

数据结构与算法

人工智能与大数据学院

本节课主要内容

- 第4章 数组和字符串
 - ① 掌握数组的定义
 - ② 掌握数组的顺序存储结构
 - ③ 掌握特殊矩阵的压缩方法
 - ④ 掌握串的基本操作实现

4.1 数组——特殊的线性表

1、数组的定义

和线性表一样，数组是由相同数据类型的数据元素组成的，每个元素由一个值和下标确定。

它用一组连续的内存空间，来存储一组类型相同的数
据。

$a[0]$
$a[1]$
$a[2]$
$a[3]$
...
$a[n]$

4.1 数组——特殊的线性表

Q_{00}	Q_{01}	Q_{02}	...	$Q_{0, n-1}$
Q_{10}	Q_{11}	Q_{12}	...	$Q_{1, n-1}$
Q_{20}	Q_{21}	Q_{22}	...	$Q_{2, n-1}$
...	$Q_{3, n-1}$
...
$Q_{m-1,0}$	$Q_{m-1,1}$	$Q_{m-1,2}$...	$Q_{m-1, n-1}$

数组是线性表的推广，它的每个数据元素也是个线性表

4.1 数组——特殊的线性表

2、数组的顺序存储结构 a[m][n]

(1) 以行序为主序的存储方式

存储地址计算方式：

$$LOC(i,j) = LOC(0,0) + (n * i + j) * d$$

例题：假设二维数组A有3行4列，按照行优先进行存储，每个元素占4个存储单元，第1个元素的存储单元地址为100，则A[2][2]的地址是？

Q00	Q01	Q02	...	Q0, n-1
Q10	Q11	Q12	...	Q1, n-1
Q20	Q21	Q22	...	Q2, n-1
...	Q3, n-1
...
Qm-1,0	Qm-1,1	Qm-1,2	...	Qm-1, n-1

4.1 数组——特殊的线性表

2、数组的顺序存储结构a[m][n]

(2) 以列序为主序的存储方式

存储地址计算方式：

$$LOC(i,j) = LOC(0,0) + (i + m * j) * d$$

例题：假设二维数组A有3行4列，按照列优先进行存储，每个元素占4个存储单元，第1个元素的存储单元地址为100，则A[2][2]的地址是？

Q00	Q01	Q02	...	Q0, n-1
Q10	Q11	Q12	...	Q1, n-1
Q20	Q21	Q22	...	Q2, n-1
...	Q3, n-1
...
Qm-1,0	Qm-1,1	Qm-1,2	...	Qm-1, n-1

4.1 数组——特殊的线性表

2、数组的顺序存储结构a[m][n]

例题1：假设将100个数据元素的线性表存储在数组中，若第5个数据元素的地址为1000，第8个数据元素的地址为1030，请问最后一个元素的存储地址是？

例题2：假设数组A[60,70]的基地址为2000，每个元素占2个存储单元，若以列序为主序顺序存储，则元素a[31,50]的存储地址是？

4.1 数组——特殊的线性表

矩阵的压缩存储方法：

压缩存储，是指为多个值相同的元素只分配一个存储空间，对零元素不分配空间。

1. 特殊矩阵
2. 稀疏矩阵

4.1 数组——特殊的线性表

1. 特殊矩阵

若值相同的元素或零元素在矩阵中的分布有一定的规律，则称此类矩阵为特殊矩阵。

(1) 对称矩阵

如果n阶矩阵A中的元素满足

$$a_{ij} = a_{ji} \quad (0 \leq i, j \leq n-1)$$

$$\begin{bmatrix} 3 & 2 & 7 \\ 2 & 5 & 6 \\ 7 & 6 & 9 \end{bmatrix}$$

4.1 数组——特殊的线性表

(1) 对称矩阵

如果n阶矩阵A中的元素满足 $a_{ij} = a_{ji} \quad (0 \leq i, j \leq n-1)$

只存储主对角线以上或以下的元素即可。
即 n^2 个元素压缩存储到 $n(n+1)/2$ 个空间中

$$\begin{bmatrix} 3 & 2 & 7 \\ 2 & 5 & 6 \\ 7 & 6 & 9 \end{bmatrix}$$

4.1 数组——特殊的线性表

若以行序为主存储主对角线以下（含主对角线）的元素，并以一维数组M[n(n+1)/2]作为n阶矩阵A的存储结构，则

a_{00}			
a_{10}	a_{11}		
a_{20}	a_{21}	a_{22}	
...
$a_{n-1,0}$	$a_{n-1,1}$	$a_{n-1,2}$	$a_{n-1,n-1}$

M[n(n+1)/2]中矩阵元素 a_{ij} 的存储状况为：

a_{00}	a_{10}	a_{11}	a_{20}	a_{21}	a_{22}	a_{30}	...	$a_{n-1,n-1}$
----------	----------	----------	----------	----------	----------	----------	-----	---------------

4.1 数组——特殊的线性表

M[n(n+1)/2]中矩阵元素 a_{ij} 的存储状况为（行序为主序）：

a_{00}	a_{10}	a_{11}	a_{20}	a_{21}	a_{22}	a_{30}	...	$a_{n-1,n-1}$
----------	----------	----------	----------	----------	----------	----------	-----	---------------

M[k]和矩阵元素 a_{ij} 的一一对应关系为：

$$k = i(i+1)/2 + j; \quad (i \geq j)$$

4.1 数组——特殊的线性表

若以列序为主存储对角线以下（含对角线）的元素，并以一维数组 $M[n(n+1)/2]$ 作为 n 阶矩阵 A 的存储结构，则

a_{00}				
a_{10}	a_{11}			
a_{20}	a_{21}	a_{22}		
...	
$a_{n-1,0}$	$a_{n-1,1}$	$a_{n-1,2}$...	$a_{n-1,n-1}$

$M[n(n+1)/2]$ 中矩阵元素 a_{ij} 的存储状况为：

a_{00}	a_{10}	a_{20}	a_{30}	...	$a_{n-1,0}$	a_{11}	...	$a_{n-1,n-1}$
----------	----------	----------	----------	-----	-------------	----------	-----	---------------

4.1 数组——特殊的线性表

2. 稀疏矩阵

若矩阵中有很多元素是零，而且非零元素的分布没有规律，则称该矩阵为稀疏矩阵。

$$\begin{bmatrix} 0 & 2 & 0 \\ 0 & 0 & 4 \\ 0 & 6 & 0 \end{bmatrix}$$

如果采用一般的存储方法表示稀疏矩阵，就会存储大量的零元素，造成存储空间的浪费。

4.1 数组——特殊的线性表

2. 稀疏矩阵

(1) 三元组顺序表

```
define MAXSIZE 20
```

```
typedef struct{
```

```
    int i, j;
```

非零元素的行、列下标

```
    elemtype v;
```

非零元素的值

```
} node;
```

```
typedef struct{
```

```
    int m, n, t;
```

稀疏矩阵的行数、列数、非零元素个数

```
    node data[MAXSIZE];
```

```
} mat;
```

$$\begin{bmatrix} 0 & 2 & 0 \\ 0 & 0 & 4 \\ 0 & 6 & 0 \end{bmatrix}$$

i	j	v
3	3	3
1	2	2
2	3	4
3	2	6

注：表中 $data[0]$ 中的 i, j, v 分别存储稀疏矩阵的行数、列数、非零元素个数

4.1 数组——特殊的线性表

$\begin{bmatrix} 0 & 2 & 0 \\ 0 & 0 & 4 \\ 0 & 6 & 0 \end{bmatrix}$

i	j	v
3	3	3
1	2	2
2	3	4
3	2	6

如何实现稀疏矩阵的转置运算，即计算转置矩阵？？

4.1 数组——特殊的线性表

$\begin{bmatrix} 0 & 2 & 0 \\ 0 & 0 & 4 \\ 0 & 6 & 0 \end{bmatrix}$

i	j	v
3	3	3
1	2	2
2	3	4
3	2	6

算法思路：

- ① 将行数和列数相互交换
- ② 将三元组中的i和j相互调换
- ③ 重排三元组之间的次序

4.1 数组——特殊的线性表

算法思路：

- ① 将行数和列数相互交换
- ② 将三元组中的i和j相互调换
- ③ 重排三元组之间的次序

如果用二维数组来表示矩阵，一般可用如下算法实现：

```
for (j=1; j<=n; j++)  
    for (i=1; i<=m; i++)  
        b[j][i]=a[i][j]
```

此时时间复杂度为 $O(m * n)$

如果用二维数组来表示矩阵，一般可用如下算法实现：

```
for (j=1; j<=n; j++)  
    for (i=1; i<=m; i++)  
        b[j][i]=a[i][j]
```

适当增加存储单元，降低时间复杂度。

- ① 确定a中每一列第一个非零元素在b中的位置；
 - ② 计算每个非零元素在b中的位置
- 此时时间复杂度为 $O(n * t)$ ，其中t为非零元素个数

4.1 数组——特殊的线性表

4.2 字符串——特殊的线性表

1、字符串的定义

字符串（简称串），是一种特殊的线性表，它的数据元素仅由一个字符组成。

串是由零个或多个字符组成的有限序列。一般记为

$$S = 'c_0c_1c_2 \dots c_{n-1}' \quad (n \geq 0)$$

其中 c_i 可以是字母、数字或其他字符

4.2 串——特殊的线性表

2、相关概念

- ① 串名
- ② 串的值
- ③ 串的长度
- ④ 子串和主串
- ⑤ 串的位置
- ⑥ 两个串相等
- ⑦ 空格串
- ⑧ 空串

$$S = 'c_0c_1c_2 \dots c_{n-1}' \quad (n \geq 0)$$

4.2 串——特殊的线性表

3、字符串的基本操作定义

字符串的基本操作常常以串或子串为单位，而不是以一个字符为单位。

假设本节中的s, t, v, a, b, c, d都是串名，并且a= 'HE'，
b= 'FEI'，
c= '，'，d= 'HEFEI'

a= 'HE'， b= 'FEI'， c= '，'， d= 'HEFEI'

(1) 赋值操作：

Assign(s, t)：将串 t 的值赋值给串 s

例如：执行Assign(s, d)之后，s= 'HEFEI'

C语言中string.h头文件中，函数strcpy(str1, str2)将字符串str2的值赋值给str1，函数strncpy(str1, str2, n)将字符串str2中前n个字节赋值给str1

4.2 串——特殊的线性表

a= 'HE' , b= 'FEI' , c= ' ' , d= 'HEFEI'

(2) 判相等函数:

Equal(s, t): 若串 t和串s相等, 则返回函数值1, 否则返回0

例如: 执行Equal(a, d)之后, 结果为0

思考: Equal(a, c), Equal(a, 'HE'), Equal(' ' , ' ')

4.2 串——特殊的线性表

a= 'HE' , b= 'FEI' , c= ' ' , d= 'HEFEI'

(3) 计算串的长度:

Len(s) : 返回串s中字符的个数

例如: 执行 Len(a)之后, 结果为2

思考: Len(c), len(' ')

4.2 串——特殊的线性表

a= 'HE' , b= 'FEI' , c= ' ' , d= 'HEFEI'

(4) 串的连接函数:

Concat(s, t) : 将串t的字符序列紧接在串s的字符序列之后, 构成一个新的字符序列, 产生一个新串, 并返回

例如: 执行Concat(a, b)之后, 结果为 'HEFEI'

思考 Equal(d, Concat(a, b))的结果?

4.2 串——特殊的线性表

a= 'HE' , b= 'FEI' , c= ' ' , d= 'HEFEI'

(5) 求子串函数:

SubStr(s, start, len) : 如果合理区间范围内 (start:[0, length(s)-1], len:[0, length(s)-start+1]), 则返回函数值为从串s中第start个字符起, 长度为len的连续字符序列, 否则返回一个特殊的串常量。

例如: 执行SubStr(a, 0, 1)之后, 结果为 'H'

4.2 串——特殊的线性表

练习题：

1、串s1= 'ABCDEFG'，串s2= 'PQRST'，则

Concat (SubStr (s1, 2, len(s2)), SubStr (s1, len(s2), 2)) 的结果是 ？

2、串s1= 'HEFEI'，串s2= 'HELLO'，则

Equal (SubStr (s1, 0, len (SubStr (s2, 1, 2))), 'HE') 的结果是 ？

4.2 串——特殊的线性表

a= 'HE'，b= 'FEI'，c= ''，d= 'HEFEI'

(6) 定位函数：

Index(s, t)：若在主串s中存在和t相等的子串，则函数值为s中第一个这样的子串在主串s中的位置，否则函数值为-1

例如：执行Index(a, 'E')之后，结果为1，执行Index(d, 'E')的结果呢？

注意⚠：t不能为空串

4.2 串——特殊的线性表

a= 'HE'，b= 'FEI'，c= ''，d= 'HEFEI'

(7) 替换函数：

Replace(s, t, v)：以串v替换所有在串s中出现的和非空串t相等的重叠的子串。

例如：执行Replace(d, 'E'，'e')之后，s结果更新为 'HeFeI'

注意⚠：t不能为空串

4.2 串——特殊的线性表

练习题：

1、串s1= 'ABCDEFG'，串s2= 'PQRST'，则

Concat (SubStr (s1, 0, Index (s2, 'Q')), SubStr (s1, len (s2), 2)) 的结果是 ？

2、串s1= 'HEFEI'，串s2= 'HELLO'，则

Replace(s2, 'O'，'0!')之后串s2的值？

4.2 串——特殊的线性表

a= 'HE' , b= 'FEI' , c= '' , d= 'HEFEI'

(8) 插入函数:

Insert(s, pos, t) : 若pos的值在合理区间范围内 ([0, length(s)]), 则在串s的第pos个字符之前插入串t

例如: 执行Insert(d, 0, 'Hi,') 之后, d结果更新为 'Hi, HEFEI'

4.2 串——特殊的线性表

a= 'HE' , b= 'FEI' , c= '' , d= 'HEFEI'

(9) 删除函数:

Delete(s, pos, len) : 若pos的值在合理区间范围内 ([0, length(s)-1]), 则从串s中删除第pos个字符起, 长度为len的子串

例如: 执行Delete(d, 2, 3) 之后, d结果更新为 'HE'

4.2 串——特殊的线性表

判断如下程序段的实现功能(1):

```
int abc(char *a){  
    int count=0,i;  
    for(i=0;a[i]!='\0';i++)  
        count++;  
    return count;  
}
```

4.2 串——特殊的线性表

判断如下程序段的实现功能(2):

```
int abc(char *a){  
    int i=0;  
    while(a[i]!='\0')  
        i++;  
    return i;  
}
```

4.2 串——特殊的线性表

判断如下程序段的实现功能 (3) :

```
void def(char *a, char *b){
    int len1=abc(a);
    int len2=abc(b);
    int i;
    for(i=0;i<len2;i++)
        a[len1+i]=b[i];
}
```

4.2 串——特殊的线性表

判断如下程序段的实现功能 (4) :

```
int uvw(char *a, char *b){
    int len1=abc(a);
    int len2=abc(b);
    int i,j;
    if(len1!=len2)
        return 0;
    for(i=0;i<len1;i++)
        if(a[i]!=b[i])
            return 0;
    return 1;
}
```

4.2 串——特殊的线性表

判断如下程序段的实现功能 (5) :

```
char *rst(char *a, int s, int m){
    char *c;
    c=malloc(sizeof(abc(a)));
    int i=0;
    for(i=0;i<m;i++)
        c[i]=a[s+i];
    return c;
}
```

4.2 串——特殊的线性表

判断如下程序段的实现功能 (6) :

```
int xyz(char *a, char *b){
    int len1=abc(a);
    int len2=abc(b);
    int i=0;
    if(len2==0)
        return -1;
    while(i<=(len1-len2)){
        if(uvw(rst(a, i, len2),b))
            return i;
        i++;
    }
    return -1;
}
```

4.2 串——特殊的线性表

课后作业：利用串的基本运算，编写一个算法，实现删除s1中所有s2子串