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	

4.1 栈的基本概念

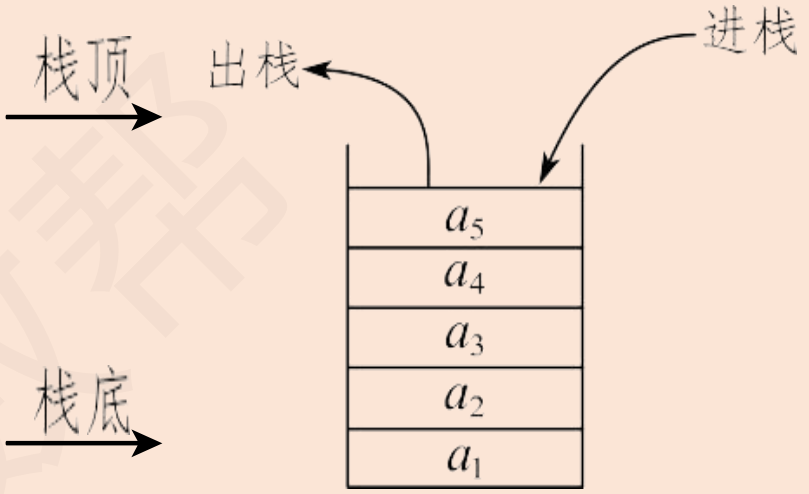
栈(*Stack*)是只允许在一端进行插入或删除操作的线性表。

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*)

4.1 栈的基本概念

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，后进先出

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 $\frac{1}{n+1}C_{2n}^{n}$

4.1 栈的基本概念

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 ture，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）。

（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，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
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];                 //记录栈顶元素
return  true;
}
```

题2.设输入序列为1，2，3，4，5，6，则通过栈的作用后可以得到的输出序列为（）

A.5，3，4，6，1，2　　B.3,2,5,6,4,1

C.3，1，2，5，4，6　　C.1,5,4,6,2,3　　答案 B

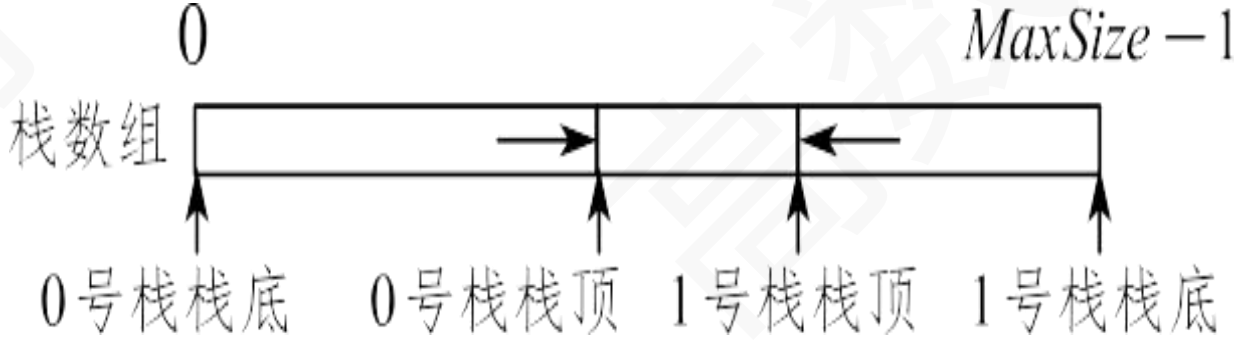
题3.一个栈的输入序列为：a，b，c，d，e，则栈的不可能输出的序列是（）。

A. a,b,c,d,e　　B. d,e,c,b,a

C. d,c,e,a,b　　D. e,d,c,b,a　　答案 C

（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…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（ ）。

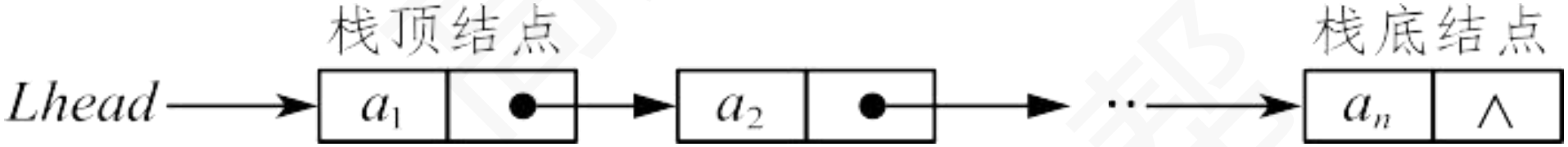
A.|top1-top2|=0　　B. top1+1=top2

C. top1+top2　　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。

4.3 队列的基本概念

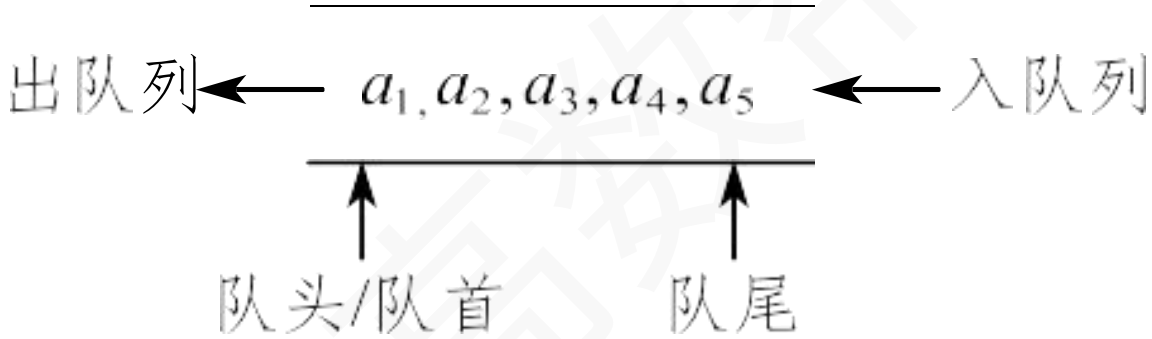
队列(*Queue*)简称队，也是一种操作受限的线性表，只允许在表的一端进行插入，而在表 的另一端进行删除。

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。



4.3 队列的基本概念

题1.队列的插入操作是在（）。

A.队尾　　B.队头　　C. 队列任意位置　　D. 队头元素后　　答案：A

题2.一个队列的入队序列是1，2，3，4，则队列的出队序列是（）。

A. 1，2，3，4　　B. 4，3，2，1

C. 1，4，3，2　　D. 3，4，1，2　　答案：B

题3.设栈S和队列Q的初始状态为空，元素a,b,c,d,e,f,g依次进入栈S。如果每个元素出栈后立即进入队列Q，且7个元素出队的顺序为b,d,c,f,e,a,g则栈S的容量至少是（）

A.1　　B.2　　C.3　　D.4　　答案：C

4.3 队列的基本概念

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。

（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;
```

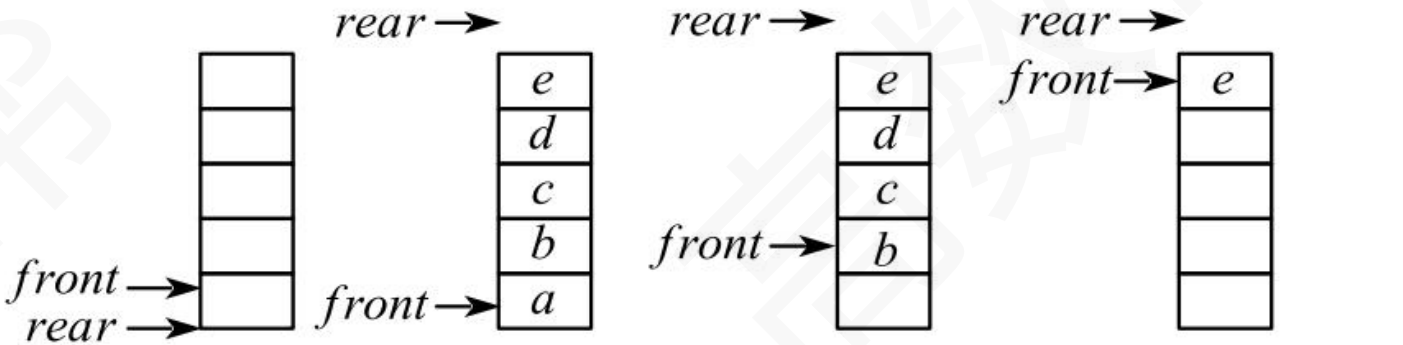
4.4 队列的存储结构

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

4.4 队列的存储结构

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 。

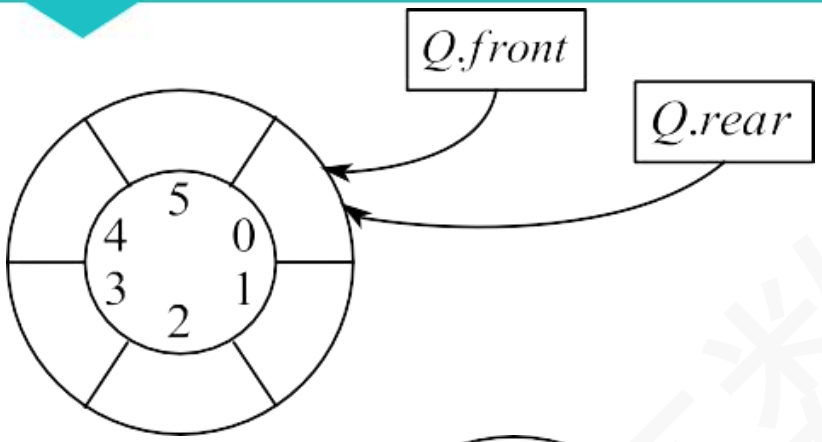
牺牲一个单元来区分队空和队满，入队时少用一个队列单元。

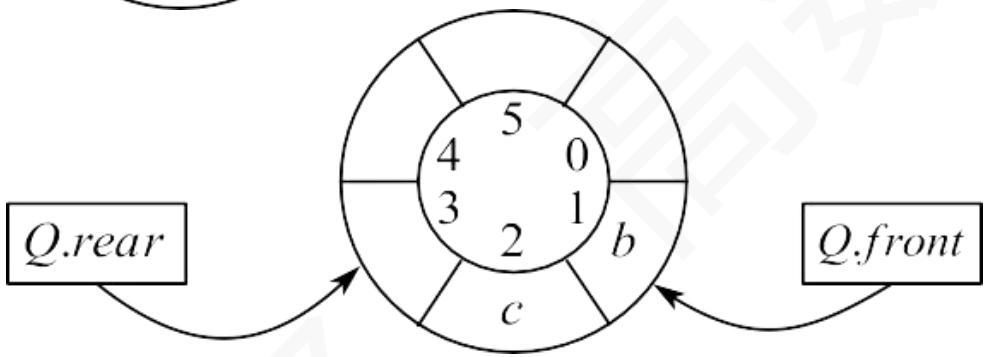
队满条件：$(Q.rear + 1)\%MaxSize == Q.front$

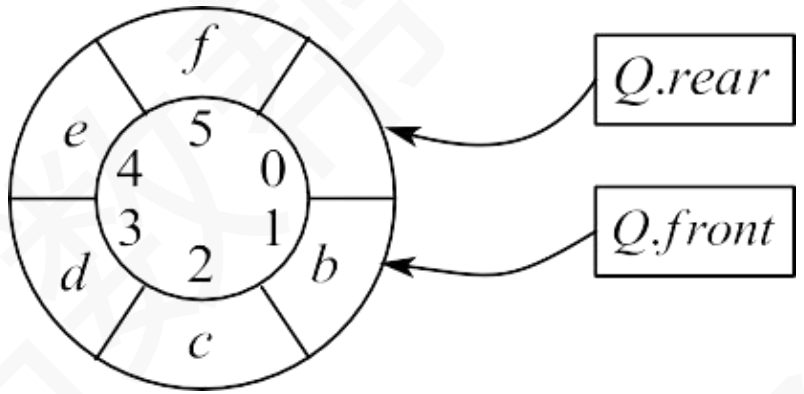
队空条件：$Q.front == Q.rear$

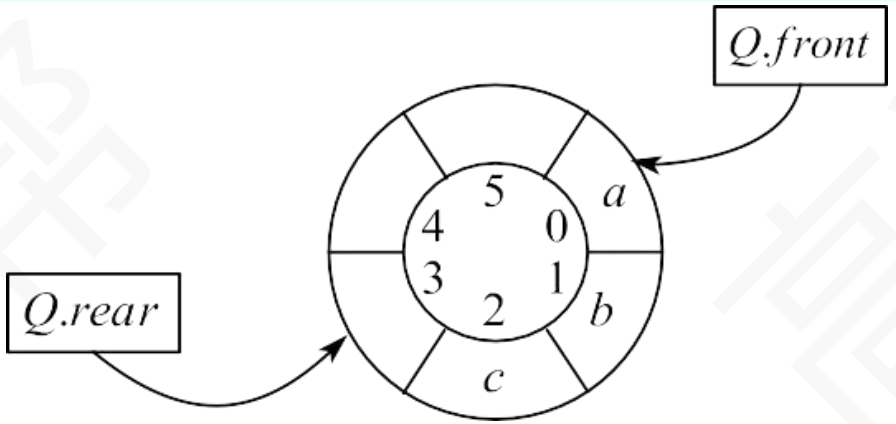
队列中元素的个数：$(Q.rear - Q.front + MaxSize)\%MaxSize$

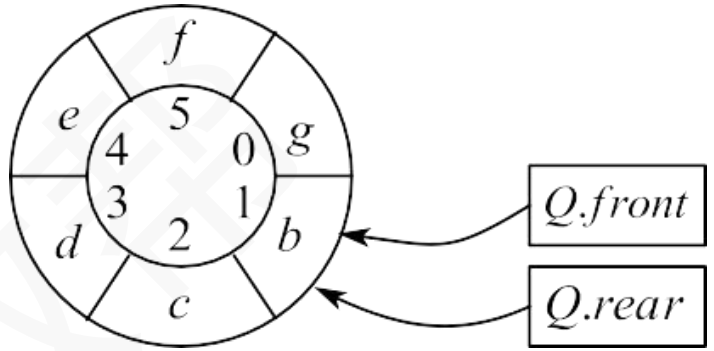
4.4 队列的存储结构


Q.front
Q.rear
5
4
0
3
1
2


Q.rear
Q.front
5
4
0
3
1
2
b
c


f
e
5
4
0
3
1
2
d
b
c
Q.rear
Q.front


Q.front
Q.rear
5
a
4
0
3
1
2
b
c


f
e
5
g
4
0
3
1
d
2
b
c
Q.front
Q.rear

4.4 队列的存储结构

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（ ）。

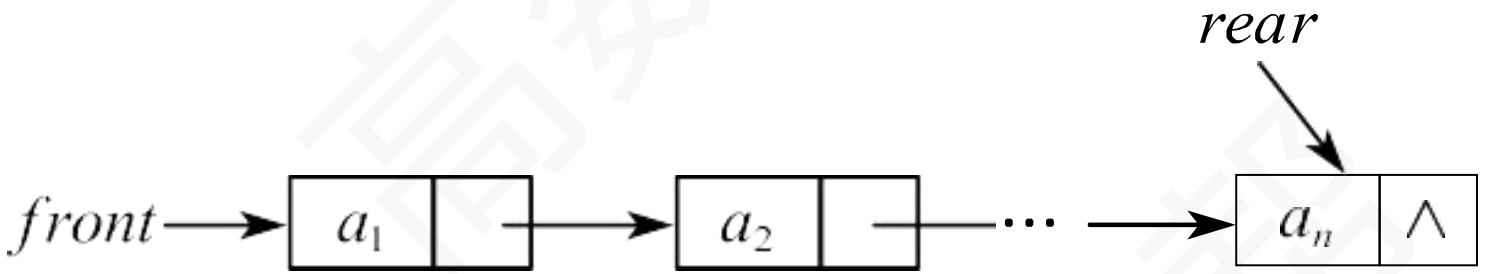
A.Q.front == Q.rear　　B.Q.front ==(Q.rear+1)%m

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（ ）。

答案：A

A.front=front->next　B.rear=rear->next　C.rear=front->next　D.front=rear->next

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	

视频讲解更清晰
仅4小时

5.1 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(∅)。

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。

5.1 串的基本概念

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.空串和空格串（ ）。

A.相同　　B. 不相同　　C. 可能相同　　D. 无法确定

答案：B

题2.设SUBSTR(S,i,k)是求S中从第i个字符开始的连续k个字符组成的子串的操作，则对于S=’Beijing&Nanjing’，SUBSTR(S,4,5)=（ ）。

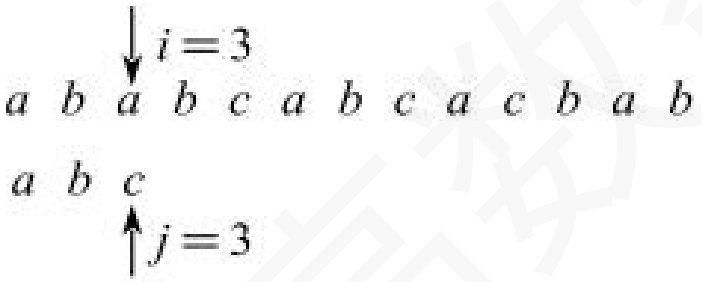
A.‘ijing’　　B. ‘jing&’　　C. ‘ingNa’　　D. ‘ing&N’

答案：B

5.2 串的简单模式匹配算法

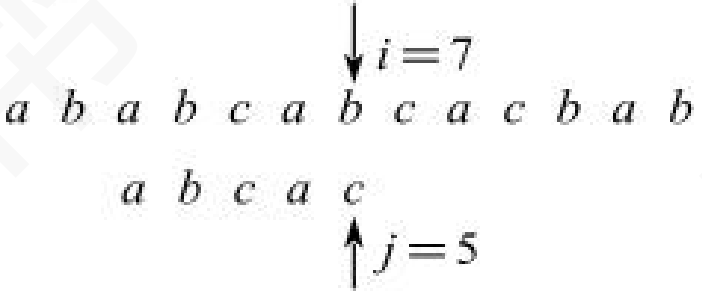
子串的定位操作通常称为串的模式匹配，它求的是子串在主串中的位置。

第一趟匹配

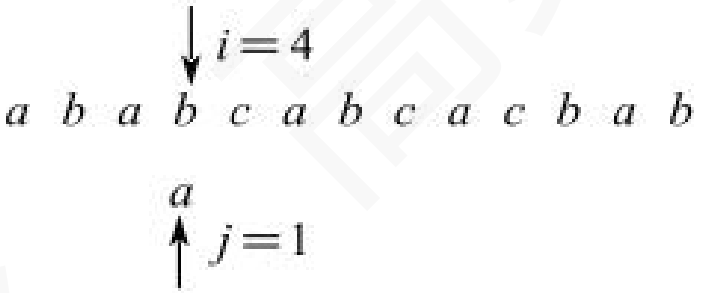


第二趟匹配

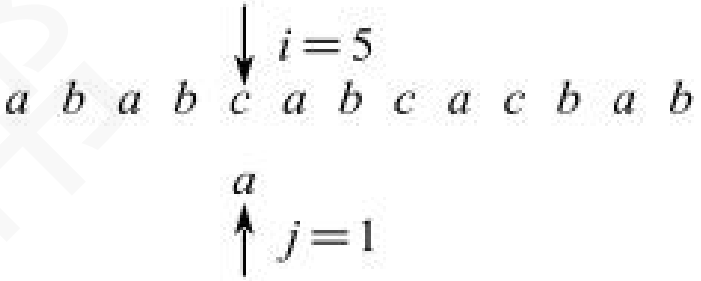
第三趟匹配



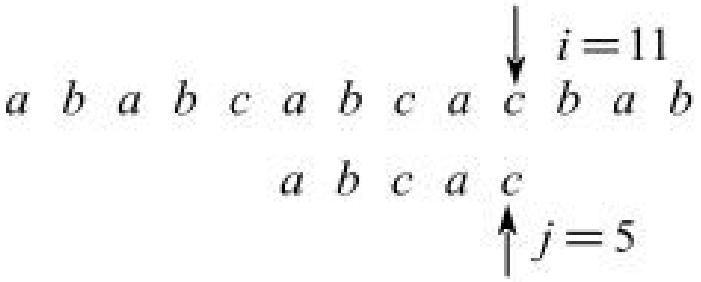
第四趟匹配



第五趟匹配



第六趟匹配



题1.设有两个串S1和S2，求串S2在S1中首次出现位置的运算称作（ ）。

A.连接　　B. 求子串　　C. 模式匹配　　D. 判断子串　　答案：C

5.3 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。

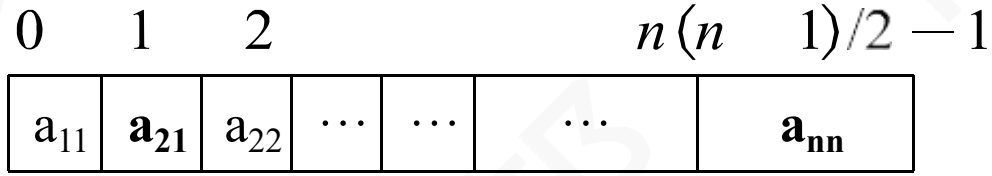
特殊矩阵：指具有许多相同矩阵元素或零元素，并且这些相同矩阵元素或零元素的分布有一定规律性的矩阵。常见的特殊矩阵有对称矩阵、上（下）三角矩阵、稀疏矩阵等。

5.3 矩阵的压缩存储

（1） 对称矩阵

若对一个 n 阶方阵$A\ 1\cdots n\quad 1\cdots n$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。

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



将对称矩阵A[1…n][1…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+…+(i-1+j-i=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，a11为第一个元素，其存储地址为1，每元素占1个地址空间，则a85的地址为（ ）。

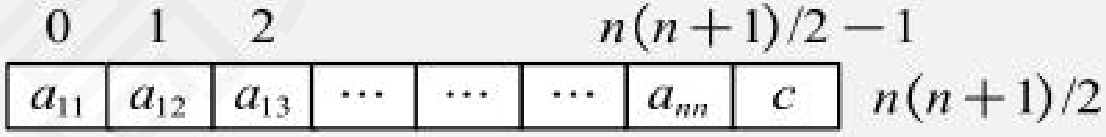
A.13　　B. 33　　C. 18　　D. 40　　答案：B

（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…n][1…n] 压缩存储在B[n(n+1)/2+1] 中。

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

B



5.3 矩阵的压缩存储

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：

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

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

B

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

$n(n+1)/2$

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.设矩阵A是一个对称矩阵，为了节省存储，将其下三角部分按行序存放在一维数组B[1,n(n-1)/2]中，对下三角部分中任一元素ai,j(i>=j)，在一维数组B的下标位置k的值是（ ）。

A.i(i-1)/2+j-1　　B. i(i-1)/2+j　　C. i(i+1)/2+j-1　　D. i(i+1)/2+j

答案：B

（3） 稀疏矩阵

矩阵中非零元素的个数，相对于矩阵元素的个数 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。

5.3 矩阵的压缩存储

题2.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