4.2串的模式匹配

- **简单的模式匹配算法**
  - 子串的定位操作通常称为串的模式匹配，它求的是子串(常称模式串)在主串中的位置
  - 实现思想
    - 将主串中与模式串长度相同的子串搞出来，挨个与模式串对比
    - 当子串与模式串某个对应字符不匹配时，就立即放弃当前子串，转而检索下一个子串
  - 缺点：主串指针会出现回溯现象导致时间开销增加
  - 最坏时间复杂度：O(mm），其中n和m分别为主串和模式串的长度。
- **改进的模式匹配算法—KMP算法**
  - 字符串的前缀、后缀和部分匹配值
  - next 数组：当模式串的第 j 个字符匹配失败时，令模式串跳到 next[j] 再继续匹配
  - 获取串的匹配值　　'ababa'为例
    - 'a'的前缀和后缀都为空集,最长相等前后缀长度为0
    - 'ab'的前缀为{a),后缀为(b},(a}n（b}=O,最长相等前后缀长度为0
    - 'aba'的前缀为{a,ab},后缀为{a,ba),（a,ab）n{a,ba）={a），最长相等前后缀长度为1
    - 'abab'的前缀{a,ab,aba）n后缀{b,ab,bab}=(ab），最长相等前后缀长度为2
    - 'ababa'的前缀{a,ab,aba,abab)门后缀{a,ba,aba,baba}={a,aba},公共元素有两个,最长相等前后缀长度为3
    - 故字符串'ababa'的部分匹配值为00123
  - 当子串和模式串不匹配时，主串指针 i 不回溯，模式串指针 j=next[j]
  - 时间复杂度：O（m+n）
  - 优点：主串不会进行回溯
- **KMP算法的进一步优化**
  - KMP算法优化：当子串和模式串不匹配时j=nextval[j]　　通过构造nextval函数