Algorithms: Assignment #6

Due on April 13, 2016

Zhaoyang Li (2014013432)

Contents

Problem 1

CLRS Exercises 32.4-8.

```
COMPUTE-TRANSITION-FUNCTION-FAST(P, Σ)
 1  π ← COMPUTE-PREFIX-FUNCTION(P)
 2  m ← P.length
 3  for each character a ∈ Σ
 4        do δ(0, a) = 0
 5  for q = 0 to m
 6        do for each character a ∈ Σ
 7              do δ(q, a) = δ(π[q], a)
 8                 if q ≠ m and P[q + 1] = a
 9                    then δ(q, a) = q + 1
10  return δ
```

Timing Lines 1 to 2 takes $\Theta(m)$ time. Lines 3 to 4 takes $\Theta(|\Sigma|)$ time. The double-loop starting from lines 5 takes $\Theta(m|\Sigma|)$ time. Therefore, total running time $T \in O(m|\Sigma|)$.

Correctness We'll prove that if $q = m$ or $P[q+1] \neq a$, then $\delta(q, a) = \delta(\pi[q], a)$.
Let's look at the definition of the prefix function π,

$$\pi[q] = \max\{k : k < q \text{ and } P_k \sqsupset P_q\}$$

Meanwhile, our transition function is supposed to meet the following equation

$$\delta(q, a) = \sigma(P_q a), \text{ where } \sigma(x) = \max\{k : P_k \sqsupset x\}$$

Now everything is ready, and all we need is an east wind.

$$\begin{aligned}
\delta(\pi[q], a) &= \delta(\max\{k : k < q \text{ and } P_k \sqsupset P_q\}, a) \\
&= \delta(\sigma(P_q), a) \\
&= \sigma(P_{\sigma(P_q)} a) \\
&= \sigma(P_q a) \\
&= \delta(q, a)
\end{aligned} \tag{1}$$

QED.

Problem 2

CLRS Problems 32-1. String matching based on repetition factors.

Problem 2 continued on next page...

a. The repetition factor is related to the prefix function introduced in KMP. As a matter of fact, we may prove the following lemma:

$$\forall i, \pi[i] = i - \frac{i}{\rho(P_i)}$$

COMPUTE-REPETITION-FACTOR(P)

```
1  let ρ[1...m] be a new array, in which
2  ρ[1] ← 1
3  π ← COMPUTE-PREFIX-FUNCTION(P)
4  for i = 2 to P.length
5        do if i − π[i] = π[i] / ρ[π[i]]
6              then ρ[k] ← ρ[π[i]] + 1
7              else  ρ[k] ← 1
8  return ρ
```

Running time $\Theta(m) + \Theta(m) = \Theta(m)$, where m is the length of the pattern string P.

b.

c.

Timing Consider line 12 where s is increased. Suppose this line is executed t times, each time $q = q_i$. Since s is increased from 0 to $n - m$, we know that

$$\sum_{i=1}^{t} \frac{q_i}{k} = n - m$$

Now look at line 8. Since q is increased by 1 at a time and decreased only on line 13, we know that lines 7-10 are executed no more than $\sum_{i=1}^{t} q_i$ times.
Each time we enters the loop body, either q is decreased or s is increased (otherwise, the loop becomes endless). Thus, times of the loop body being entered

$$\begin{aligned} \sum_{i=1}^{t} q_i + t &= k \times (n - m) + t \\ &= (1 + \rho^*(P)) \times (n - m) + t \\ &= O\left(\rho^*(P)n + m\right) \end{aligned} \tag{2}$$

Running time $T \in O\left(\rho^*(P)n + m\right)$.

Problem 3

Compare string matching algorithms Brute-Force, KMP and BM.

Problem 3 continued on next page...

Implementation and environment

Language The C++ programming language

IDE Visual Studio 2015 and Qt Creator 3.6.1

Compiling Desktop Qt 5.6.0, MSVC2015 64bit Release

OS Windows 10 Education, Build 10586

Hardware Intel Core i7-6600U @ 2.60GHz, 16GB LPDDR3, ThinkPad X1 Carbon 20FB

Verification of correctness

Code written in reference[a] to a trust-worthy academic institute.

Correctness of the code is verified by comparing the respective results from the three algorithms for consistency. All test samples are passed.

[a] http://www-igm.univ-mlv.fr/ lecroq/string/

Timing

There are several factors that may influence running-time of string matching algorithms, including

- text style (size of set of possible characters, repetition factor, etc)
- length of pattern string
- length of text string
- ...

So we need to do several sets of experiments here.
In order to be more precise, all timing are done by repeating the algorithm several times and calculating an average running-time.

Text style?

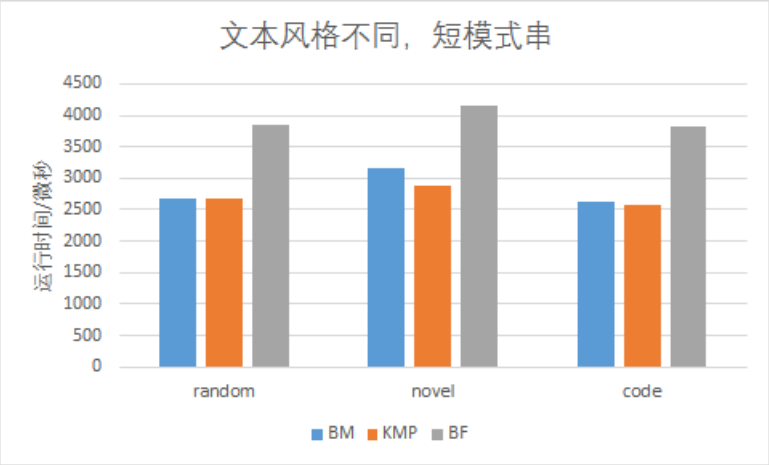
Many of my classmates use randomly generated text as their string to be searched in, as far as I know. I doubt if it's a reasonable thing to do. So I managed to collect three blocks of text which are in different styles -

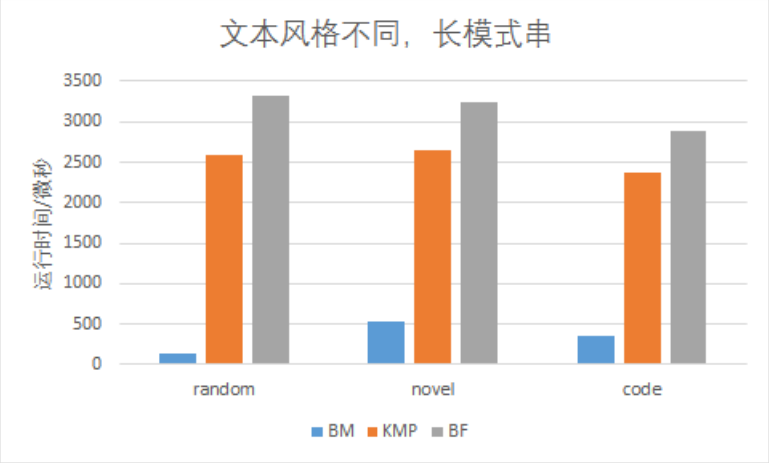
- C source code of the XV6 operating system
- full text of an English novel The Wonderful Wizard of Oz
- randomly generated text, full of CAPITALIZED English letters

each of which are made to have an approximate length of 10^6, by performing Ctrl+C and Ctrl+V if necessary.

Results illustrated as below.

Problem 3 continued on next page...



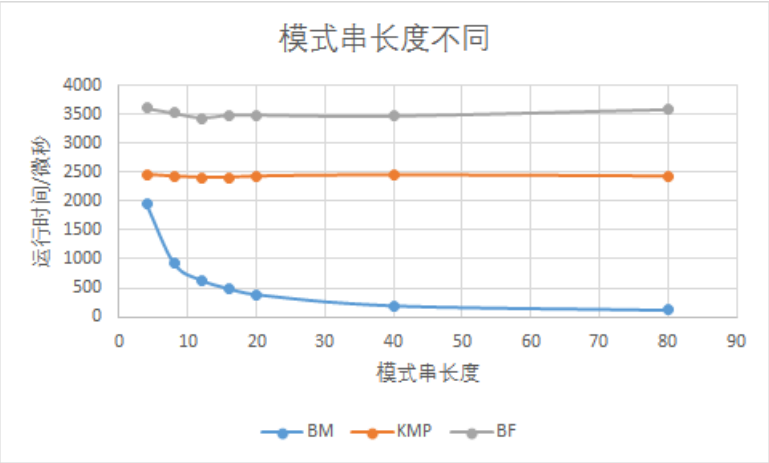


It can be concluded that style of the text (randomly generated or not) has no affect on timing comparison. In another word, the algorithm that takes the longest time always takes the longest time. Thus, it's reasonable to use randomly generated text to represent general cases. That's exactly what I did hereinafter.

Also, BM has more advantage with longer patterns. I did more experiments to confirm this.

Pattern length

For BF and KMP, pattern length has little affect on running-time of the algorithm. However, as length of pattern grows, running-time of BM reduces significantly.



Text length

Finally, let's take a look at how running-time grows as length of the text string grows.

Perfect linearity, as expected. BF > KMP > BM, as expected.