String Matching Experiment Report

洪方舟
2016013259
Email: hongfz16@163.com

2018 年 4 月 6 日

1. 实验目的

a. 实现 $Brute-Force, KMP, Boyer-Moore$ 三种字符串匹配算法
b. 要求字母表至少含有 26 个英文字母，0-9 数字以及若干英文标点符号
c. 实现方便测试的图形界面，程序可以输入 txt 文件做测试
d. 测试比较不同算法在不同的问题规模下的运行效率

2. 实验环境

操作系统：Windows 10
处理器：Intel Core i7-7700k CPU @ 4.20GHz × 8
编程语言：C++
IDE: Visual Studio

3. 实验方法

a. 编写三种算法，在命令行版本中将三者结果相比较来保证算法的正确性
b. 在命令行版本中随机生成不同规模的数据，记录运行时间
c. 使用 QT 编写方便测试的用户界面
d. 在 QT 界面中使用可以打开文件的控件来实现打开 txt 文件的要求

4. 实验结果

Table 1: 不同字符串匹配算法在不同数量级数据下所耗时间 (ms)

模式串长度	待匹配串长度	$Brute-Force$	KMP	BM
10^0	10^6	2	2	9
10^1	10^6	2	3	2
10^2	10^6	2	3	0
10^3	10^6	2	3	0
10^4	10^6	2	2	1
10^5	10^6	2	3	2
10^0	10^7	19	24	97
10^1	10^7	19	24	14
10^2	10^7	19	24	6
10^3	10^7	19	24	5
10^4	10^7	19	24	5
10^5	10^7	18	23	6
10^6	10^7	16	26	15
10^0	10^8	194	239	966
10^1	10^8	188	233	136
10^2	10^8	189	253	60
10^3	10^8	188	232	54
10^4	10^8	187	232	53
10^5	10^8	188	232	55
10^6	10^8	186	235	63
10^7	10^8	170	266	149

5. 分析与总结

a. 通过上面的运行时间可以发现，暴力方法与 KMP 算法的运行时间相差不大，总体来说，暴力算法的效率甚至高于 KMP 算法

b. BM 算法在特定的条件下运行效率较高：1. 模式串的长度大于 10^2；2. 待匹配串与模式串的长度比值大于 100；

c. 尽管暴力方法的时间复杂度为 $O(mn)$，KMP 算法的时间复杂度是线性的，但是在随机数据的情况下暴力方法很难达到最坏情况，因此甚至比线性算法运行效率高；另外，由于 KMP 算法模式串预处理时间复杂度为 $O(m)$，因此当模式串规模增长时，KMP 算法耗时也相应增长；

d. 由于模式串预处理耗时较多，BM 算法在模式串较短时的运行时间较长，无法体现其优势；当模式串长度增长，BM 算法的优势也就体现出来，在一定范围内，模式串越长，BM 算法效率越高；但是当待匹配串的长度与模式串长度之比接近 10 的时候，模式串预处理的时间劣势就显示了出来

e. 综上，不难发现，由于字符串的随机性，暴力方法的效率非常高，因此的日常使用中完全足够；KMP 算法虽然在理论上的时间复杂度优于暴力方法，但是在实际应用中效率很差，因此 KMP 算法仅仅具有理论意义，而不具备使用价值；BM 算法在一定的数据规模范围内效率最高，尤其是长模式串的情况下，应当选用 BM 算法。

6. 源代码及可执行文件说明

a. 控制台版本用于不同数据规模自动测试，和手动输入数据测试 (如果需要显示结果，请在源代码 *main.cpp* 文件中注释宏定义 *_NOT_SHOW_ANS_*)，使用方法见命令行中提示文字；可执行文件位于 *bin/Console*

b. GUI 版本用于打开外部 *txt* 文件进行测试，打开界面后点击 *Open* 按钮选择载入测试文件，在第二行的文本框中输入模式串，而后提供两种测试模式：*FindFirst* 按钮对应于找到测试文件中第一处出现模式串的位置，*FindAll* 按钮对应于找到测试文件中所有的模式串位置；在下方的文本框中会显示匹配结果，以及三种方法用时；可执行文件位于 *bin/GUI*