字符串匹配实验报告

姓名：覃果　学号： 2020012379　班级：软件02

2022 年 5 月 14 日

摘要

实现了 naive，kmp，bm 三种算法，分析了三种算法的特点，并在不同大小的数据上进行了实验，比较了实验结果。

1　实验环境

CPU: Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz
OS: ubuntu
Python interpreter: Python 3.8.0

2　算法分析

2.1　暴力匹配

2.1.1　算法描述

对text的每一个位置，逐一和pattern进行比较

2.1.2　算法分析

时间复杂度：

$O((n-m+1)m)$，其中 n 为 text 的长度，m 为 pattern 的长度。以下沿用此记号。

空间复杂度：

$O(1)$

算法评价：

算法实现简单，但复杂度太高

2.2 KMP算法

2.2.1 算法描述

先对 pattern 进行预处理，计算出 $\pi[0..m-1]$

在进行匹配的时候，利用计算出的 $\pi[0..m-1]$ 减小无用的匹配次数

2.2.2 算法分析

时间复杂度：

需要使用聚合分析的方法分析复杂度，最终结果预处理$\Theta(m)$，匹配 $\Theta(n)$

总复杂度 $\Theta(n+m)$

空间复杂度：

用于存储$\pi[0..m-1]$，$O(m)$

算法评价：

算法复杂度已经比较优秀，实现难度也不大，但是当字母表非常大（比如所有中文）时，算法的效率仍然有改进的空间

2.3 BM算法

2.3.1 算法描述

通过预处理，计算出 $bmBc$ 和 $bmGs$。

在匹配时利用这两个数组减少无用的匹配次数。

2.3.2 算法分析

时间复杂度：

通过良好的实现方法，用聚合分析的分析方法，可以使预处理的复杂度为 $O(m+|\Sigma|)$

最坏情况下匹配的复杂度为 $O(mn)$，最好情况下为 $\Omega(n/m)$，总复杂度 $O(mn+|\Sigma|)$

看上去复杂度比 KMP 高，但是在实际运行时，尤其当字母表很大的时候，BM算法的效率要比 KMP 更优秀

空间复杂度：

用于存储$bmBc$ 和 $bmGs$，$O(m+|\Sigma|)$

算法评价：

虽然最坏情况下的复杂度看上去不是很好，但实际运行的效率很高。

并且基于此算法的改进算法 Turbo-BM 实现了其他时间复杂度不变的情况下，将最坏情况的复杂度降低到了 $O(n)$。而空间复杂度只提高了常数级别。

这个算法被广泛使用于现实中的字符串匹配的应用

3 实验设计思路

比较在 (m, n) 为 (1e5, 10) (1e6, 10) (1e7, 10) (1e8, 10) 以及 (1e5, 100) (1e6, 100) (1e7, 100) (1e8, 100)时三种算法的运行时间，其中 pattern 和 text 都是随机生成

4 结果分析

实验结果如下

pattern	text	naive/s	kmp/s	bm/s
10	100000	0.0476	0.015698	0.007215
	1000000	0.369818	0.11568	0.057388
	10000000	3.440828	1.161855	0.571599
	100000000	34.93732	11.80853	5.670834
100	100000	0.044305	0.0152	0.001161
	1000000	0.342461	0.116132	0.008724
	10000000	3.487676	1.163078	0.09051
	100000000	35.47436	11.72028	0.876742

比较数据可以看到，当 pattern 的长度不变时，text 长度乘10，三个算法用时基本也乘10。这与他们$O(mn), O(m+n), O(mn)$的复杂度一致

当 text 长度不变时，pattern长度变为原来的10倍时，我们发现前两种算法所用时间都没有发生明显的变化

对于 kmp 算法，复杂度为 $O(m+n)$, 因为此时 n 远比 m 大，故时间不会有明显变化

对于 naive 算法，虽然复杂度为 $O(mn)$，但实际上的运行时间与失配时已经比较的长度有关，而本实验中数据都是随机产生，故每次不会比较 m 次就会失配，故时间也不会有明显的变化

而对于 bm 算法，我们发现 pattern 变长反而让花费的时间变少，这和该算法的最好情况下的复杂度为 $\Omega(n/m)$ 一致。

可以看出，bm算法的实际效率是最优秀的，在 1e8 长的 text 上，运行时间是其他两个算法的几十分之一

可视化数据如下：

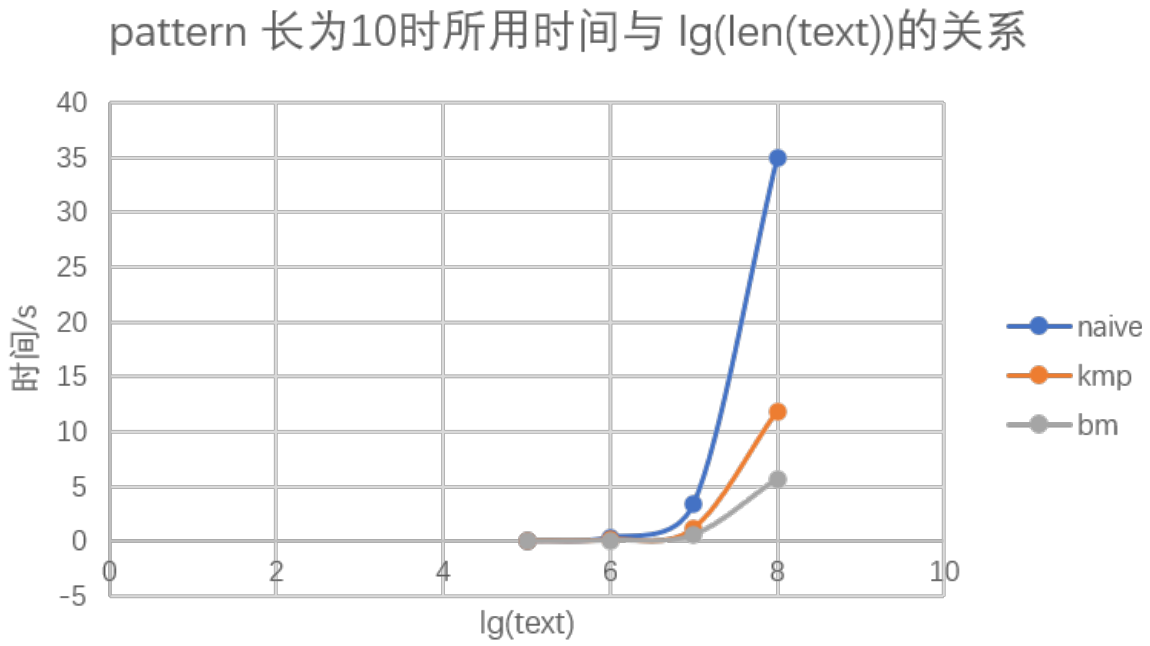

图 1: pattern 长为10

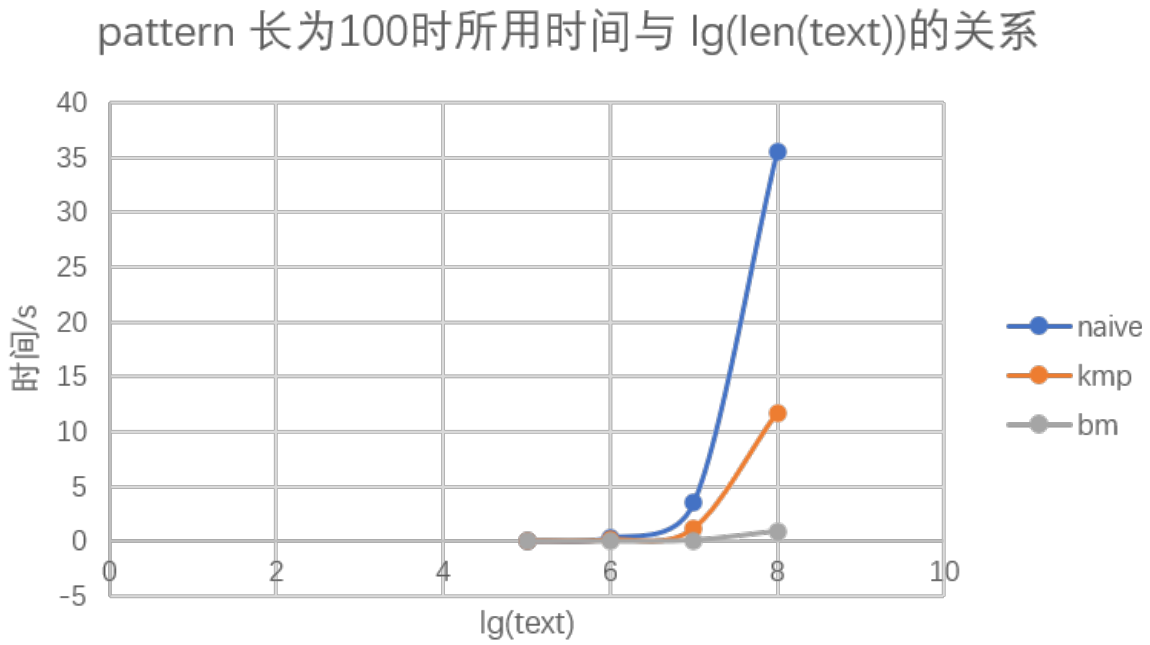

图 2: pattern 长为100