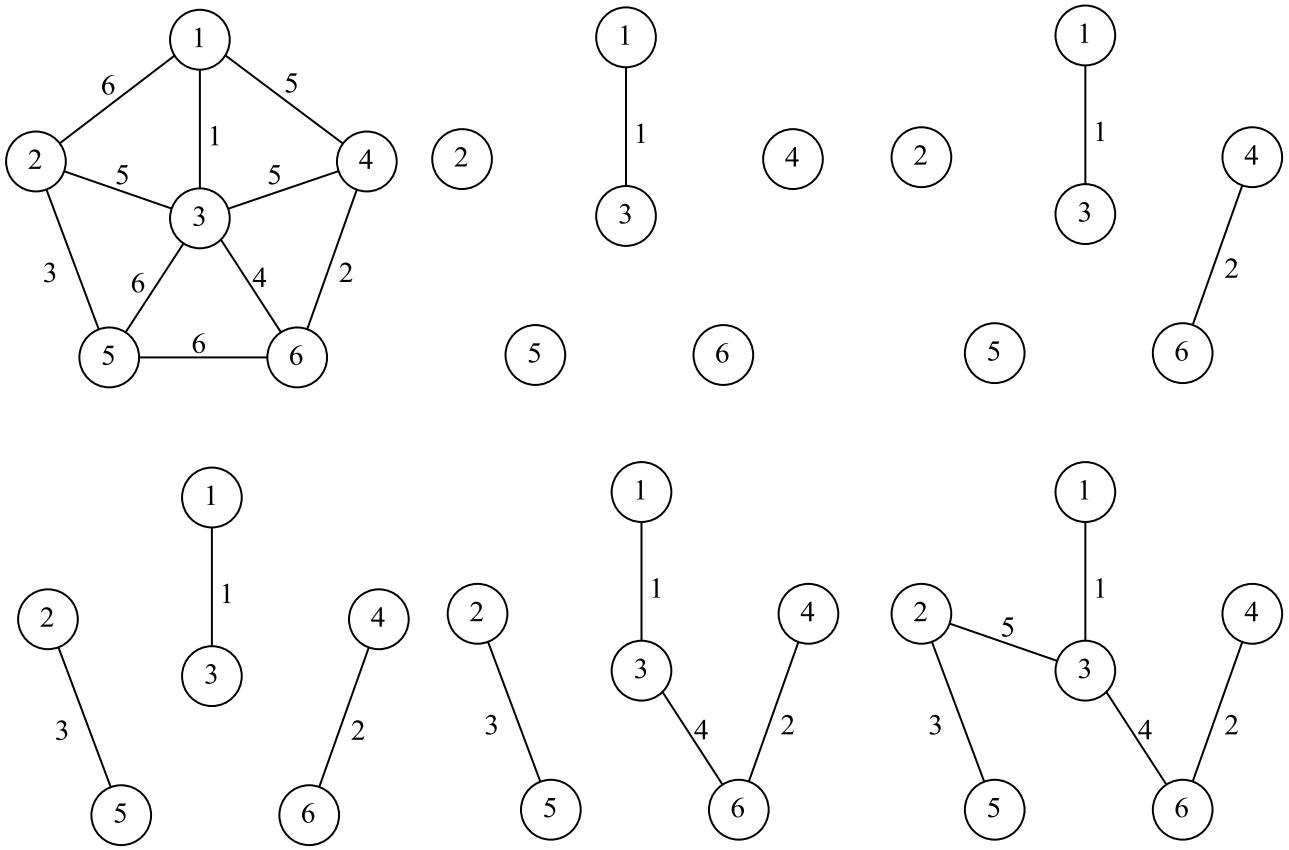


Kruskal 算法：

①去掉所有边

②选边（权最小，且不构成回路）

③一直重复第②步，直至图中所有顶点都并入



2. 最短路径

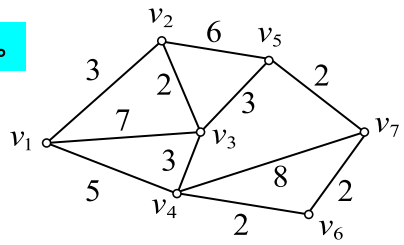
设图$G=<V,E>$（无向图或有向图），给定$W: E\to R$，对于G的每一条边e，称$W(e)$为边e的权，把这样的图称作带权图，记作$G=<V,E,W>$。当$e=(u,v)$（$<u,v>$）时，把$W(e)$记作$W(u,v)$。

设P是G中的一条通路，P中所有边的权之和称作P的长度，记作$W(P)$，即$W(P)=\sum_{e\in E(p)} W(e)$。类似地，可以定义为回路$C$的长度$W(C)$。

设带权图$G=<V,E,W>$（无向图和有向图），其中每一条边e的权$W(e)$为非负实数。$\forall u,v\in V$，当u和v连通（u可达v）时，称从u到v的最短路径，称其长度为从u到v的距离，记作$d(u,v)$。约定：$d(u,v)=0$；当u和v不连通（u不可达v）时，$d(u,v)=+\infty$。

最短路径问题：给定带权图$G=<V,E,W>$及顶点u和v，其中每一条边e的权$W(e)$为非负实数，求从u到v的最短路径。

题 1. 带权图G如图所示，求从v_1到其余各点的最短路径和距离。



步骤 / 顶点	1	2	3	4	5	6	7
v_1	0						
v_2	∞	3					
v_3	∞	7	5				
v_4	∞	5	5	5			
v_5	∞	∞	9	8	8	8	
v_6	∞	∞	∞	∞	7		
v_7	∞	∞	∞	∞	13	9	9
	0	3	5	5	7	8	9
	v_1	v_2	v_3	v_4	v_5	v_6	v_7

从v_1到其余各点的最短路径和距离如下：

v_1v_2　　$d(v_1,v_2)=3$

$v_1v_2v_3$　　$d(v_1,v_3)=5$

v_1v_4　　$d(v_1,v_4)=5$

$v_1v_2v_3v_5$　　$d(v_1,v_5)=8$

$v_1v_4v_6$　　$d(v_1,v_6)=7$

$v_1v_4v_6v_7$　　$d(v_1,v_7)=9$

3. 拓扑排序

AOV网：若用DAG图表示一个工程，其顶点表示活动，用有向边$\langle V_i,V_j\rangle$表示活动V_i必须先于活动V_j进行的这样一种关系，则将这种有向图称为顶点表示活动的网络，记为AOV网。

在AOV网中，活动V_i是活动V_j的直接前驱，活动V_j是活动V_i的直接后继，这种前驱和后继关系具有传递性，且任何活动V_i不能以它作为自己的前驱或后续。

在图论中，由一个有向无环图的顶点组成的序列，当且仅当满足下列条件时，称为该图的一个拓扑排序：

①每个顶点出现且只出现一次。

②若存在一条从顶点A到顶点B的路径，则在排序中顶点B出现在顶点A的后面。

每个AOV网都有一个或多个拓扑排序序列。

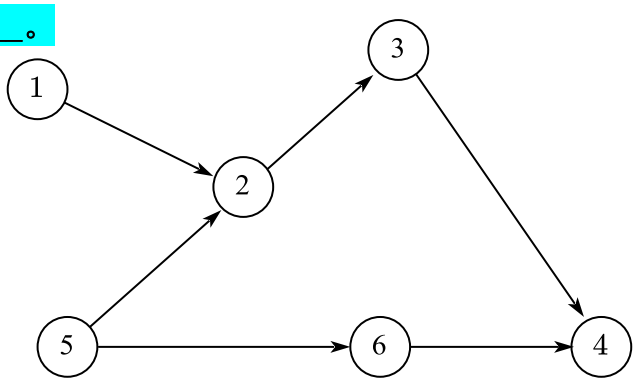
拓扑排序的算法的步骤：

①从AOV网中选择一个没有前驱的顶点并输出。

②从网中删除该顶点和所有以它为起点的有向边。

③重复①和②直到当前的AOV网为空或当前网中不存在无前驱的顶点为止。后一种情况说明有向图中必然存在环。

题 1. 下图的一个拓序排序序列为_____。



答案：152364

4. 关键路径

在带权有向图中，以顶点表示事件，以有向边表示活动，以边上的权值表示完成该活动的开销（如完成活动所需的时间），称之为用边表示活动的网络，简称 *AOE* 网。*AOE* 网和 *AOV* 网都是有向无环图，不同之处在于它们的边和顶点所代表的含义是不同的，*AOE* 网中的边有权值；而 *AOV* 网中的边无权值，仅表示顶点之间的前后关系。

从开始顶点到结束顶点的所有路径中，具有最大路径长度的路径称为关键路径。而把关键路径上的活动称为关键活动。

题 1. 关键路径是事件网络中（　　）。

A. 从源点到汇点的最短路径　　*B*. 从源点到汇点的最长路径

C. 最长的回路　　*D*. 最短的回路

答案：*B*

寻找关键活动时所用到的几个参量：

（1）事件 v_k 的最早发生时间 $ve(k)$：指从源点 v_1 到顶点 v_k 的最长路径长度。

可用下面的递推公式来计算：

ve（源点）$=0$

$ve(k)=Max\{ve(j)+Weight(v_j,v_k)\}$，$v_k$ 为 v_j 的任意后续，$Weight(v_j,v_k)$ 表示 $\langle v_j,v_k\rangle$ 上的权值。

注意：计算 $ve(k)$ 值时，按从前往后的顺序进行。

（2）事件 v_k 的最迟发生时间 $vl(k)$：指在不推迟整个工程完成的前提下，即保证它的后续事件 v_j 在其最迟发生时间 $vl(j)$ 能够发生时，该事件最迟必须发生的时间。

vl(汇点)=ve(汇点)

$vl(k)=Min\{vl(j)-Weight(v_k,v_j)\}$，$v_k$为$v_j$的任意前驱

注意：在计算$vl(k)$时，按从后往前的顺序进行。

（3）活动a_i的最早开始时间$e(i)$：指该活动弧的起点所表示的事件的最早发生时间。若边$\langle v_k,v_j\rangle$表示活动a_i，则有$e(i)=ve(k)$。

（4）活动a_i的最迟开始时间$l(i)$：指该活动弧的终点所表示事件的最迟发生时间与该活动所需时间之差。若边$\langle v_k,v_j\rangle$表示活动a_i，则有$l(i)=vl(j)-Weight(v_k,v_j)$。

（5）一个活动a_i的最迟开始时间$l(i)$和其最早开始时间$e(i)$的差额$d(i)=l(i)-e(i)$

$l(i)-e(i)=0$即$l(i)=e(i)$的活动a_i是关键活动。

求关键路径的算法步骤如下：

①从源点出发，令ve(源点)=0，按拓扑有序求其余顶点的最早发生时间ve()。

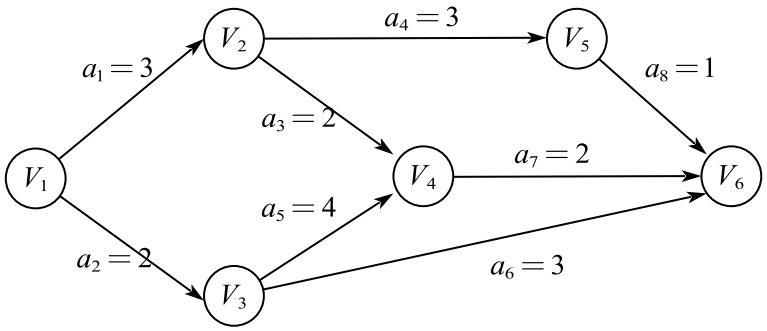
②从汇点出发，令vl(汇点)=ve(汇点)，按逆拓扑有序求其余顶点的最迟发生时间vl()。

③根据各顶点的ve()值求所有弧的最早开始时间e()。

④根据各顶点的vl()值求所有弧的最迟开始时间l()。

⑤求AOE网中所有活动的差额d()，找出所有d()=0的活动构成关键路径。

题 2. 求下图的关键路径。



	v_1	v_2	v_3	v_4	v_5	v_6
$ve(i)$	0	3	2	6	6	8
$vl(i)$	0	4	2	6	7	8

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
$e(i)$	0	0	3	3	2	2	6	6
$l(i)$	1	0	4	4	2	5	6	7
$l(i)-e(i)$	1	0	1	1	0	3	0	1

得到关键路径(v_1, v_3, v_4, v_6)。

注意：

（1）关键路径上的所有活动都是关键活动，因此可以加快关键活动来缩短整个工程的工期。

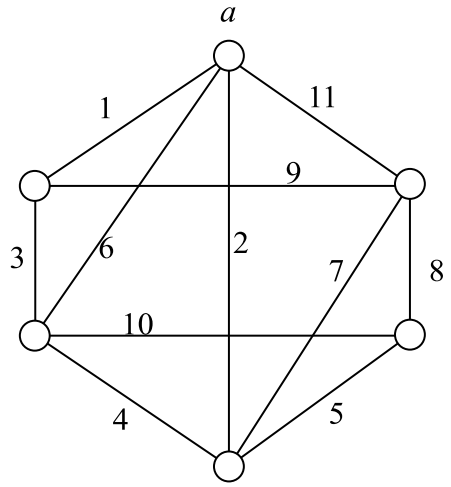
（2）网中的关键路径并不唯一，且对于有几条关键路径的网，只提高一条关键路径上的关键活动并不能缩短整个工程的工期，只有加快那些包括在所有关键路径上的关键活动才能达到缩短工期的目的。

课时九　练习题

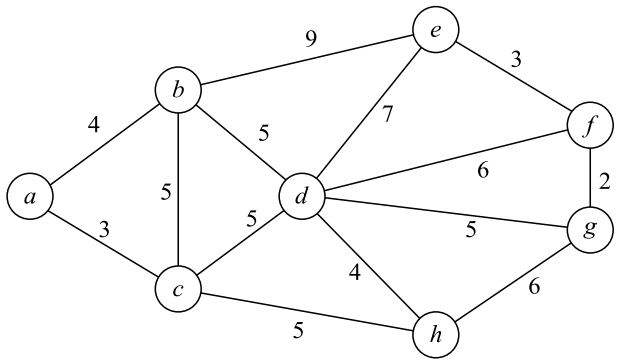
1. 任何一个带权的无向连通图的最小生成树（　　）。

A. 只有一棵　　*B.* 有一棵或多棵　　*C.* 一定有多棵　　*D.* 可能不存在

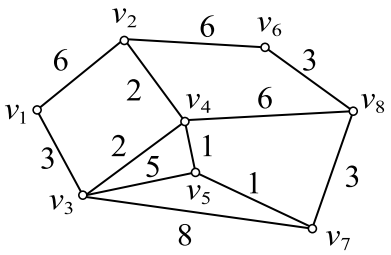
2. 一个赋权网络如下图所示。从顶点a开始，用*Prim*算法求出一棵最小生成树。



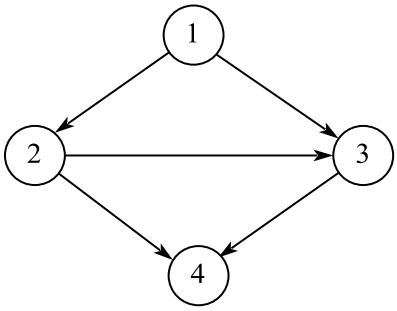
3. 请对下图的无向带权图按克鲁斯卡尔算法求其最小生成树。(要求使用图画出每一步过程)。



4. 用 *dijkstra* 标号法求下图中从 v_1 到其余顶点的最短路径和距离。



5. 已给下图，(　　) 是该图的拓扑排序序列。



A. 1, 2, 3, 4　　*B*. 1, 3, 2, 4　　*C*. 1, 2, 4, 3　　*D*. 1, 4, 2, 3

6. 为判别有向图是否存在回路，可利用 (　　) 算法。

A. 最短路径　　*B*. 深度优先遍历　　*C*. 拓扑排序　　*D*. 最小生成树

7. 下列关于 *AOE* 网的叙述中，不正确的是 (　　)。

A. 关键活动不按期完成就会影响整个工程的完成时间。

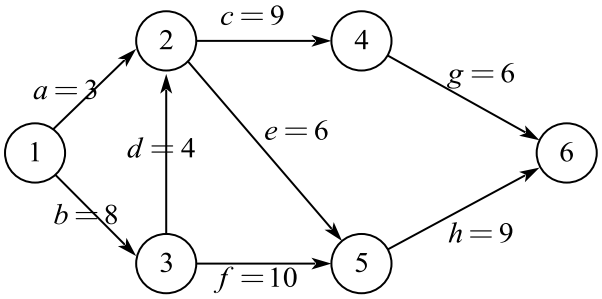
B. 任何一个关键活动提前完成，那么整个工程将会提前完成。

C. 所有的关键活动提前完成，那么整个工程将会提前完成。

D. 某些关键活动提前完成，那么整个工程将会提前完成。

8. 下列 *AOE* 网表示一项包含 8 个活动的工程，通过同时加快若干活动的进度可以缩短整个工程的工期，下列选项中，加快其进度就可以缩短工程工期的是（　　）。

A. c 和 e　　*B.* d 和 e　　*C.* f 和 d　　*D.* f 和 h



9. 已知一个 *AOE* 网采用邻接矩阵存储，其邻接矩阵的三元组表示为 $(1,2,1)$，$(1,3,3)$，$(2,4,4)$,$(2,5,10)$,$(3,5,3)$,$(4,6,6)$,$(5,7,6)$,$(5,8,7)$,$(6,8,5)$,$(7,8,6)$。

（1）画出该 *AOE* 网。

（2）分别求出各事件的最早和最迟发生时间。

（3）分别求出各活动的最早和最迟发生时间。

（4）求出关键活动。

课时十　查找

考点	重要程度	占分	题型
1. 查找	★★	0～2	选择
2. 顺序查找	★★★★★	4～8	选择、填空
3. 折半查找	必考	4～8	选择、填空
4. 散列表	★★★★★	6～10	解答

1. 查找的基本概念

（1）查找。在数据集合中，寻找满足某种条件的数据元素的过程称为查找。

（2）查找表（查找结构）。用于查找的数据集合称为查找表。

（3）关键字是数据元素中某个数据项的值，用它可以标识一个数据元素，若此关键字可以唯一地标识一个记录，用词关键字为主关键字。

（4）平均查找长度是所有查找过程中进行关键字的比较次数的平均值。

2. 顺序查找

顺序查找，主要用于线性表中进行查找。

其基本思想是从线性表的一端开始，逐个检查关键字是否满足给定的条件。若查找到某个元素的关键字满足给定条件，则查找成功，返回该元素在线性表中的位置。若查找到表的另一端，仍未找到符合给定条件的元素，则返回查找失败的信息。

```
typedef struct{
   ElemType *elem;                    //查找表的数据结构
   int TableLen;                      //元素存储空间基址，建表时按实际长度
}SSTable;                             //分配，0 号元素留空
int Search_Seq(SSTable ST,ElemType key){
   ST.elem[0]=key;                    //“哨兵”
   for(i=ST.TableLen;ST.elem[i]!=key;--i);
   return i;
}
```

引入“哨兵”的目的是使得 Search_Seq 内的循环不必判断是否会越界，因为满足 $i==0$，循环一定会跳出。

对于 n 个元素的表，给定值 key 与表中第 i 个元素相等，需进行 $n-i+1$ 次关键字的比较，查找成功时，顺序查找的平均查找长度为 $ASL_{成功}=\sum_{i=1}^{n}p_i(n-i+1)=\frac{n+1}{2}$

查找不成功时，与表中各关键字的比较次数是 NLR 次，从而顺序查找不成功的平均查找长度为 $ASL_{不成功}=n+1$。

注意：对线性链表只能进行顺序查找。

题 1. 采用顺序查找方法查找长度为 n 的顺序表时，它的平均查找长度为（　　）。

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

答案：C

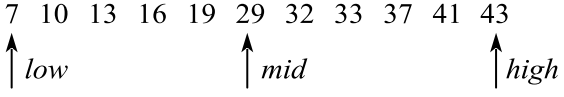
3. 折半查找

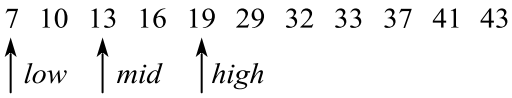
折半查找（二分查找），仅适用于有序的顺序表。

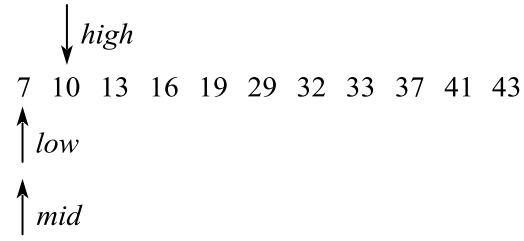
折半查找的基本思想，首先将给定值与表中中间位置的元素比较，若相等，则查找成功，返回该元素的存储位置；若不等，则所需查找的元素只能在中间元素以外的前半部分或后半部分。然后在缩小的范围内继续进行同样的查找，如此反复，直至找到为止，或确定表中没有所需要查找的元素，则查找不成功，返回查找失败的信息。

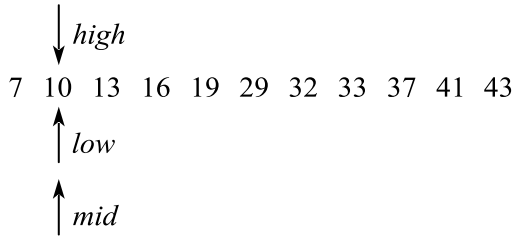
```
int Binary_Search(SeqList L,ElemType key){
    int low=0,high=L.TableLen-1,mid;
    while(low<=high){
        mid=(low+high)/2;
        if(L.elem[mid]==key)
            return mid;
        else if(L.elem[mid]>key)
            high=mid-1;
        else
            low=mid+1;
    }
}
```

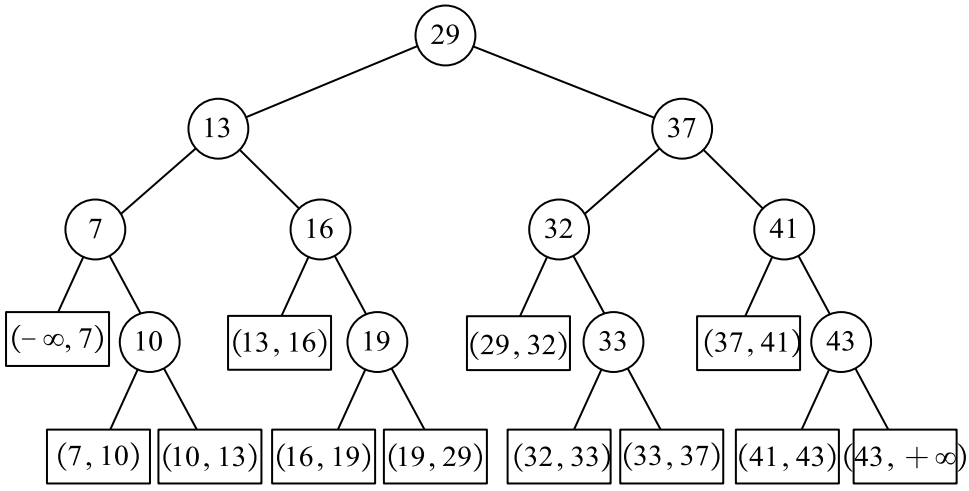
已知11个元素的有序表{7, 10, 13, 16, 19, 29, 32, 33, 37, 41, 43}，查找值为11的元素。











在等概率查找时，查找成功的平均查找长度是

$$ASL=\frac{1}{n}\sum_{i=1}^{n}l_i=\frac{1}{n}(1\times1+2\times2+\cdots+h\times2^{h-1})=\frac{n+1}{n}\log_2(n+1)-1\approx\log_2(n+1)-1$$

其中，h是树的高度，并且元素个数为n时树高$h=\lceil\log_2(n+1)\rceil$。所以，折半查找的时间复杂度为$O(\log_2 n)$。

折半查找法仅适用于顺序存储结构，不适用于链式存储结构，且要求元素按关键字有序排列。

题 1. 二分查找要求结点（　　）。

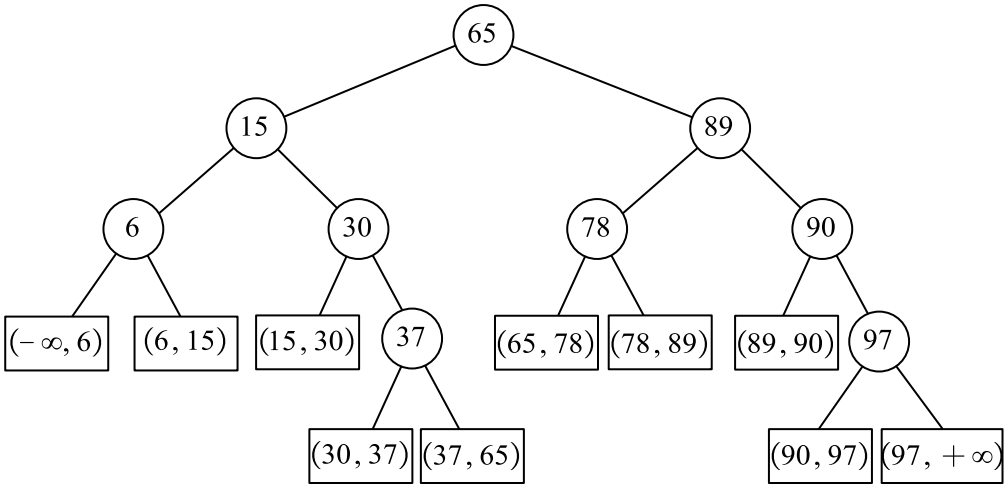
A. 有序、顺序存储　　B. 有序、链式存储

C. 无序、顺序存储　　D. 无序、链式存储

答案：*A*

题 2. 采用对分查找序列数据$(6,15,30,37,65,78,89,90,97)$，若查找元素 89 时比较次数为_____。

答案：2



4. 散列表

散列函数：一个把检查表中的关键字映射成该关键字对应的地址的函数，记为

$Hash(key)=Addr$。

散列函数可能会把两个或两个以上的不同关键字映射到同一地址，称这些情况为冲突，这些发生碰撞的不同关键字称为同义词。

散列表建立了关键字和存储地址之间的一种直接映射关系。

常见的散列函数：

（1）直接定址法：直接取关键字的某个线性函数值为散列地址，散列函数为

$H(key)=key$ 或 $H(key)=a\times key+b$（a,b 是常数）

（2）除留余数法：假定散列表表长为 m 取一个不大于 m 但最接近或等于 m 的质数 p，利用以下公式把关键字转换成散列地址。

$H(key)=key\%p$

（3）数字分析法

（4）平方取中法

解决冲突的方法：

（1）开放定址法：可存放新表项的空闲地址既向它的同义词表项开放，又向它的非同义词表项开放。其数学递推公式为 $H_i=(H(key)+d_i)\%m$（$H(key)$ 为散列函数，m 表示散列表表长，d_i 为增量序列）。

对增量序列通常有以下取法：

① 线性探测法。当 $d_i=0,1,2,\cdots,m-1$ 时，称为线性探测法。这种方法的特点是：冲突发生时，顺序查看表中下一个单元（探测到表尾地址 $m-1$ 时，下一个探测地址就是表首地址 0），直到找出一个空闲单元或查遍全表。

② 平方探测法。当 $d_i=0^2,1^2,-1^2,2^2,-2^2,\cdots,k^2,-k^2$ 时，称为平方探测法，其中 $k\leqslant m/2$。

③ 再散列法。当 $d_i=Hash_2(key)$ 时，称为再散列法。

题 1. 关键字序列为 $\{19,14,23,01,68,20,84,27,55,11,10,79\}$，按散列函数 $H(key)=key\%13$ 和线性探测处理冲突构造散列表，其中散列表长度为 16。

答案：

$H(19)=19\%13=6$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字							19									
比较次数							1									

$H(14)=14\%13=1$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字		14					19									
比较次数		1					1									

$H(23)=23\%13=10$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字		14					19				23					
比较次数		1					1				1					

$H(01)=01\%13=1$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字		14	01				19				23					
比较次数		1	2				1				1					

$H(68)=68\%13=3$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字		14	01	68			19				23					
比较次数		1	2	1			1				1					

$H(20)=20\%13=7$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字		14	01	68			19	20			23					
比较次数		1	2	1			1	1			1					

$H(84)=84\%13=6$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字		14	01	68			19	20	84		23					
比较次数		1	2	1			1	1	3		1					

$H(27)=27\%13=1$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字		14	01	68	27		19	20	84		23					
比较次数		1	2	1	4		1	1	3		1					

$H(55)=55\%13=3$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字		14	01	68	27	55	19	20	84		23					
比较次数		1	2	1	4	3	1	1	3		1					

$H(11)=11\%13=11$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字		14	01	68	27	55	19	20	84		23	11				
比较次数		1	2	1	4	3	1	1	3		1	1				

$H(10)=10\%13=10$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字		14	01	68	27	55	19	20	84		23	11	10			
比较次数		1	2	1	4	3	1	1	3		1	1	3			

$H(79)=79\%13=1$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字		14	01	68	27	55	19	20	84	79	23	11	10			
比较次数		1	2	1	4	3	1	1	3	9	1	1	3			

查找成功的平均查找长度 $ASL_{成功}$=查找次数/元素个数=$(1\times6+2+3\times3+4+9)/12=2.5$

查找不成功的平均查找长度为 $ASL_{不成功}$=查找次数/散列后的地址个数=

$(1+13+12+11+10+9+8+7+6+5+4+3+2)/13=7$

散列表的查找长度取决于三个因素：散列函数、处理冲突的方法和装填因子。

装填因子α＝表中记录数n/散列表长度m。

散列表的平均查找长度依赖于散列表的装填因子α，不直接依赖于n或m。

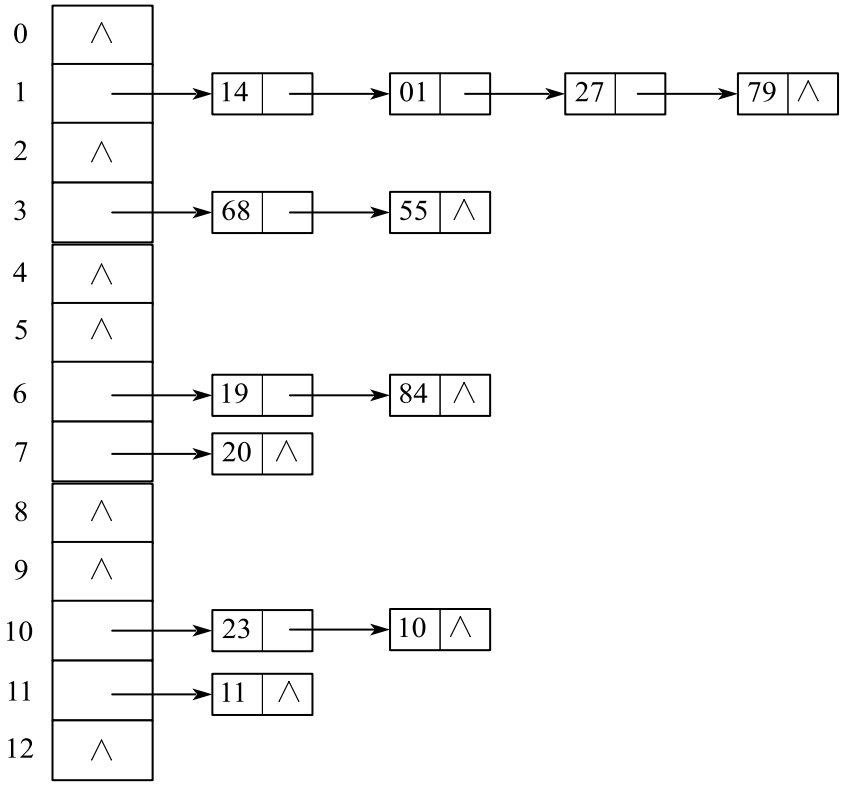
α越大，表示装填的记录越“满”，发生冲突的可能性就越大。

（2）拉链法：为了避免非同义词发生冲突，把所有的同义词存储在一个线性链表中。

题 2. 关键字序列为$\{19,14,23,01,68,20,84,27,55,11,10,79\}$，散列函数为$H(key)=key\%13$，用拉链法解决冲突。

答案：

$H(19)=19\%13=6$
$H(14)=14\%13=1$
$H(23)=23\%13=10$
$H(01)=01\%13=01$
$H(68)=68\%13=3$
$H(20)=20\%13=7$
$H(84)=84\%13=6$
$H(27)=27\%13=1$
$H(55)=55\%13=3$
$H(11)=11\%13=11$
$H(10)=10\%13=10$
$H(79)=79\%13=1$



查找成功的平均查找长度为$(1\times6+2\times4+3+4)/12=1.75$

查找不成功的平均查找长度为$(1+5+1+3+1+1+3+2+1+1+3+2+1)/13=25/13$

课时十　练习题

1. 若在线性表中采用折半查找法查找元素，该线性表应该（　　）。

A. 元素按值有序　　*B.* 采用顺序存储结构

C. 元素按值有序，且采用链式存储结构　　*D.* 元素按值有序，且采用顺序存储结构

2. 折半查找有序表$(4,6,10,12,20,30,50,70,88,100)$。若查找表中元素58，则它将依次与表中（　　）比较大小，查找结果是失败。

A. 20, 70, 30, 50　　*B.* 30, 88, 70, 50

C. 20, 50　　*D.* 30, 88, 50

3. 请完成折半查找算法的相关填空。

```
int Binary_Search(SeqList L,ElemType key){
    int low=0,high=L.TableLen-1,mid;
    while(_____){
        mid=_____;
        if(L.elem[mid]==key)
            return mid;
        else if(L.elem[mid]>key)
            _____;
        else
            _____;
    }
}
```

4. 具有12个关键字的有序表，折半查找的平均查找长度为（　　）。

A. 3.1　　*B*. 4　　*C*. 2.5　　*D*. 5

5. 顺序查找法适合于存储结构为（　　）的线性表。

A.散列存储　　*B*.顺序存储或链式存储

C.压缩存储　　*D*.索引存储

6. 在*HASH*函数$H(key)=key\%p$中，p的值应取（　　）。

A.最接近该*HASH*表长（设为m，下同）　　*B*.奇数

C.小于或等于m的最大质数　　*D*.偶数

7. 已知散列表长度为13，散列函数为$H(key)=key\%11$，处理冲突的方法为线性探测法，请画出插入关键字$(10,8,40,27,21,57,46,23,19,56)$以后的散列表，并计算查找成功和不成功时的平均查找长度。

8. 对给定的关键字序列$19,14,23,1,68,20,84,27,55,11$，给定哈希函数为$H(key)=key\%13$，使用拉链法解决冲突建立哈希表，并给出查找成功的平均查找长度。

课时十一　排序（1）

考点	重要程度	占分	题型
1. 排序的基本概念	★★★	2～4	选择
2. 插入排序	★★★★	6～10	解答
3. 交换排序	★★★★★	6～10	

1. 排序的基本概念

排序，就是重新排列表中的元素，使表中的元素满足按关键字有序的过程。

算法的稳定性。若待排序表中有两个元素 R_i 和 R_j，其对应的关键字相同即 $Key_i = Key_j$，且在排序前 R_i 在 R_j 的前面，若使用某一排序算法排序后，R_i 仍然在 R_j 的前面，则称这个排序算法是稳定的，否则称排序算法是不稳定的。

排序：①内部排序：指在排序期间元素全部存在内存中的排序

②外部排序：是指在排序期间元素无法全部同时存放在内存中，必须在排序的过程中根据要求不断地在内，外存之间移动的排序。

内部排序算法在执行过程中都要进行两种操作：比较和移动。

通常可以将内部排序算法分为插入排序、交换排序、选择排序、归并排序和基数排序五大类。（基数排序不基于比较）

题 1. 内部排序方法的稳定性是指该排序算法不允许有相同的关键字记录。（　　）

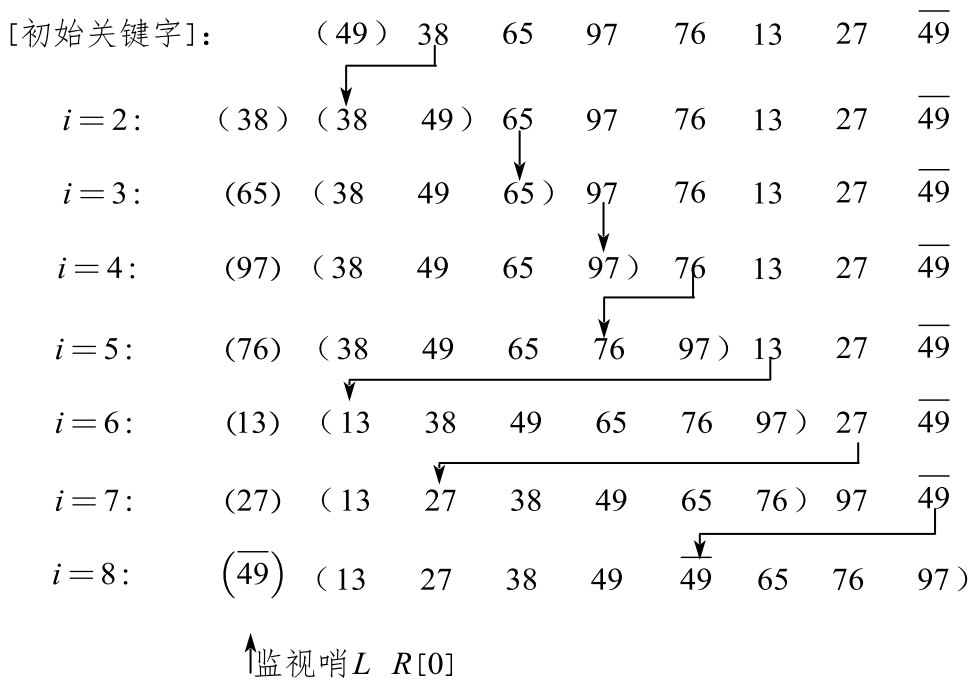
答案：错误

2. 插入排序

基本思想是每次将一个待排序的记录按其关键字大小插入到前面已排好序的子序列中，直到全部记录插入完成。

（1）直接插入排序

假定初始序列为 $49, 38, 65, 97, 76, 13, 27, \overline{49}$，初始时 49 可以视为一个已排好序的子序列，按照上述算法进行直接插入排序的过程如图所示，括号内是已排好序的子序列。



下面是直接插入排序的代码，其中再次用到了我们前面提到的“哨兵”(作用相同)。

```
void InsertSort(ElemType A[],int n){
   int i,j;
   for(i=2;i<=n;i++)                    //依次将A[2]～A[n]插入到前面已排序序列
      if(A[i]<A[i-1]){                  //A[i]关键码小于前驱，将A[i]插入有序表
         A[0]=A[i];                     //复制为哨兵，A[0]不存放元素
         for(j=i-1;A[0]<A[j];--j)       //从后往前查找待插入位置
            A[j+1]=A[j];                //向后挪位
         A[j+1]=A[0];                   //复制到插入位置
      }
}
```

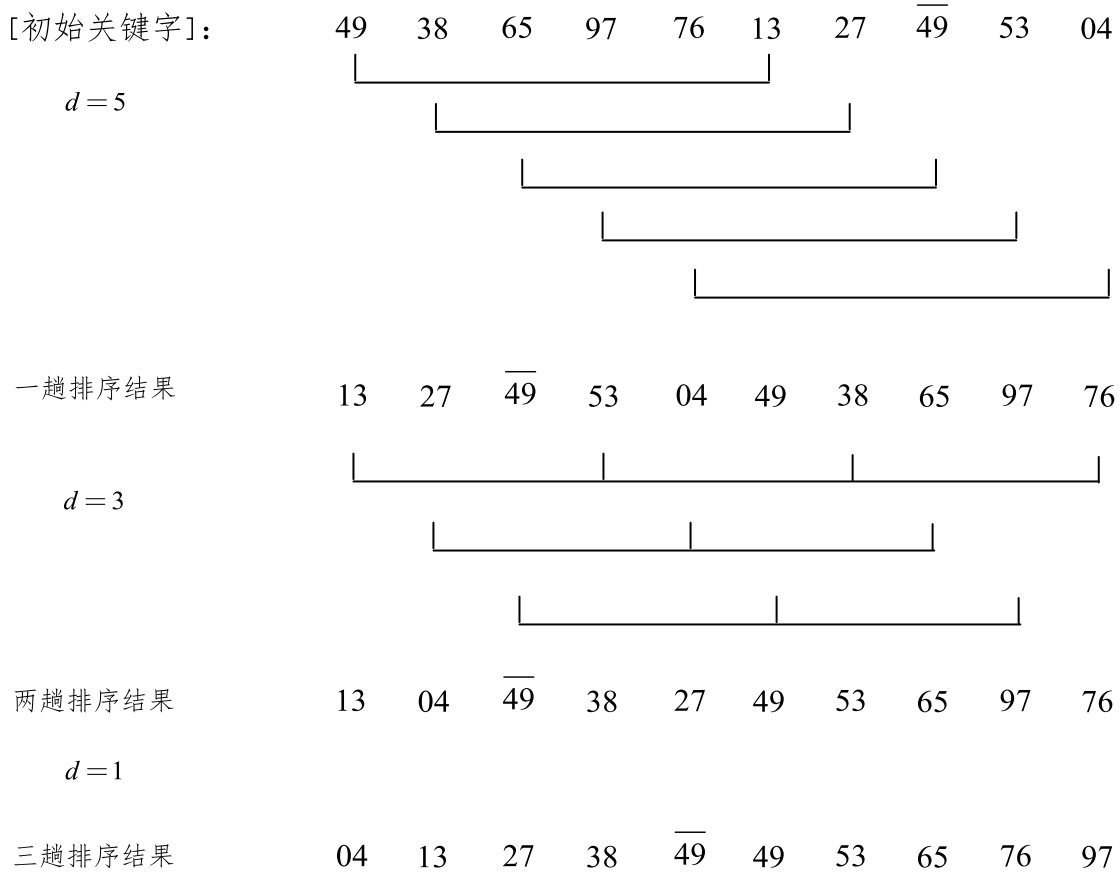
空间效率：仅使用了常数个辅助单元，因而空间复杂度为$O(1)$。

时间效率：在排序过程中，向有序子表中逐个地插入元素的操作进行了$n-1$趟，每趟操作都分为比较关键字和移动元素，而比较次数和移动次数取决于待排序表的初始状态。直接插入排序算法的时间复杂度为$O(n^2)$。

稳定性：由于每次插入元素时总是从后向前先比较再移动，所以不会出现相同元素相对位置发生变化的情况，即直接插入排序是一个稳定的排序方法。

（2）希尔排序

先将待排序表分割成若干形如$L[i,i+d,i+2d,...i+kd]$的特殊子表，即把相隔某个增量的记录组成一个子表，对各个子表分别进行直接插入排序，当整个表中的元素已呈基本有序时，再对全体记录进行一次直接插入排序。



空间效率：仅使用了常数个辅助单元，因而空间复杂度为$O(1)$。

时间效率：由于希尔排序的时间复杂度依赖于增量序列的函数，所以其时间复杂度分析比较困难。

稳定性：不稳定。

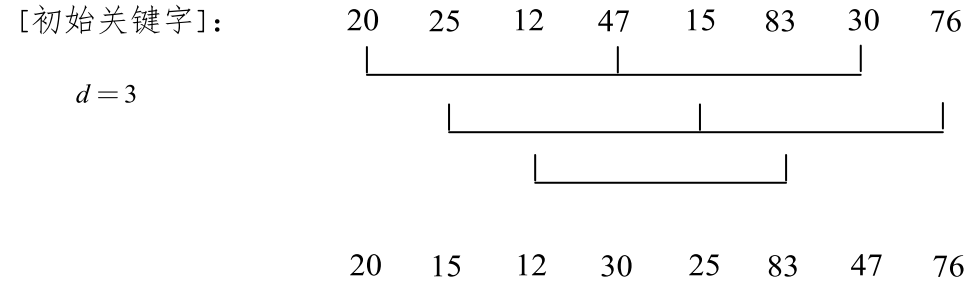
题 1.（　）方法是从未排序的序列中挑选元素，并将其放入已排序序列的一种。

A. 归并排序　　*B*. 插入排序　　*C*. 快速排序　　*D*. 选择排序

答案：*B*

题 2. 用希尔排序法对关键字序列{20, 25, 12, 47, 15, 83, 30, 76}进行排序时，增量为 3 的一趟排序结果是_____。

答案：20 15 12 30 25 83 47 76



3. 交换排序

交换，是指根据序列中两个元素关键字的比较结果来对换这两个记录在序列中的位置。

（1）冒泡排序

基本思想是：从后往前（或者从前往后）两两比较相邻元素的值，若为逆序，则进行交换，直到序列比较完，第一趟冒泡结束，结果是将最小的元素交换到待排序列的第一个位置（或将最大的元素交换到待排序列的最后一个位置）。下一趟冒泡时，前一趟确定的最小元素不再参与比较，每趟冒泡的结果是把序列中的最小元素（或最大元素）放到了序列的最终位置。这样经过 $n-1$ 趟冒泡后就能把所有元素排好序。

初始状态	49	38	65	97	76	13	27	$\overline{49}$
第一趟后	[13]	49	38	65	97	76	27	$\overline{49}$
第二趟后	[13	27]	49	38	65	97	76	$\overline{49}$
第三趟后	[13	27	38]	49	$\overline{49}$	65	97	76
第四趟后	[13	27	38	49]	$\overline{49}$	65	76	97
第五趟后	[13	27	38	49	$\overline{49}$]	65	76	97
第六趟后	[13	27	38	49	$\overline{49}$	65]	76	97
第七趟后	[13	27	38	49	$\overline{49}$	65	76]	97

冒泡排序算法的代码如下：

```
void BubbleSort(ElemType A[],int n){
    for(i=0;i<n-1;i++)
        flag=false;
        for(j=n-1;j>i;j--)
            if(A[j-1]>A[j]){
                swap(A[j-1],A[j]);
                flag=true;
            }
        if(flag==false)
            return;
}
```

空间效率：仅使用了常数个辅助单元，因而空间复杂度为$O(1)$。

时间效率：时间复杂度为$O(n^2)$。

稳定性：稳定。

题 1. 对n个不同的关键字由小到大进行冒泡排序，在下列（　　）情况下交换的次数最多。

A. 从小到大排列好的　　*B*. 从大到小排列好的

C. 元素无序　　*D*. 元素基本有序

答案：*B*

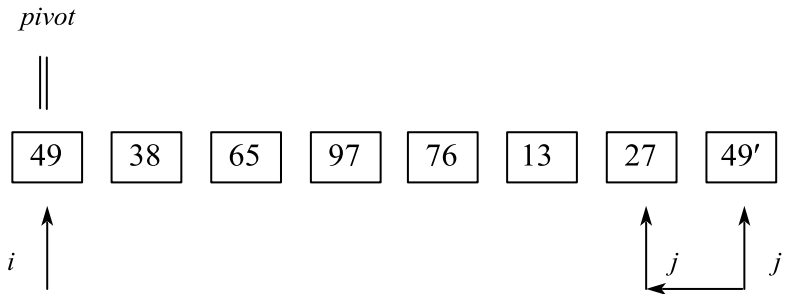
（2）快速排序

快速排序的基本思想是：在待排序表$L[1\cdots n]$中任取一个元素*pivot*作为枢轴（通常取首元素），通过一趟排序将待排序表划分为独立的两部分$L[1\cdots k-1]$和$L[k+1\cdots n]$，使得$L[1\cdots k-1]$中的所有元素小于*pivot*，$L[k+1\cdots n]$中的所有元素大于*pivot*，则*pivot*放在了最终位置$L[k]$上，这个过程称为一趟快速排序。然后分别递归地对两个子表重复上述过程，直至每部分内只有一个元素或空为止。

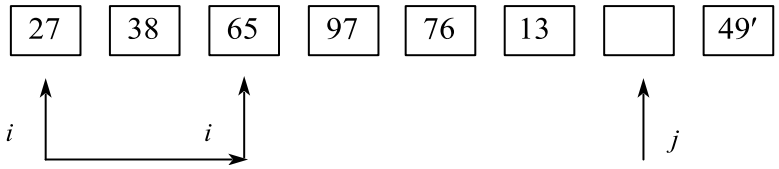
题 1. 对关键字序列$\{49,38,65,97,76,13,27,49'\}$进行快速排序。

设两个指针 i 和 j，初值分别为 low 和 $high$，取第一个元素 49 为数轴赋值到变量 $pivot$。

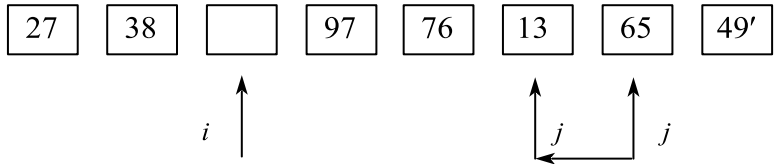
指针 j 从 $high$ 往前搜索找到第一个小于枢轴的元素 27，将 27 交换到 i 所指位置。



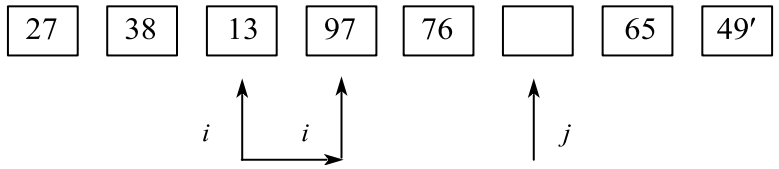
指针 i 从 low 往后搜索找到第一个大于枢轴的元素 65，将 65 交换到 j 所指位置。



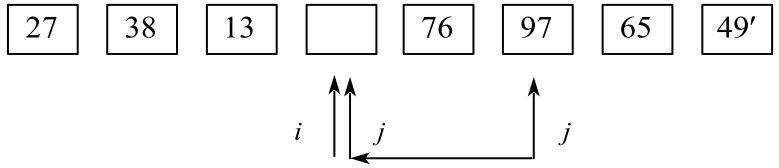
指针 j 继续往前搜索找到小于枢轴的元素 13，将 13 交换到 i 所指位置。



指针 i 继续往后搜索找到大于枢轴的元素 97，将 97 交换到 j 所指位置。



指针 j 继续往前搜索小于枢轴的元素，直到 $i==j$。



此时，经过一次划分，将原序列分割成了前后两个子序列。

{27　38　13}　49　{76　97　65　49′}

按照同样的方法对各子序列进行快速排序，若待排序中只有一个元素，显然已有序。

{27　38　13}　49　{76　97　65　49′}

{13}　27　{38}　　{49′　65}　76　{97}

49′　{65}

最后得到序列　13　27　38　49　49′　65　76　97

空间效率：由于快速排序是递归的，需要借助栈来保存每层递归调用的必要信息，其容量要与递归调用的最大深度一致，因而空间复杂度为$O(\log_2 n)$。

时间效率：时间复杂度为$O(n\log_2 n)$。

稳定性：不稳定。

快速排序是所有内部排序算法中平均性能最优的排序算法。

快速排序并不适用于原本有序或基本有序的记录序列进行排序。

课时十一　练习题

1. 在待排序的记录集中，存在多个具有相同键值的记录，这些记录的相对次序仍然保持不变，称这种排序是稳定排序。（　　）

2. 若数据元素序列$\{11, 12, 13, 7, 8, 9, 23, 4, 5\}$，是采用下列排序方法之一得到的第二趟排序后的结果，则该排序算法只能是（　　）。

A. 冒泡排序　　*B.* 直接插入排序

C. 简单选择排序　　*D.* 二路归并排序

3. 设一组初始记录关键字序列为$\{49, 38, 65, 97, 76, 13, 27, 50\}$，则以$d=4$为增量的一趟希尔排序结束后的结果为_____。

4. 在对一组记录$\{18, 6, 27, 12, 52, 15, 47, 29\}$进行直接插入排序时，当把第6个记录15插入到有序表时，为寻找插入位置需比较_____次。

5. 直接插入排序算法在平均情况下的时间复杂度为_____，在最好情况下的时间复杂度为_____。

6. 用直接插入排序对下面四个序列进行递增排序，元素比较次数最少的是（　　）。

A. 94, 32, 40, 90, 80, 46, 21, 69　　*B*. 32, 40, 21, 46, 69, 94, 90, 80

C. 21, 32, 46, 40, 80, 69, 90, 94　　*D*. 90, 69, 80, 46, 21, 32, 94, 40

7. 从未排序序列中选择一个元素，该元素将未排序序列分成前后两个部分，前一部分中所有元素都小于等于所选元素；后一部分中所有元素都大于等于所选元素，而所选元素处在排序的最终位置，这种排序方法称作（　　）。

A. 插入排序　　*B*. 希尔排序　　*C*. 快速排序　　*D*. 堆排序

8. 已知待排序记录的关键字序列{77, 71, 52, 22, 15, 30, 3}，用冒泡排序法按从小到大顺序写出每趟排序的结果，直到排序结束。

9. 用某种排序方法对关键字序列{25, 84, 21, 47, 15, 27, 68, 35, 20}进行排序时，序列的变化情况如下：

20, 15, 21, 25, 47, 27, 68, 35, 84
15, 20, 21, 25, 35, 27, 47, 68, 84
15, 20, 21, 25, 27, 35, 47, 68, 84

则所采用的排序方法是（　　）。

A. 选择排序　　*B*. 希尔排序　　*C*. 归并排序　　*D*. 快速排序

10. 给出一组关键字{29, 18, 25, 47, 58, 12, 51, 10}，写出快速排序方法（第1个元素为枢轴）进行排序的每一趟排序。

课时十二　排序（2）

考点	重要程度	占分	题型
1. 选择排序	★★★★★	6～10	选择、填空、解答
2. 归并排序	★★★★	4～6	
3. 基数排序	★★★★	4～6	
4. 各种排序算法的比较	★★★★	2～4	选择

1. 选择排序

选择排序的基本思想：

每一趟（第i趟）在后面$n-i+1(i=1,2,\cdots,n-1)$个待排序元素中选取关键字最小的元素，作为有序子序列的第i个元素，直到$n-1$趟做完，待排序元素只剩下一个，就不用再选了。

（1）简单选择排序

基本思想：假设排序表为$L[1\cdots n]$，第i趟排序即从$L[i\cdots n]$中选择关键字最小的元素与$L[i]$交换，每一趟排序可以确定一个元素的最终位置，这样经过$n-1$趟排序就可使得整个排序表有序。

简单选择排序算法的代码如下：

```
void SelectSort(ElemType A[],int n){
    for(i=0;i<n-1;i++){
        min=i;
        for(j=i+1;j<n;j++)
            if(A[j]<A[min])
                min=j;
        if(min!=i)
            swap(A[i],A[min]);
    }
}
```

空间效率：$O(1)$

时间效率：$O(n^2)$

稳定性：不稳定。

例如，表$L=\{2,2',1\}$，经过一趟排序后$L=\{1,2',2\}$，最终排序序列也是$L=\{1,2',2\}$

题 1. 给定关键字序列{87, 45, 78, 32, 17, 65, 53, 09}，用简单选择排序算法进行排序。

答案：

	87	45	78	32	17	65	53	09
第一趟排序	[09]	45	78	32	17	65	53	87
第二趟排序	[09	17]	78	32	45	65	53	87
第三趟排序	[09	17	32]	78	45	65	53	87
第四趟排序	[09	17	32	45]	78	65	53	87
第五趟排序	[09	17	32	45	53]	65	78	87
第六趟排序	[09	17	32	45	53	65]	78	87
第七趟排序	[09	17	32	45	53	65	78]	87

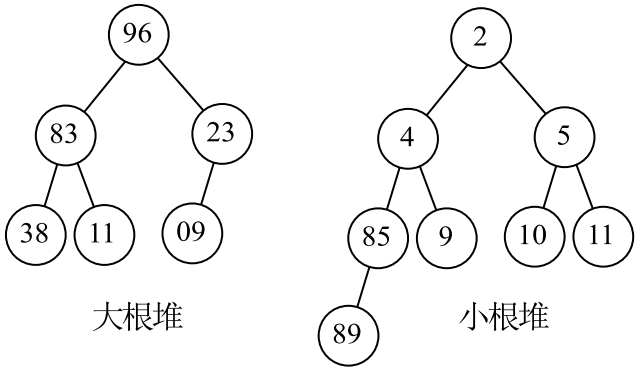
（2）堆排序

堆的定义如下，n 个关键字序列 $L[1\cdots n]$ 称为堆，当且仅当该序列满足：

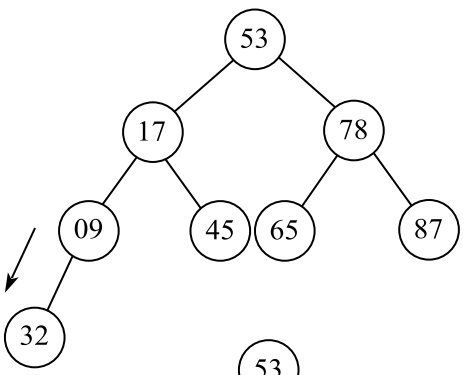
①$L[i]>=L[2i]$ 且 $L[i]>=L[2i+1]$

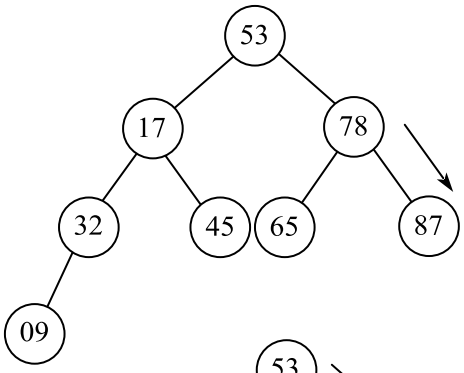
②$L[i]<=L[2i]$ 且 $L[i]<=L[2i+1]$ $1\leqslant i\leqslant\lfloor n/2\rfloor$

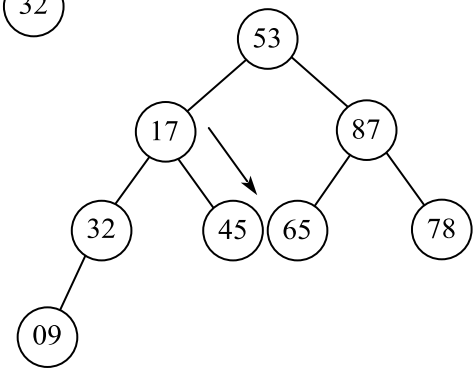
满足条件①的堆称为大根堆，大根堆的最大元素存放在根结点，且其任一非根结点的值小于等于其双亲结点值。满足条件②的堆称为小根堆。

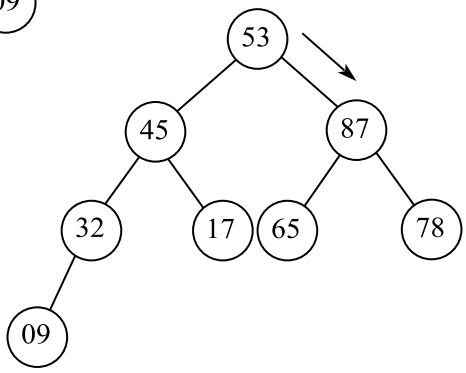


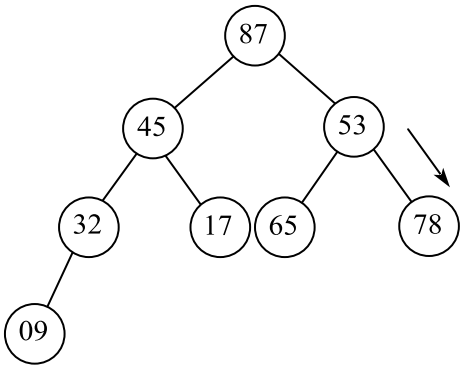
堆排序的关键是构造初始堆。n 个结点的完全二叉树，最后一个结点是第 $\lfloor n/2\rfloor$ 个结点的孩子。对第 $\lfloor n/2\rfloor$ 个结点为根的子树进行筛选，使该子树成为堆。之后向前依次对各结点为根的子树进行筛选，看该结点值是否大于其左右子结点的值，若不大于，则将左右子结点中的较大值与之交换，交换后可能破坏下一级的堆，于是继续采用上述方法构造下一级的堆，直到以该结点为根的子树构成堆为止。反复利用上述调整堆的方法建堆，直到根结点。

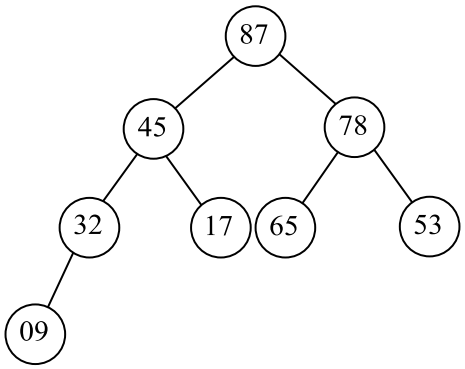










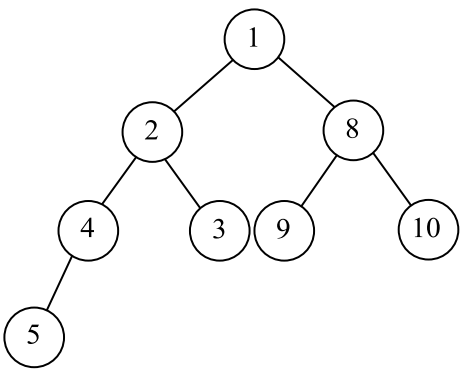


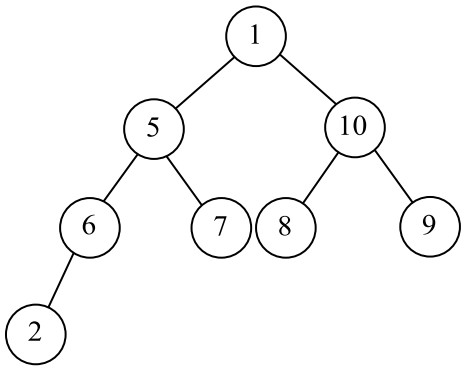
题 1. 下面的序列中，（　　）是堆。

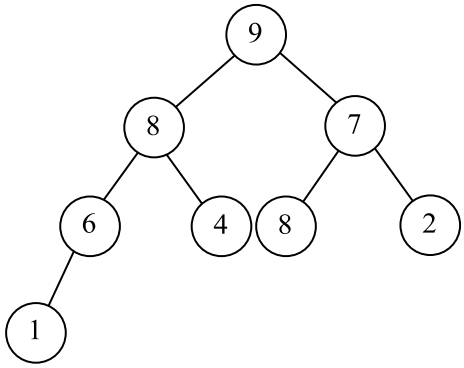
A. 1, 2, 8, 4, 3, 9, 10, 5　　*B*. 1, 5, 10, 6, 7, 8, 9, 2

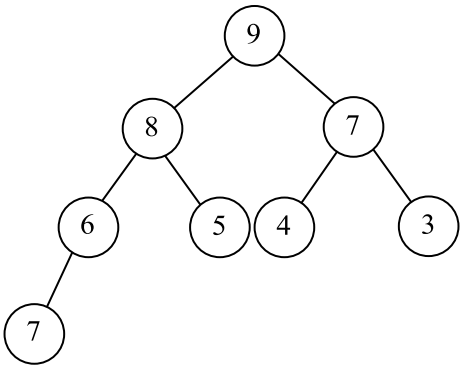
C. 9, 8, 7, 6, 4, 8, 2, 1　　*D*. 9, 8, 7, 6, 5, 4, 3, 7

答案：*A*

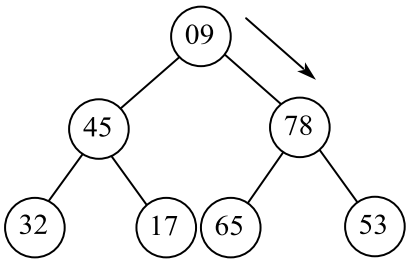


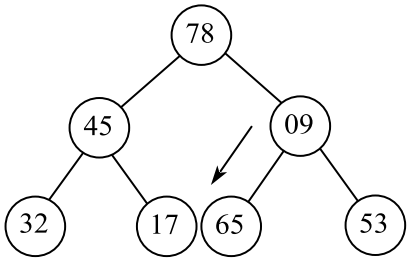


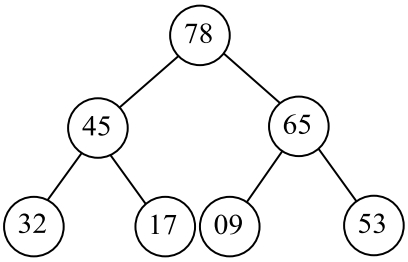




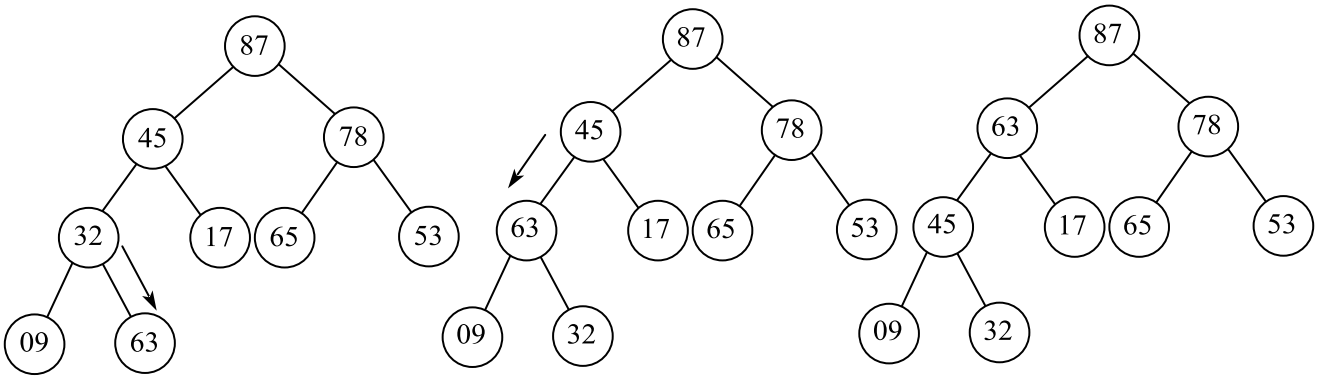
输出栈顶元素后，将堆的最后一个元素与栈顶元素交换，此时堆的性质被破坏。







同时，堆也支持插入操作。对堆进行插入操作时，先将新结点放在堆的末端，再对这个新结点向上执行调整操作。



空间效率：$O(1)$

时间效率：$O(n\log_2 n)$

稳定性：不稳定。

例如，表$L=\{1,2,2'\}$，构造初始堆时可能将2交换到堆顶，最终排序序列也是$L=\{2',2,1\}$。

题 2. 设有5000 个待排序的记录关键字，如果需要用最快的方法选出其中最小的10 个记录关键字，则用下列（　　）方法可以达到目的。

A. 快速排序　　*B*. 堆排序

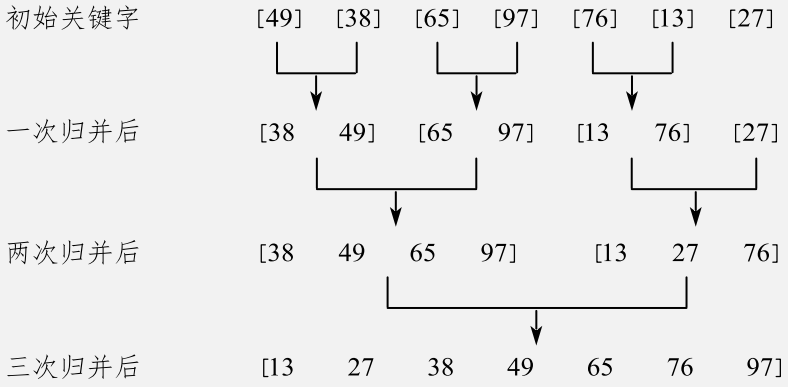
C. 归并排序　　*D*. 直接插入排序

答案：*B*

2. 归并排序

“归并”的含义是将两个或两个以上的有序表组合成一个新的有序表。

假定待排序表含有n个记录，则可将其视为n个有序的子表，每个子表的长度为1，然后两两归并，得到$\lceil n/2 \rceil$个长度为1或2的有序表；继续两两归并……如此反复，直到合并成一个长度为n的有序表为止，这种排序方法称为2路归并排序。



空间效率：$O(n)$

时间效率：$O(n\log_2 n)$

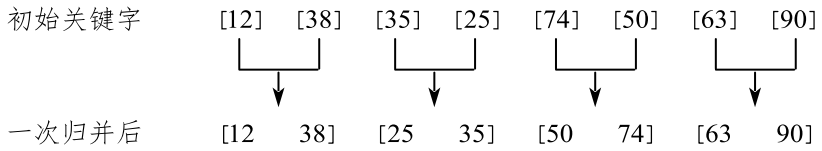
稳定性：稳定。

题 1. 一组记录值为$(12,38,35,25,74,50,63,90)$，按2路归并排序方法对序列进行一趟归并后的结果是（　　）。

A. $12,38,25,35,50,74,63,90$　　*B*. $12,38,35,25,74,50,63,90$

C. $12,25,35,38,50,74,63,90$　　*D*. $12,35,38,25,63,50,74,90$

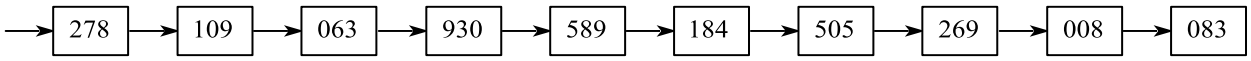
答案：*A*



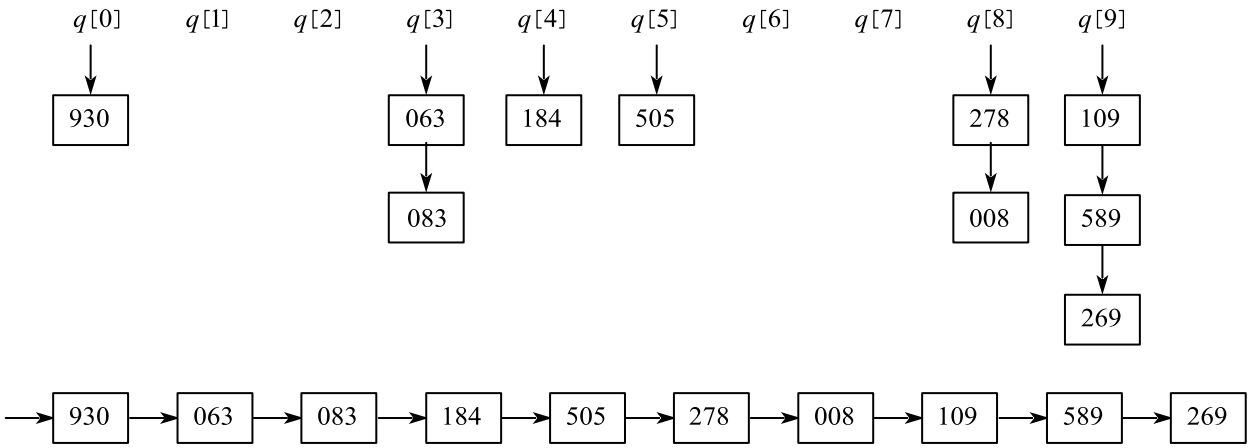
3. 基数排序

基数排序不基于比较和移动进行排序，而基于关键字各位的大小排序。

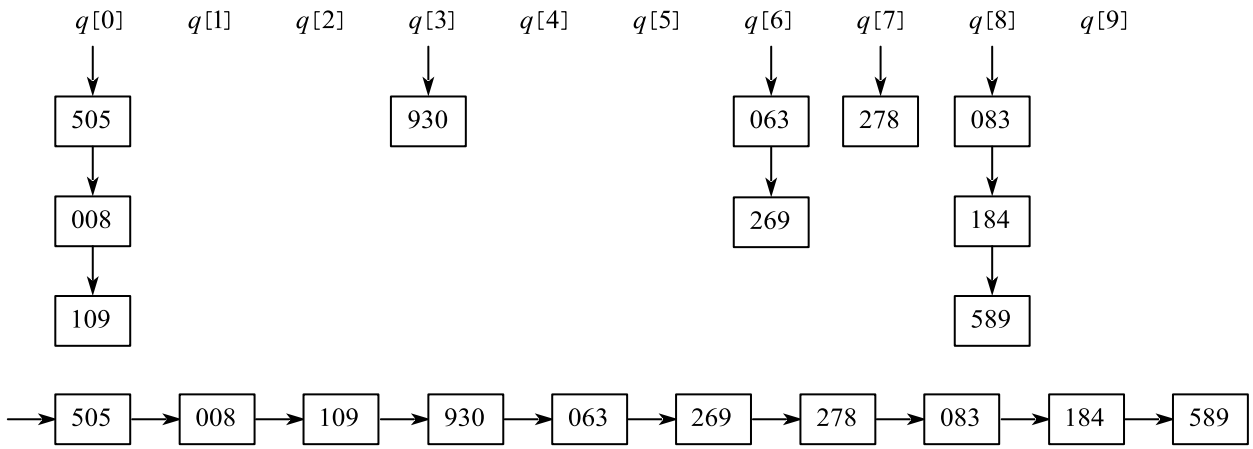
通常采用链式基数排序，假设对如下10 个记录：



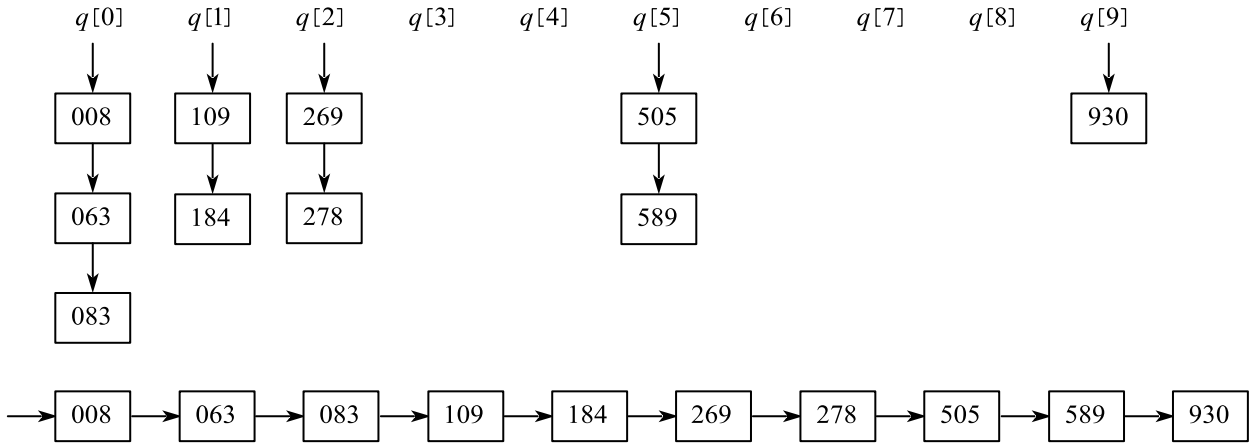
第一趟基于个位：



第二趟基于十位：



第三趟基于百位：



空间效率：$O(r)$

时间效率：$O(d(n+r))$，它与序列的初始状态无关。

稳定性：稳定。

题 1. 如果将所有中国人的按照生日（不考虑年份，只考虑月、日）来排序，那么使用下列排序算法中的（　　）算法最快。

A. 归并排序　　*B*. 希尔排序

C. 快速排序　　*D*. 基数排序

答案：*D*

4. 各种排序算法的比较

由于希尔排序的时间复杂度依赖于增量函数，所以无法准确给出其时间复杂度。

算法种类	时间复杂度			空间复杂度	是否稳定
	最好情况	平均情况	最坏情况		
直接插入排序	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	是
希尔排序				$O(1)$	否
冒泡排序	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	是
快速排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n^2)$	$O(\log_2 n)$	否
简单选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	否
堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(1)$	否
2 路归并排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n)$	是
基数排序	$O(d(n+r))$	$O(d(n+r))$	$O(d(n+r))$	$O(r)$	是

题 1. 在直接插入排序和快速排序中，若初始数据基本有序，则选用_____；在冒泡排序和堆排序中，若要求数据的稳定性，则选用_____。

答案：直接插入排序；冒泡排序

题 2. 以下四种排序方法中，需要附加的内存空间最大的是（　　）。

A. 插入排序　　*B*. 选择排序　　*C*. 快速排序　　*D*. 归并排序

答案：*D*

题 3. 一趟排序结束后不一定能够选出一个元素放在其在最终位置上的是（　　）。

A. 简单选择排序　　*B*. 冒泡排序　　*C*. 快速排序　　*D*. 希尔排序

答案：*D*

课时十二　练习题

1. 对具有 n 个元素的任意序列采用选择排序，排序趟数为（　　）。

A. $n-1$　　*B*. n　　*C*. $n+1$　　*D*. $\lfloor \log_2 n \rfloor$

2. 堆是完全二叉树，完全二叉树不一定是堆。（　　）

3. 以下序列不是堆（大根堆或小根堆）的是（　　）。

A. $\{100, 85, 98, 77, 80, 60, 82, 40, 20, 10, 66\}$

B. $\{100, 98, 85, 82, 80, 77, 66, 40, 20, 10\}$

C. $\{10, 20, 40, 60, 66, 77, 80, 82, 85, 98, 100\}$

D. $\{100, 85, 40, 77, 80, 60, 66, 98, 82, 10, 20\}$

4. 已知关键字序列$\{5, 8, 12, 19, 28, 20, 15, 22\}$是小根堆，插入关键字3，调整后得到的小根堆是（　　）。

A. $3, 5, 12, 8, 28, 20, 15, 22, 19$　　*B*. $3, 5, 12, 19, 20, 15, 22, 8, 28$

C. $3, 8, 12, 5, 20, 15, 22, 28, 19$　　*D*. $3, 12, 5, 8, 28, 20, 15, 22, 19$

5. 已知序列$\{21,41,7,43,58,63,29,8\}$，判断该序列是否为堆。若不是，将其调整为小根堆，画出调整完的小根堆。

6. 将初始关键字序列$\{51,12,55,23,49,07,60,36,72,12'\}$，写出归并排序算法的每一趟结果。

7. 给出关键字序列$\{112,204,312,902,156,712,451,623,643,834\}$进行基数排序时每一趟的结果。

8. 在最好和最坏情况下的时间复杂度均为$O(n\log_2 n)$且稳定的排序方法是（　　）。

A. 快速排序　　*B*. 堆排序　　*C*. 归并排序　　*D*. 基数排序

9. 在直接插入排序、希尔排序、冒泡排序和快速排序中，平均情况下_____最快，其时间复杂度为_____。

10. 下列排序算法中，第一趟排序结束后，其最大或最小元素一定在其最终位置上的算法是（　　）。

A. 归并排序　　*B*. 直接插入排序　　*C*. 快速排序　　*D*. 冒泡排序