第四章 串

串的定义与实现

- 概念
  - 串的概念：零个或多个任意字符组成的有限序列（内容受限的线性表）
  - 子串的概念：一个串中任意个连续字符组成的子序列（含空串）
- 串的存储结构（和线性表几乎一样，仅将存储数据的类型改为字符）
  - 顺序存储结构
  - 链式存储结构：一个结点若只存一个字符，存储密度过低，引出块链存储结构
  - 块链存储结构：每个结点既可以存放一个字符，也可以存放多个字符。每个结点称为块

串的模式匹配

简单模式匹配算法

- 概念：子串的定位操作通常称为串的模式匹配，它求的是子串（模式串）在主串中的位置
- 算法思想：将主串中与模式串长度相同的子串逐个与模式串匹配，暴力法
- 缺点：主串指针回溯导致时间开销，时间复杂度为：O(nm)，nm分别为主串和模式串的长度

KMP算法

- 基本概念
  - 前缀：除最后一个字符外，字符串的所有头部子串
  - 后缀：除第一个字符外，字符串的所有尾部子串
- 算法思想：主串指针不必回溯，模式串指针根据next数组部分回溯（如果已匹配相等的前缀序列中有某个后缀正好是模式的前缀，可以将模式串向后滑动到与这些相等字符对齐的位置）
- next数组
  - 表明当模式中第j个字符与主串中相应字符匹配失败时，在模式中需重新和主串中该字符进行比较的字符位置：即衡量模式串向后滑动的量
  - 计算

$$next[j]=\begin{cases}\max\{k \mid 1<k<j, \text{且} \text{“}p_1\cdots p_{k-1}\text{”}=\text{“}p_{j-k+1}\cdots p_{j-1}\text{”}\} & \text{当此集合非空时} \\ 0 & \text{当j=1时} \\ 1 & \text{其他情况}\end{cases}$$

（“$p_1\cdots p_{k-1}$”：从头开始的k-1个元素；“$p_{j-k+1}\cdots p_{j-1}$”：j前面的k-1个元素）

以“abab”为例

- 序号j = 1时，属于j == 1情况，next[j] = 0
- 序号j = 2时，‘a’的前后缀都为空集，属于其他情况，next[j] = 1
- 序号j = 3时，‘ab’的前缀为{a}，后缀为{b}，交集为空，属于其他情况，next[j] = 1
- 序号j = 4时，‘aba’的前缀为{a,ab}，后缀为{a,ba}，交集为{a}，k - 1 = 1（长度），next[j] = k = 2（长度+1）

王道和这里有差异，那个是部分匹配值

- 时间复杂度：O(m + n)

进一步优化

构造nextval数组

序号j	1	2	3	4	5	6
模式串	a	b	a	b	a	a
next[j]	0	1	1	2	3	4
nextval[j]	0	1	0	1	0	4

- 若模式串指向的位置的字符等于本身，可以直接赋值
- 举例
  - 5号的a中next数组指向3号位置，3号位置也是a故直接把3号的nextval数组值复制过去
  - 6号的a中next数组指向4号位置，两者字符不同，不能优化

代码

```
cin >> n >> p+1 >> m >> s+1;// char数组s是长文本，p是模式串，且从数组下标1开始存储
for(int i=2,j=0;i<=n;i++){  //求next数组
   while(j&&p[i]!=p[j+1]) j=ne[j];
   if(p[i]==p[j+1]) j++;
   ne[i]=j;
}
for(int i=1,j=0;i<=m;i++){  //匹配
   while(j&&s[i]!=p[j+1]) j=ne[j];
   if(s[i]==p[j+1]) j++;
   if(j==n){
      j=ne[j];
      //匹配成功
   }
}
```

红色边框为一级知识点：熟悉

橙色边框为二级知识点：掌握

By（b站）分享笔记的好人儿
上一章的数组属于本章，但我们按某道来
本章只有KMP一个考点，理解KMP算法思想，明白next数组干嘛用的，考试其实只需知道咋手算就行

重点考点：
1. KMP算法（在简单模式匹配算法基础上知道优化了什么，知道如何求next数组和nextVal数组即可）

Presented with xmind