山东大学网络空间安全学院

数据结构与算法 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202400460042 | 姓名：江李阳 | | 班级：1班 |
| 实验题目：**KMP模式匹配算法的实现** | | | |
| 实验学时：4 | | 实验日期：2025.3.21，20 | |
| 实验目的：利用KMP模式匹配算法在主串中查找指定模式串。如果主串包含该模式串，则返回该模式串在主串中的序号；否则返回-1。分别利用next数组和优化过的nextval数组实现。 | | | |
| 硬件环境：内存：16384MB 处理器：13th Gen intel(R) Core(TM) i5-13500HX (20 CPUs),~2.5GHz | | | |
| 软件环境：windows系统，visual studio | | | |
| 实验步骤与内容：  分析问题：题目要求使用KMP算法实现字符串的查找，查找到返回位置，否则返回-1。KMP算法能实现快速进行字符串的匹配查找在于利用next数组避免不必要的回溯，优化KMP也是在基于已有next数组的基础上进行优化，因此关键在于next数组的构建。  简化问题模板：构造next数组(nextval数组)->KMP模式匹配  解决问题：   1. 构造next数组：next数组的长度与模式串的长度一致，每一位的数代表当这一位与目标串匹配失败时，让模式串的哪一位重新来与目标串这一位来比较。next数组第i个数的值与前(0~i-1)位字符串的最大公前后缀的长度相同，第0位不存在这样的字符串，特殊记作-1，第1位前面字符串仅一位，不含最大公前后缀，对应的next数组的值为0。其余位的数值采用递归的方式求解，如求next[j],就比较pattern[j]与pattern[next[j-1]]是否相等，相等则让next[j]等于next[j-1]+1,否则就递归地去找符合条件的次长的len位字符串，让next[j]=len+1。 2. 在知道next数组的情况下就可以进行KMP模式匹配了，声明两个指针i,j分别指向target目标串和pattern模式串。在i, j都可以继续递增时一致执行:当target[i]和pattern[j]相等时，同时推进i和j;不相等时就利用next数组更新pattern中要拿来匹配的位置，其中由于next数组中第一位为-1，不能用来做字符串索引，因此要拿出来特殊处理。最终退出循环时通过判断pattern模式串成功匹配到了哪一位，确定返回匹配成功的在target串的对应起始位置，还是匹配失败返回-1。 3. 可以进一步优化next数组，构造nextval数组。利用next数组在匹配失败时回退，但如果pattern中匹配失败那一位的字符与回退后再次尝试的字符相等，则没有比较的必要，因为一点过匹配失败，将其回退到第一个不相等的字符，这时去匹配才有意义，基于这一点构造nextval数组。相比next在i++，j++后直接确定这一位的next的值，nextval进一步比较回退前后是否相等，不相等则与next数组操作一样，相等则令nextval的这一位j等于nextval的第i位。由于nextval的构建是从前往后的，所以不需要每一步递归寻找第一个不匹配的字符。   代码编写：  利用面向对象的编程思想将所有具体代码抽象到KMP类中，构造函数接受string类型的参数target和pattern，按照上面的分析思路实现next数组、nextval数组、字符串模式匹配。测试代码返回预期结果，成功解决问题。 | | | |
| 结论分析与体会： 通过KMP算法、next数组和优化的nextval数组的实现，我体会到了利用已知信息加快字符串匹配相较于暴力循环的高效性和巧妙性，实现next和nextval时利用递归的算法降低了暴力计算的时间复杂度，也让我锻炼了对递归算法的掌握使用。在具体写代码时，我遇到了使用整型的计数变量i、j与通过字符串的内置size()方法得到的字符串长度比较时，出现比较类型不匹配的报错，尤其在目标串与模式串比较时循环条件里还会出现严重错误的现象。经查阅资料发现size()方法返回的是无符号整型，由于next数组里有值-1，当i被赋值为-1，在循环判断条件里会被强转成无符号整型的一个很大的正数，导致循环错误退出，得到错误结果。通过显示类型转换可以解决这些问题。这个问题的出现和解决，让我明白了思路到具体实现还有很大差距，会遇到各种意想不到的结果。这次实验让我锻炼了代码编写和解决实际问题的能力。 | | | |