字符串匹配实验报告

Shuxin Chen, 2015013229

2017 年 4 月 10 日

1 实验内容

本实验使用三种字符串匹配算法，暴力算法、KMP 算法和 BM 算法，进行模式串与文本串的匹配操作。实验中要求有图形用户界面，能够使用户在简单的操作后得到匹配结果。

2 注意事项

本实验的测试环境为：

- 操作系统：macOS Sierra 10.12.3
- 处理器：2.5 GHz Intel Core i7
- 内存：16GB 1600MHz DDR3
- 编辑器：Qt Creator 3.5.1
- 编译环境：clang-800.0.38

因为涉及到文件的输入和输出操作，而 macOS 发布的 Qt 可执行文件实际上是一个包，因此要访问与可执行文件同目录下的输入输出文件，就必须要手动调整工作目录。但在其它的操作系统中就没有这个问题。源代码 main.cpp 中有一段手动调整工作目录的代码，如果希望在其它操作系统下编译、运行及发布，必须删除这段代码，否则输入输出文件无法定位。

3 实验过程

由于三种算法的代码在课堂上已经讲过，并且它们的实现都非常简单，因此在实验报告中不再赘述。这里只说一个没有讲过的地方，就是在 BM 算法中 *Osuff* 数组的求解方式。在讲课的 ppt 中，并没有说明如何求解 *Osuff* 数组，若根据定义暴力求解，时间复杂度为 $O(n^2)$，其中 n 为模式串的长度。在一般的情况下，模式串的长度 n 会远小于文本串的长度 m，但 n^2 和 m 的关系并不清楚。若数据规模非常大，n 为 10^6 数量级，m 为 10^9 数量级，此时 $O(n^2)$ 的时间不能忽略。此时必须要使用 $O(n)$ 的算法来求出 *Osuff* 数组。

我们求解的方法基于 KMP 算法中 π 数组的计算，这一部分的时间复杂度为 $O(n)$。在 KMP 算法中，我们求出的是前缀与后缀的关系，但 *Osuff* 数组表示的是后缀与后缀的关系，要使得 *Osuff* 与前缀相关，我们可以考虑把模式串翻转一下，设模式串为 P，文本串为 T，对于 $Osuff[i] = k$，此时表示的是 $P[1\,..\,k] = T[i\,..\,i+k-1]$，根据 π 数组的定义，我们有 $\pi[i+k-1] \geq k$。反推过来，若是有了 π 数组，那么 $\forall l \in \pi^*[s]$，有 $P[1\,..\,l] = T[s-l+1\,..\,s]$，即 $Osuff[s-l+1] \geq l$。那么我们对于每个位置 s，枚举 $\pi^*[s]$ 中的所有元素，设其中的一个元素为 l，我们用 l 值来更新 $Osuff[s-l+1]$，最终便得到了 *Osuff* 数组。因为 $pi^*[s]$ 在最坏的情况下是 $O(s)$ 的，因此算法的总时间复杂度仍然为 $O(n^2)$。

有没有优化的方法呢？我们可以从后往前枚举位置 s，假设有两个位置 s_0 和 s_1，且 $s_0 < s_1$，$s_0 - l_0 + 1 = s_1 - l_1 + 1 = p$，那么 $Osuff[p] = l_1$，因为 $l_0 < l_1$。因此，当某一个时刻落到之前已经有过 *Osuff* 值的位置时，它和之后的那些位置更新都是没有意义的，因此每个位置只会被更新一次，时间复杂度降低到 $O(n)$。

4 使用说明

bin 文件夹中存放着 macOS 下的可执行文件和一个输入文件，该文件夹的同目录下为源代码。打开可执行文件后便可以进行字符串的匹配操作，用户界面中有 4 个 Help 键，详细地说明了使用方法。