高数帮

数据结构

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	★★★		

视频讲解更清晰
仅4小时

1.1 数据结构

（1）数据是描述客观事物的数值、字符以及能输入机器且能被处理的各种符号集合。

（2）数据元素是组成数据的基本单位，是数据集合的个体。

（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）数据对象是性质相同的数据元素的集合，是数据的一个子集。例如，整数数据对象是集合N = {O, 士1,士2,.....}。

（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。

1.1 数据结构

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

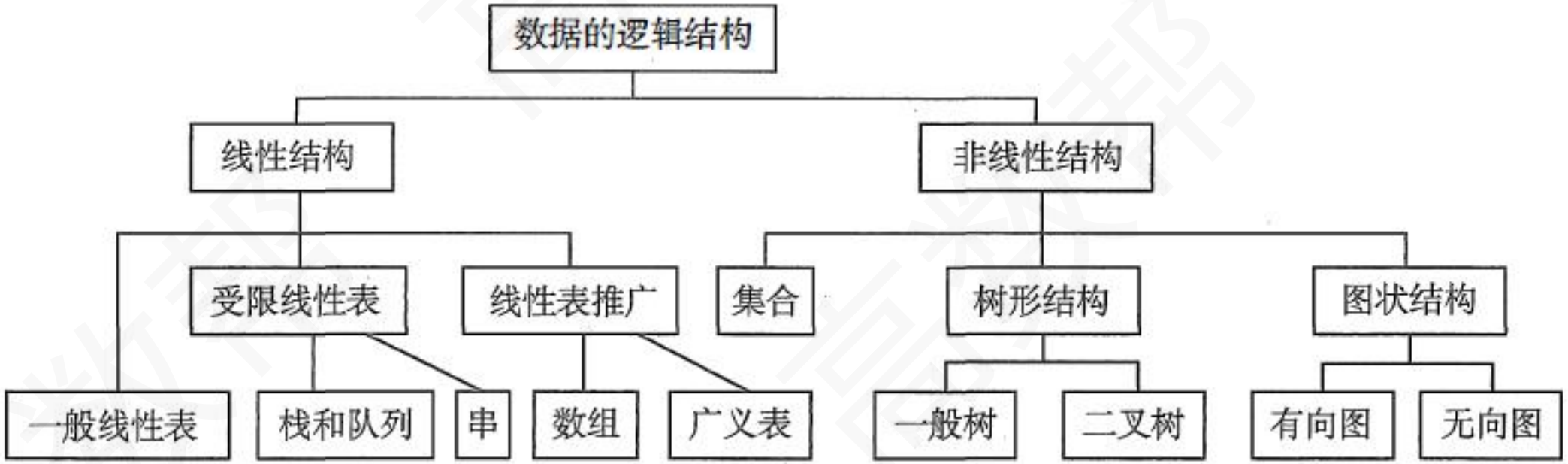
题2.数据的逻辑结构从形式上可用二元组（D，R）表示，其中R是（ ）的有限集。

A.算法　　B.数据元素　　C.数据项　　D.数据关系

答案 D

（1）数据的逻辑结构

逻辑结构是指数据元素之间逻辑关系描述。数据的逻辑结构分为线性结构和非线性结构，线性表是典型的线性结构；集合、树和图是典型的非线性结构。数据的逻辑结构分类如图1.1所示。



集合结构。结构中的数据元素之间除“同属一个集合”外，别无其他关系，如图1.2(a)所示。

线性结构。结构中的数据元素之间只存在一对一的关系，如图1.2(b)所示。

树形结构。结构中的数据元素之间存在一对多的关系，如图1.2(C)所示。

图状结构或网状结构。结构中的数据元素之间存在多对多的关系，如图1.2(d)所示。

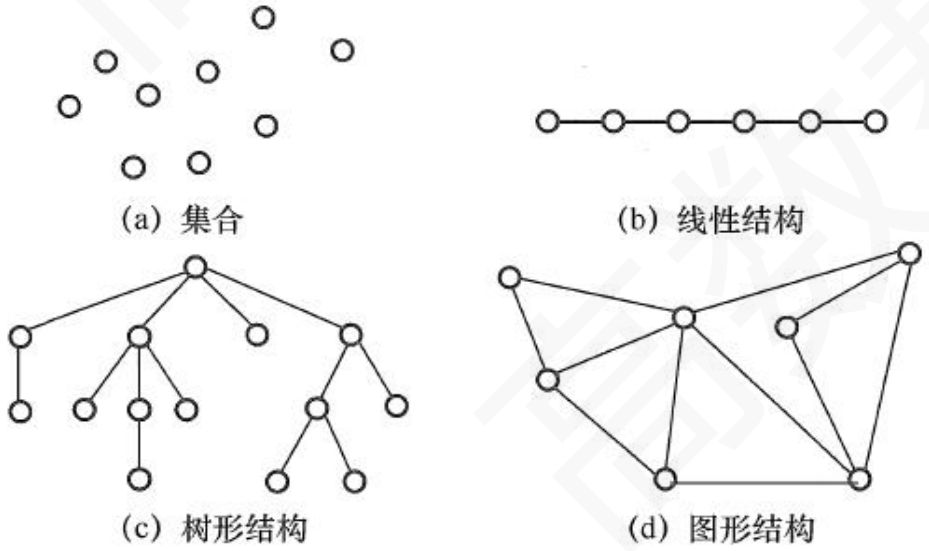


图 1.2　4 类基本结构关系示例图

（2）数据的存储结构

存储结构(又称物理结构)是逻辑结构在计算机中的存储影响。它包括数据元素映像和关系映像。数据的存储结构主要有顺序存储、非顺序存储（包括链式存储、索引存储和散列存储）。

1) 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。其优点是可以实现随机存取，每个元素占用最少的存储空间；缺点是只能使用相邻的一整块存储单元，因此可能产生较多的外部碎片。

2) 链式存储。不要求逻辑上相邻的元素在物理位置上也相邻， 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。其优点是不会出现碎片现象，能充分利用所有存储单元；缺点是每个元素因存储指针而占用额外的存储空间，且只能实现顺序存取。

3) 索引存储。在存储元素信息的同时， 还建立附加的索引表。索引表中的每项称为索引项，索引项的一般形式是（关键字，地址）。其优点是检索速度快；缺点是附加的索引表额外占用存储空间。另外，增加和删除数据时也要修改索引表，因而会花费较多的时间。

4) 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。其优点是检索、增加和删除结点的操作都很快；缺点是若散列函数不好，则可能出现元素存储单元的冲突，而解决冲突会增加时间和空间开销。

1.1 数据结构

题1.以下数据结构中，（）是非线性数据结构

A.树　　B.字符串　　C.队列　　D.栈　　答案A

题2.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　　答案C

题2.以下关于数据结构的说法中，正确的是（ ）

A.数据的逻辑结构独立于其存储结构

B.数据的存储结构独立于其逻辑结构

C.数据的逻辑结构唯一决定了其存储结构

D.数据结构仅由其逻辑结构和存储结构决定　　答案A

1.2 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 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。

1.2 算法

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。

1.2 算法

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.若一个算法的语句频度之和为T(n)=n+4nlog2n,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

2.1 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

2.1 线性表的类型定义

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 的长度，即 中数据元素的个数。

(3)LocateElem(L,e) : 按值查找操作。获取表 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 中第 个位置的元素的值。

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

(6)ListDelete(&L,i,&e) : 删除操作。删除表 中第 个位置的元素，并用 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 的元素值。

(8)Empty(L) : 判空操作。

(9)DestroyList(&L) : 销毁操作。

2.2 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	a1	LOC(A)
1	a2	LOC(A)+sizeof(ElemType)
	``	
i-1	a4	LOC(A)+sizeof(ElemType)*(i-1)

2.2 顺序表的结构

题1.顺序表中，逻辑上相邻的元素，其物理位置 _____相邻。

答案：一定

题2.如果一个顺序表中第一个元素存储地址为2001，每个元素占用4个地址单元，那么第6个元素的存储地址是

答案：2001+（6-1）*4=2021

2.2 顺序表的结构

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{                                     //顺序表的元素
ElemType data[MaxSize]; 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。

2.2 顺序表的结构

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                    //定义线性表的最大长度
typedef
struct{ ElemTyp                        //顺序表的元素
e *data;                               //顺序表的当前长度
int MaxSize,length;                    //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

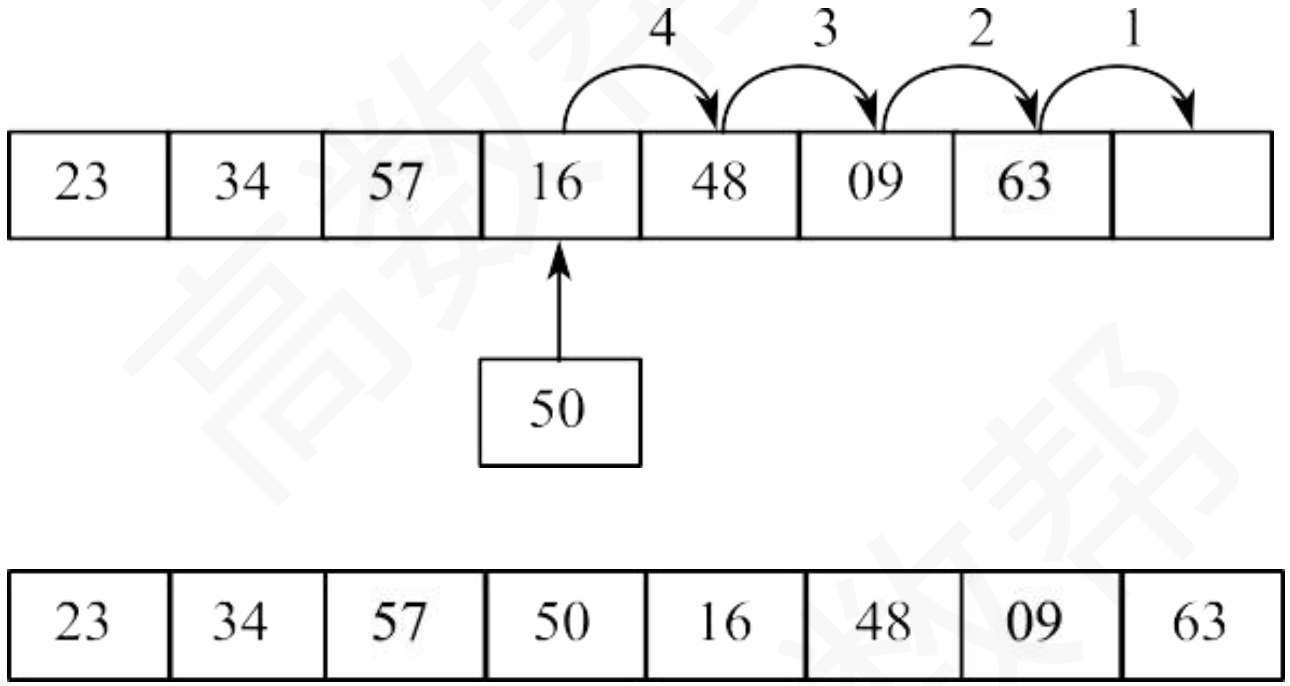
顺序表的存储密度高，每个结点只存储数据元素。

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.在顺序表中插入或删除一个元素，需要平均移动_____元素，具体移动元素个数与_____有关

答案：一半，插入或删除的位置

(1)插入操作



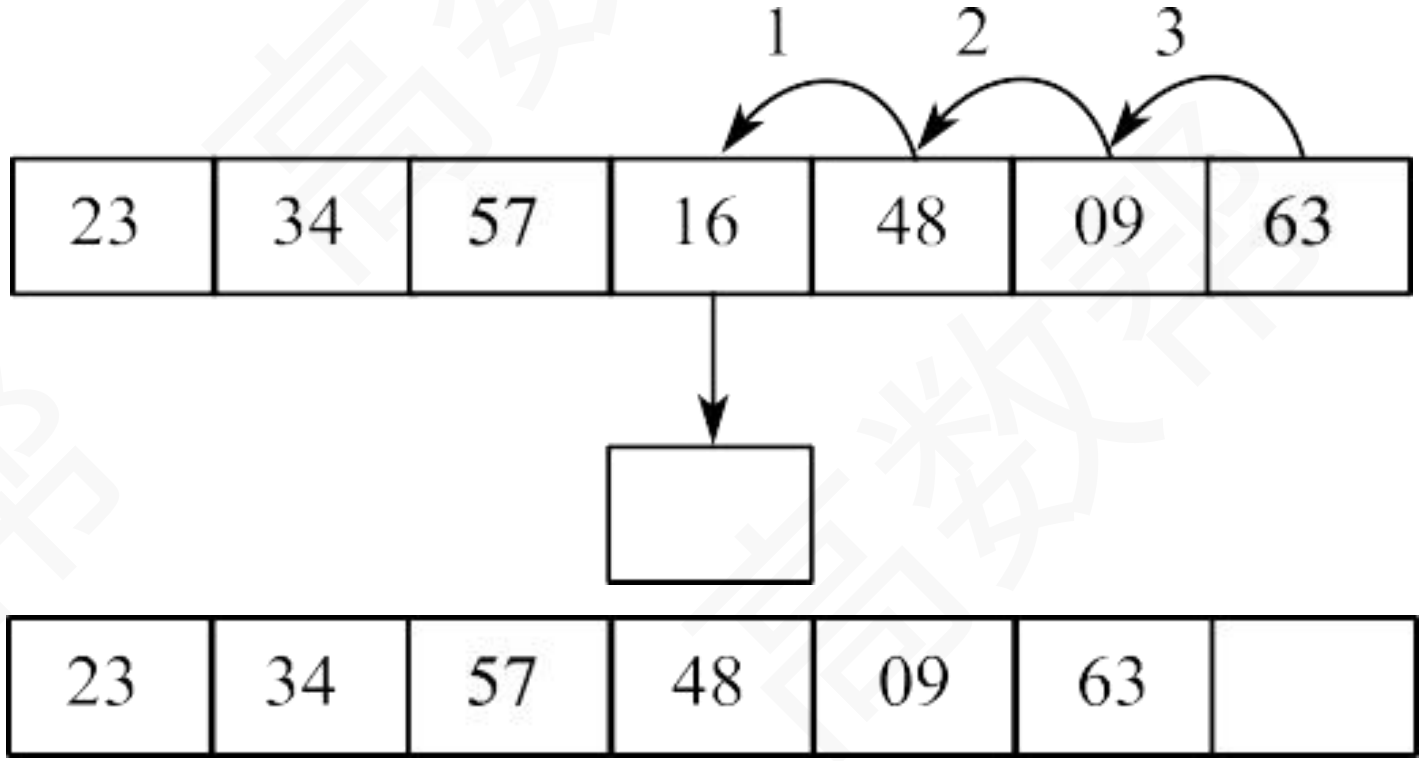
在顺序表L的第i(1≤i≤L.length+1)个位置插入新元素 e。若i的输入不合法，则返回false ，表示插入失败；否则，将顺序表的第i个元素及其后的所有元素右移一个位置，腾 出一个空位置插入新元素e，顺序表长度增加1，插入成功，返回 true。

```
 bool ListInsert(SqList &L,int i,ElemType e){
if(i<1||i>L.length+1)                    //判断 的范围是否有效
return false; for(int j=L.length;j>=i;j--) //将第 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 处放入
 L.length++;                             //线性表长度增加
return true;
}
```

题1.在长度为 的顺序表的第i个位置上插入一个元素(1≤i≤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≤i≤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false，否则返回 。

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

删除线性表L中第i(1≤i≤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false，否则返回 。

```
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 的元素值等于 ， 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 的范围是否有效
return 0;
return L.data[i-1];
}
```

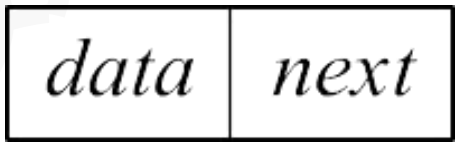
课时三 链表

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	

3.1 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。



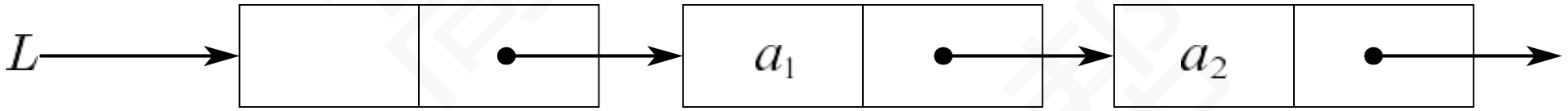
其中data为数据域，存放数据元素；next为指针域，存放其后继结点的地址。

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;
```

3.1 单链表的结构

通常用头指针来标识一个单链表L，如单链表，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.在单链表中，逻辑上相邻的元素怒，其物理位置___相邻

答案：不一定

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）。

(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
 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 个结点的指针，若 大于表长则返回
}
```

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
p=p->next;
return p;                                    //找到后返回该结点指针，否则返回
}
```

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e的结点时，在查找成功的情况下，平均需比较（　　）个元素结点。

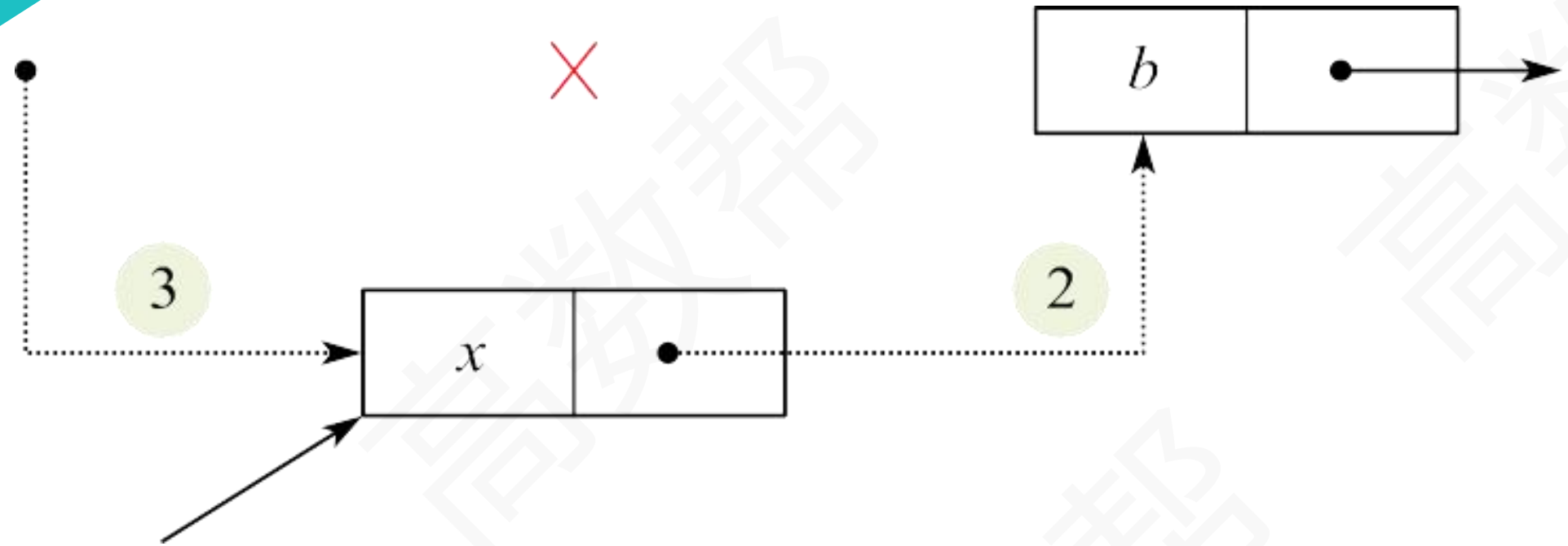
A.n/2　　B.n　　C.(n+1)/2　　D.(n-1)/2　　答案：C

（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。

3.2 单链表的实现



实现插入结点的代码如下：

①p=GetElem(L,i-1)　　//查找插入位置的前驱结点

②s->next=p->next

③p->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-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

题 3.已知单链表的头指针变量为head且该链表不带头结点，则该单链表的判空条件是（）

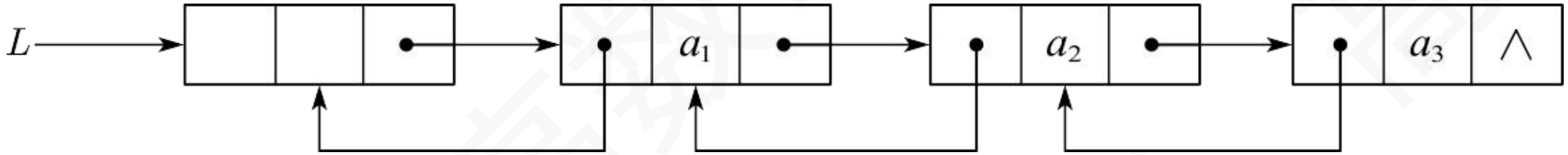
A.head==NULL　　B.head->next==NULL

C.head->next==head　　D.head!=NULL

答案：A

（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;
```

视频讲解更清晰
仅4小时

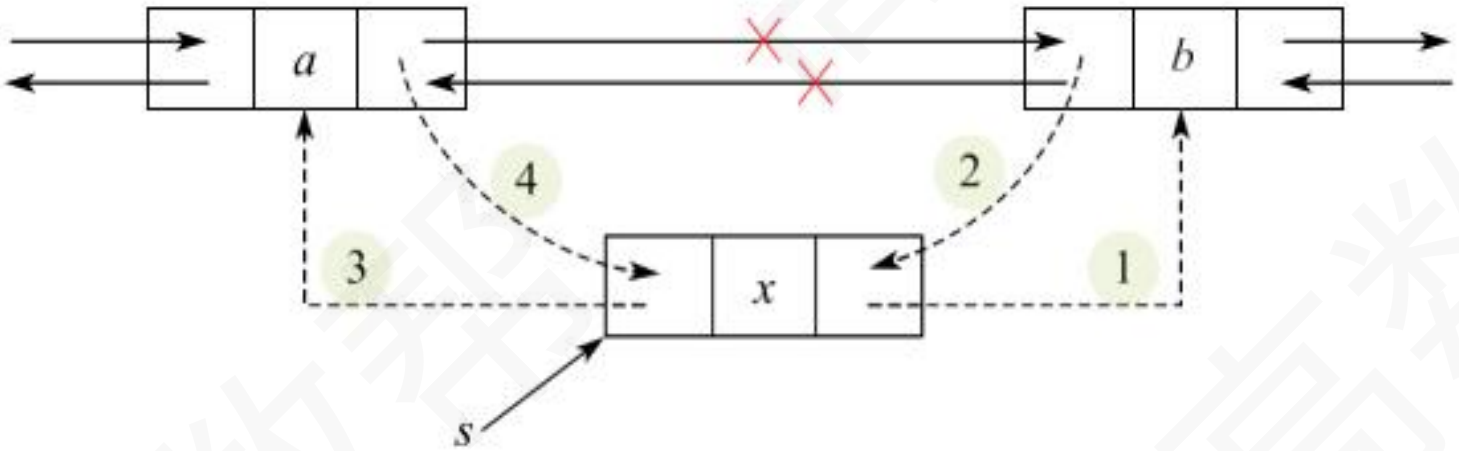
3.3 双链表、循环链表

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：

①s->next-p->next;

②p->next->prior=p;

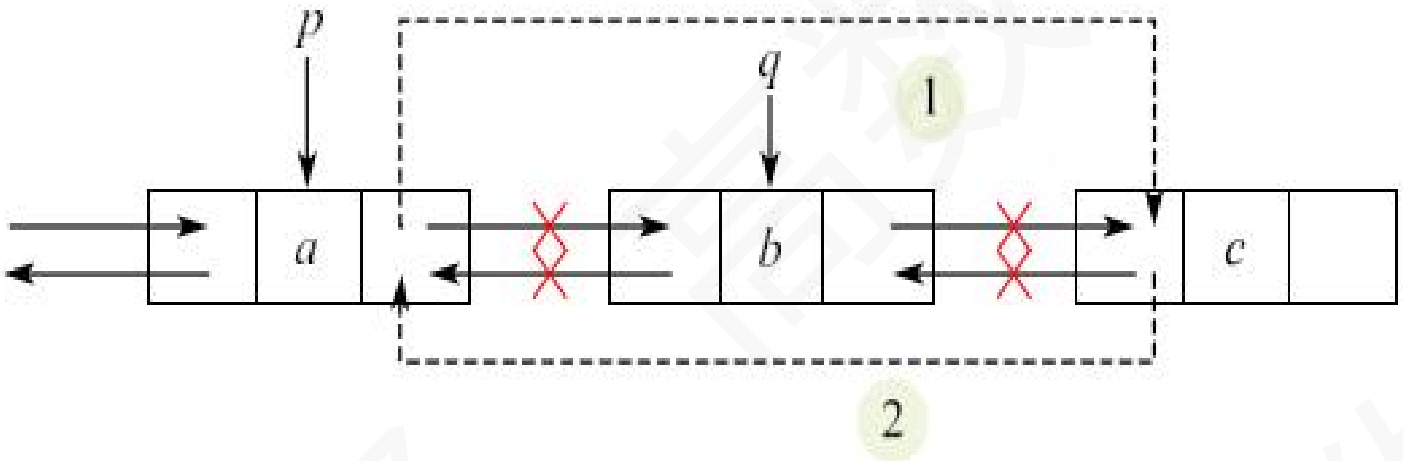
③s->prior=p;

④p->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;
p->next->prior=p;
free(q);
```

视频讲解更清晰
仅4小时

插入操作的代码片段如下：

①s->next-p->next;

②p->next->prior=p;

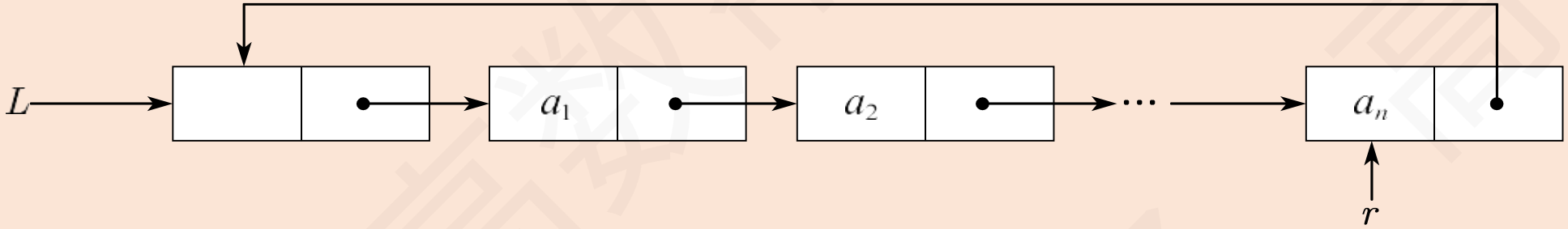
③s->prior=p;

④p->next=s;

其中，①②语句必须在④语句之前。

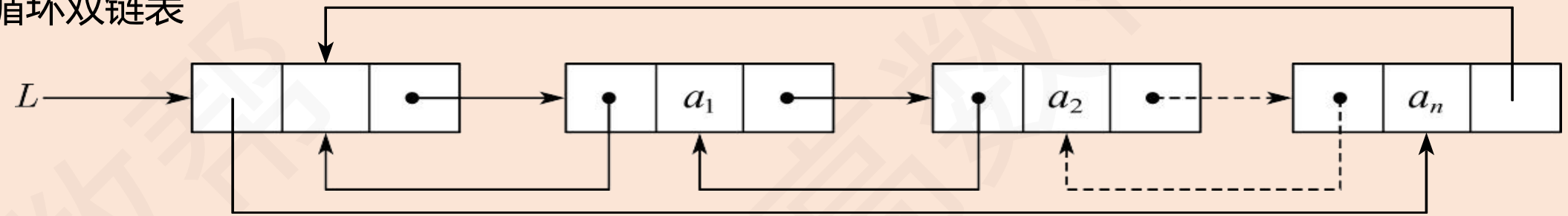
(2)循环链表

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.在链表中最常用的操作是删除表中最后一个结点和在最后一个结点之后插入元素，则采用（）最节省时间

A.带头指针的单向循环链表　　B.双向链表

C.带尾指针的单向循环链表　　D.带头指针的双向循环表

答案：C