第四章 字符串

一. 概念

1. 串
2. 模式匹配

二. 方法

3. 串的基本操作
4. 串的存储及运算
5. ★串的KMP快速模式匹配算法，求特征向量数组（N数组）和利用N向量完成匹配的方法（注意变种KMP算法的特征定义、KMP算法的灵活应用）

1.串

基本概念

字符串是字符的线性表

串：零个或多个字符/符号构成的有限序列

$S=c_0c_1...c_{n-1}$
串名：S
串长：n
串值：$c_0c_1...c_{n-1}$
空串：串长为0，不包含任何字符内容

子串：一个字符串中任意个连续的字符组成的子序列
空串是任意串的子串，任意串都是其自身的子串
真子串：非空且不为自身的子串

模式匹配

概念

在目标文本T中寻找和定位一个给定模式P的过程

分类

精确匹配：若目标 T 中存在至少一处与模式 P 完全相同的子串，则称为匹配成功，否则匹配失败

近似匹配: 若模式P与目标T(或其子串)存在某种程度的相似，则认为匹配成功

- 字符串相似度通常定义串变换所需基本操作数目
- 基本操作包括插入、删除和替换三种操作 (“编辑距离“)

3.串的基本操作

操作类别	方法	描述
子串	substr ()	返回一个串的子串
拷贝/交换	swap ()	交换两个串的内容
	copy ()	将一个串拷贝到另一个串中
赋值	assign ()	把一个串、一个字符、一个子串赋值给另一个串中
	=	把一个串或一个字符赋值给另一个串中
插入/追加	insert()	在给定位置插入一个字符、多个字符或串
	append () / +=	将一个或多个字符、或串追加在另一个串后
拼接	+	通过将一个串放置在另一个串后面来构建新串
查询	find ()	找到并返回一个子序列的开始位置
替换/清除	replace ()	替换一个指定字符或一个串的字串
	clear ()	清除串中的所有字符
统计	size () / length()	返回串中字符的数目
	max_size ()	返回串允许的最大长度

4.串的存储及运算

- 串长变化不大的字符串采用的定长存储方式，有三种方案：
  (1) 用S[0]作为记录串长的存储单元
    - **缺点**：限制了串的最大长度不能超过256
  (2) 另辟空间存储串的长度
    - **缺点**：串的最大长度一般是静态给定的，而非动态申请
  (3) 特殊标记串的结束
    - '\0' 是ASCII码中8位全0码，又称为 NULL 符
    - C/C++ 语言的的标准字符串（（#include <string.h>）采用

5.串的KMP快速模式匹配算法

```
string T,P;
int Plen=P.size();
int*next=new int[Plen];
void getNext(){
    next[0]=-1;
        j=0,k=0;
    while(j<len){
```

```
        while(k>=0&&P[j]!=P[k])
            k=next[k];
        k++,j++;
        if(j==len)
            break;
        next[j]=k;
        if(P[j]==P[k])//优化
            next[j]=next[k];
    }
}
int kmp(){
    getNext();
    int i=0,j=0;
    int Tlen=t.size();
    if(Tlen<Plen)
        return -1;
    while(i<Tlen&&j<Plen){
        if(j==-1||T[i]==P[j])
            i++,j++;
        else{
            j=next[j];
        }
    }
    if(j==Plen)
        return i-Plen;
    return -1;
}
```