第四章　串

王　　勇
计算机/软件学院　　大数据分析与信息安全团队
21#518　　电　话 13604889411
Email：　wangyongcs@hrbeu.edu.cn

问题提出

- 非数值信息字符串处理
- 信息检索系统
- 用户以关键词查找信息
- 数据挖掘

本章说明

知识点

- 串的类型定义
- 串的存储表示

重　点

- 了解基本操作的定义
- 串的实现方法
- 会利用这些基本操作

本章说明

理解

“串”类型定义中各基
进行串的 特
其它操作

正确利用

学习目标

本章内容

1 串及其操作

2 串的表示和实现

3 串的模式匹配

4 本章小结

- 串
- 串与线性表区别
- 串的抽象数据类型

串

是由零个或多个字符组成的

一般记为 $S='a_1a_2a_3\ldots a_n'$（

a_i（$1\le i\le n$）可以是字母、

串值必须用一对单引号括起来

- S—串名
- 串的值—表头用单引号括起的字符序列

概　念

- 串的长度：串中字 为串的长度
- 空串：零个字符 ，用Φ或’ ’来表示

空串与空格串的区别？

- 空格串：由一个或多个空格组成的串。
- 子串：串中任意个连续的字符组成的子序列
- 主串：包含子串的串
- 位置：字符在序列中序号
- 子串在主串中的位置:其第1个字符在主串中的位置
- 相等：当且仅当这两个串的值相等

例 如

a=‘BEI’
b=‘JING’
c=‘BEIJING’
d=‘BEI JING’
长度分别为3、4、7、8
a和b都是c和d的子串
a在c和d中的位置都是1
b在c中的位置是4，b在d中的位置是5
a、b、c、d彼此不相等

注 意

串变量与其它变量的区别
x=‘123’（是串变量）与x=123是不同的

- 串的数据对象约束为字符集
- 串的基本操作与线性表有很大差别
  - 线性表大多以“单个元素”作为操作对象，如查找某个元素、在某个位置上插入一个元素和删除一个元素
  - 串通常以“串的整体”作为操作对象。如在串中查找某个子串、在串的某个位置上插入一个子串以及删除一个子串

ADT String {

数据对象： $D=\{a_i|a_i \in CharacterSet, i=1,2,...,n, n\geq 0\}$

数据关系： $R1=\{<a_{i-1},a_i>|a_{i-1},a_i \in D, i=2,...,n\}$

基本操作：

StrAssign (&T, chars)

初始条件： chars 是串常量。

操作结果： 赋于串T的值为 chars。

StrCopy (&T, S)

初始条件： 串 S 存在。

操作结果： 由串 S 复制得串 T。

DestroyString (&S)

初始条件：串 S 存在。

操作结果：串 S 被销毁。

StrEmpty (S)

初始条件：串 S 存在。

操作结果：若 S 为空串，则返回 TRUE，否则返回 FALSE。

StrCompare (S, T)

初始条件：串 S 和 T 存在。

操作结果：若S>T，则返回值>0；若S=T，则返回0

若S<T，则返回值<0。

StrLength (S)

初始条件：串 S 存在。

操作结果：返回串 S 序列中的字符个数，即串长度

ClearString (&S)

初始条件：串 S 存在。

操作结果：将 S 清为空串。

Concat (&T, S1, S2)

初始条件：串 S1 和 S2 存在。

操作结果：用 T 返回由 S1 和 S2 联接而成的新串

SubString (&Sub, S, pos, len)

初始条件：串S存在，1≤pos≤StrLength(S)且

0≤len≤StrLength(S)-pos+1。

操作结果：用 Sub 返回串S的第 pos 个字符起长度为len的子串。

Index (S, T, pos)

初始条件：串S和T存在，T 非空串，

1≤pos≤StrLength(S)

操作结果：若主串S中存在和串T值相同的子串，则返回它在主串S中第pos个字符之后第一次出现的位置；否则函数值为0。

Replace (&S, T, V)

初始条件：串 S，T 和 V 存在，T 是非空串。

操作结果：用V替换主串S中出现的所有与T相等的不重叠的子串。

StrInsert (&S, pos, T)

初始条件：串S和T存在，1≤pos≤StrLength(S)＋1

操作结果：在串 S 的第 pos 个字符之前插入串 T

StrDelete (&S, pos, len)

初始条件：串 S 存在，1≤pos≤StrLength(S)-len+1操作结果：从 S 中删除第 pos 个字符起长为len 子串

算 法 4.1 子串定位函数Index，利用已有操作函数

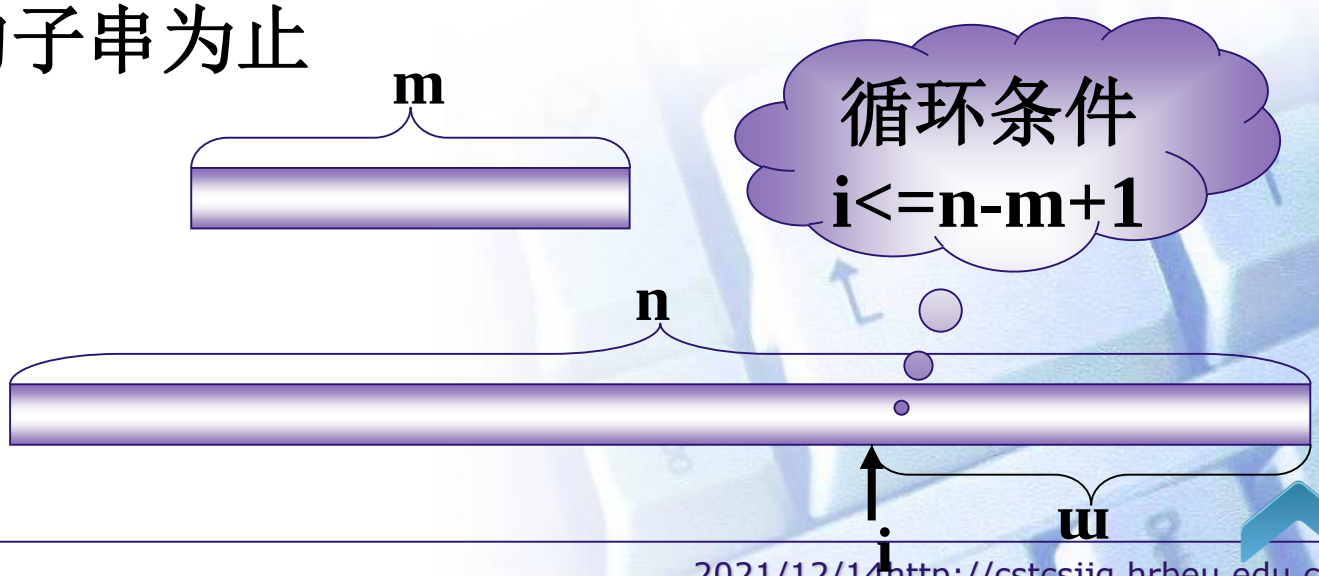
思　想

从主串S中取“第i个字符起、长度和串T相等的子串”与串T比较：

若相等，则求得函数值为i，否则i值增1

直至找到相等的子串或者不存在和T相等的子串为止

Click

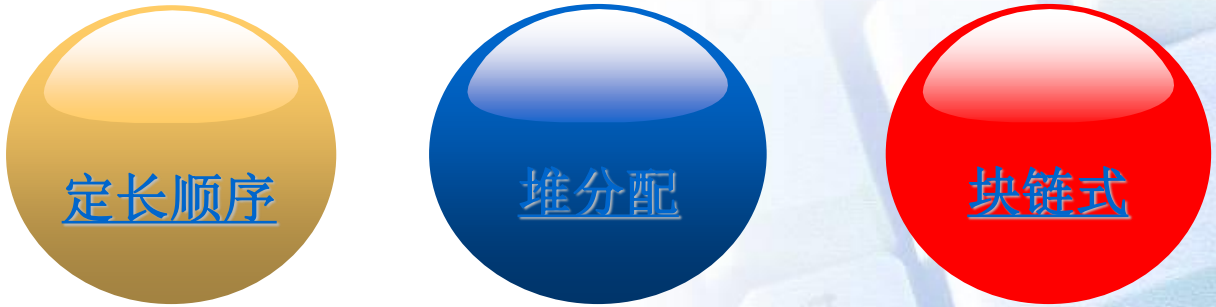


子串定位算法

```
int Index(String S, String T, int pos)
{  //T为非空串。若主串S中第pos个字符之后存在与T相等的子串
   //则返回第一个这样的子串在S中的位置，否则返回0
   if (pos>0)
      {  n=StrLength(S);  m=StrLength(T);  i=pos;
         while (i<=n-m+1)    //
            {  SubString(sub, S, i, m);
               if (StrCompare(sub, T) !=0)   ++i;
               else  return  i;        //返回子串在主串的位置
             }//while
       }//if
   return  0;                          //S中不存在与T相等的子串
}//Index
```

串的表示和实现

顺序存储、链式存储



串的表示和实现　　定长顺序存储表示

串的表示　用一组地址连续的存储单元存储串值的字符序列

数组描述

```
#define  MAXSTRLEN 255;
typedef  unsigned  char
SString[MAXSTRLEN +1];
```

提　　示　串长有两种表示方法

- Sstring[0]表示
- 串值后加一个不计入串长的结束标记“\0”

操作实现　串联接、求子串等

截断：串长度超过了最大长度的串值被舍去

例　如　串联接操作时，设串变量的最大串长度为10

S1=‘ABCDEF’　S2=‘GHIJ’

S3=‘KLMNOP’　S4=‘QRSTUVWXYZ’

则：T1=S1联接S2=‘ABCDEFGHIJ’

（将S1、S2串值复制到T1）

T2=S1联接S3=‘ABCDEFKLMN’

（S3的‘OP’部分截断）

T3=S4联接S1=‘QRSTUVWXYZ’

（S1被全部截去）

算 法 4.2

思　想

串联接

S1[0], S2[0]存串长度，T[0]联接后长度

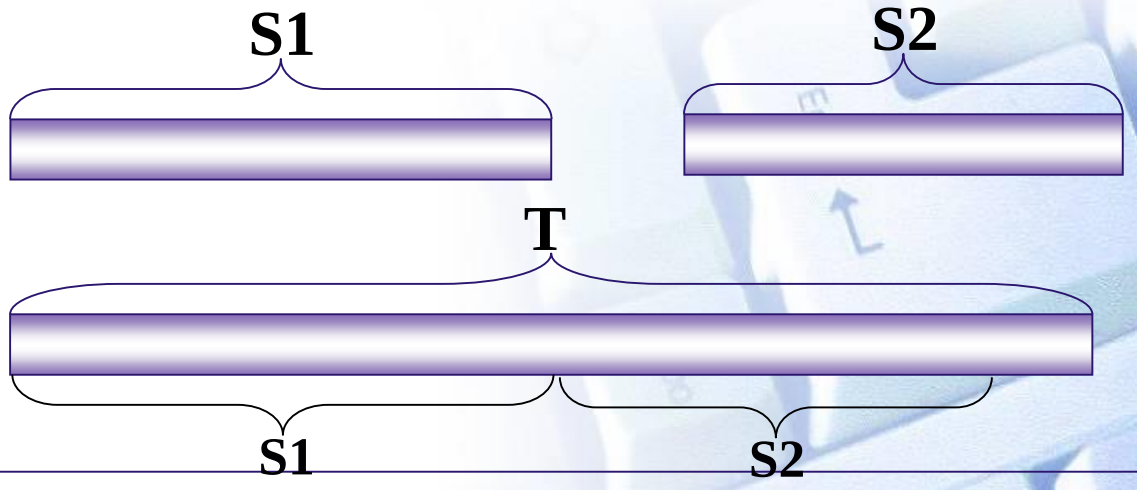
三种情况：

（1）S1[0]+S2[0] <= MAXSTRLEN

（2）S1[0]<MAXSTRLEN，S2截断

（3）S1[0]>=MAXSTRLEN，S1截断

是否截断设一标志位uncut，未截断T，否则F



串联接算法

```
Status Concat(SString &T, SString S1, SString S2)
{ //T返回由S1和S2联接而成的新串。若未截断，则返回T，否则F
  if (S1[0]+S2[0]<=MAXSTRLEN)          //未截断
      { T[1..S1[0]]=S1[1..S1[0]];
        T[S1[0]+1..S1[0]+S2[0]]=S2[1..S2[0]];
        T[0]=S1[0]+S2[0];
        uncut=TRUE;
      }
  else if (S1[0]<MAXSTRLEN)            //截断
          { T[1..S1[0]]=S1[1..S1[0]];
            T[S1[0]+1..MAXSTRLEN]=S2[1..MAXSTRLEN-S1[0]];
            T[0]=MAXSTRLEN;
            uncut=FALSE;
          }
       else { T[1..MAXSTRLEN]=S1[1..MAXSTRLEN];
              uncut=FALSE;          //截断，仅取S1
            }
}
```

这种情况，在插入、置换等操作中也可能发生。克服这个弊病唯有不限定串的最大长度，即采用动态存贮结构

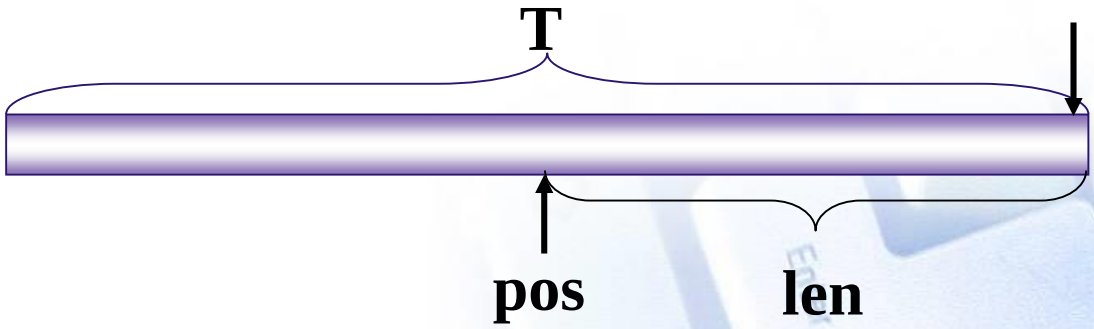
算 法 4.3　求子串，串S的第pos个字符起长度为len的子串

思　想

- 判pos,len是否合法:　　Click

  1=<pos=<S[0]，0< len<=S[0]-pos+1

- 合法，从pos开始取len个字符，即pos..pos+len-1
- 设定子串长度len



特　点

- 实现串操作的原操作为“字符序列的复制”
- 操作的时间复杂度基于复制字符序列的长度
- 在操作中如出现结果超长，约定用截断法处理

求子串算法

```
Status SubString(SString &Sub, SString S, int pos, int len)
{  //用Sub返回串S的第pos个字符起长度为len的子串
   //其中，1=<pos=<StrLength(S) 且0=<len=<StrLength(S)-pos+1
    if (pos<1 || pos>s[0] || len<0 || len>S[0]-pos+1)
        return  ERROR;
    sub[1..len]=S[pos..pos+len-1];
    sub[0]=len;
    return  OK;
}//SubString
```

堆 分 配

系统开辟一个串值存储空间（串值可利用空间），同时建立一个符号表

建立一个新串时，在可利用空间分配，并在符号表中记录下串变量名、串值在可利用空间的位置、串长度等信息

符号表		
串名	位置	长度
a	1	3
b	4	4
c	8	8
d	16	6

串值存储空间

B	E	I	J	I	N	G	S	H	A
N	G	H	A	I	H	A	R	B	I
N									

串的表示和实现

堆分配存储表示

存储表示

```
typedef  struct
      {    char  *ch;
           int    length;
       } Hsstring;
```

算 法 4.4

串插入，在串S的第pos个字符之前插入串T

思　想

- 判pos是否合法:

  1=<pos=<S.length+1

- 如果T不空，变更S空间
- 从最后一个字符到pos依次后移T[0]个位置
- 插入T

Click

串的ADT

插入串算法

```
Status StrInsert(HString &S, int pos, HString T)
{   //1=<pos=<Strleng(S)+1 在串S的第pos个字符之前插入串T
    if (pos<1 || pos>S.length+1) return ERROR;    //pos不合法
    if (T.length)                          //T非空，则重新分配空间，插入T
        {
  if(!(S.ch=(char*)realloc(S.ch,(S.length+T.length)*sizeof(char))))
                exit (OVERFLOW);
          for (i=S.length-1; i>=pos-1; --i)  //为插入T而腾出位置
                S.ch[i+T.length]=S.ch[i];
          S.ch[pos-1..pos+T.length-2]=T.ch[0..T.length-1];
             //插入T，即[pos-1..pos-1+T.length-1]
          S.length+=T.length;
        }
    return OK;
}//StrInsert
```

存储表示

用链表方式存储串值，每个结点大小相同

```
typedef  struct CNode
    {    char              data;
         struct  CNode  *next;
     } LString;
```

例　如

结点大小为1的链表

#占满

结点大小为4的链表

存储密度=串值所占的存储位/实际分配的存储位

提 示

用实际应用时，可以根据问题所需来设置结点的大小

例 如

在编辑系统中，整个文本编辑区可以看成是一个串，每一行是一个子串，构成一个结点。即: 同一行的串用定长结构(80个字符)，行和行之间用指针相联接

定　义

在串中寻找子串（第一个字符）在串中的位置

术　语

在模式匹配中，子串称为模式

主串称为目标

示　例

目标 S :’Beijing’　模式 T:’jin’　匹配结果 = 4

第1趟　S　a b b a b a

　　T　a b a

第2趟　S　a b b a b a

　　T　　a b a

第3趟　S　a b b a b a

　　T　　　a b a

第4趟　S　a b b a b a

　　T　　　　a b a

　　　　　　√

穷举的模式匹配过程

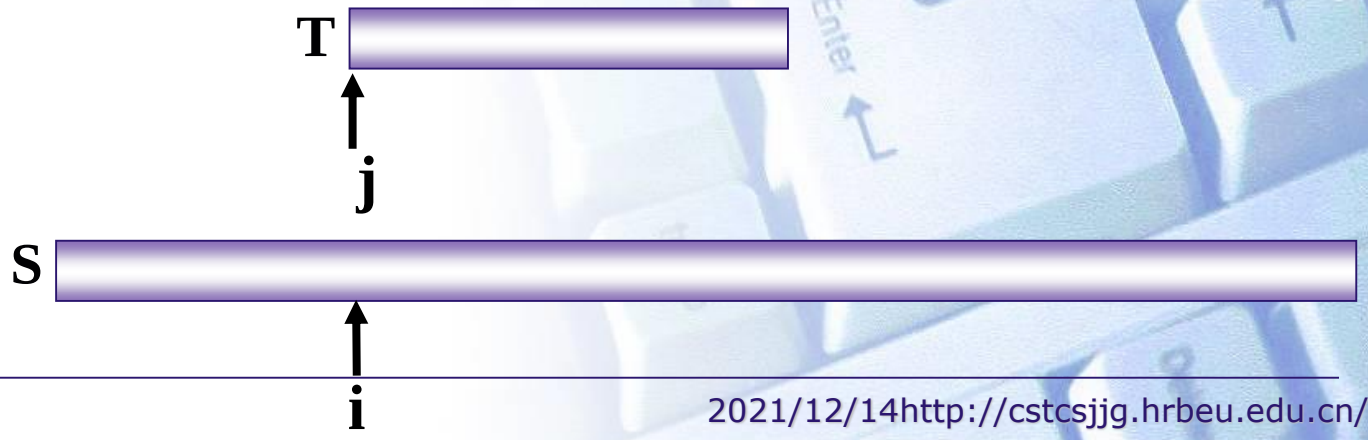
Click

算法思想

- 设两个指针i和j，分别指向主串S的第pos个字符和子串T第1个字符
- i小于等于主S串长，且j小于等于子串T的长度
  - i和j所指对应字符比较
  - 若相等，则i, j后移
  - 指针后退(i-j+2)，重新开始匹配
- 如果j大于T[0]，成功

Click

next



穷举的模式匹配过程

对应字符不等时，i,j重置为j=1,i=i-j+2

```
int index(Sstring S, Sstring T, int pos)
{  //返回子串T在主串S中第pos个字符之后的位置。若不存在返回0
   //其中，T非空，1<=pos<=Strlength(S)
   i=pos;   j=1;
   while(i<=S[0]&&j<=T[0])
        {
          if (S[i]==T[j])
            {  ++i;  ++j;}      //继续比较后继字符
          else
            {  i=i-j+2;         //指针后退重新开始匹配
               j=1;
            }
        }
   if (j>T[0])   return i-T[0];
   else  return 0;
}//Index
```

本章小结

- 数据元素都是字符，因此它的存储结构和线性表有很大不同。
- 多数情况下，实现串类型采用的是“堆分配”的存储结构。
- 串的基本操作通常以”串的整体”作为操作对象，而不像线性表是以”数据元素”作为操作对象。

下课休息一会！