

Rabin-Karp and Finite Automaton Algorithms

Advanced Algorithms (CSE 794)

Instructor: Tamal K. Dey

Scribe: Zhixue Lu *

May 26, 2009

1 Introduction

String-matching is a most fundamental problem in the domain of text processing in Computer Science, as well as in DNA sequence matching in Molecular Biology.

There are many approaches for this problem, of which, Rabin-Karp algorithm treats a string as a number, and gives a complexity of $\Theta(m)$ preprocessing time together with $O(mn)$ matching time; while in the finite automaton solution, a deterministic automaton is built and searching is done based on this automaton, which asks $O(m|\Sigma|)$ preprocessing time and $\Theta(n)$ matching time; Finally, Knuth-Morris-Pratt algorithm gives a similar idea as in Automaton, but with a better performance of $\Theta(m)$ in preprocessing and $\Theta(n)$ in searching.

2 Notations

ε denotes empty string.

$|x|$ denotes the length of string x .

Σ denotes the alphabet.

Σ^* denotes all possible strings made of the alphabet.

$w \sqsubset x$ means string w is a prefix of string x , i.e. for some $y \in \Sigma^*$, $wy = x$.

$w \sqsupset x$ means string w is a suffix of string x , i.e. for some $y \in \Sigma^*$, $yw = x$.

$T[1 \dots n]$ denotes a text which is a string of length n .

$P[1 \dots m]$ denotes a pattern of length m .

We call P occurs with shift s in T , if $T[s + 1 \dots s + m] = P[1 \dots m]$, for $s \in 0, 1 \dots n - m$.

Lemma 1. *Suppose x, y, z are strings s.t. $x \sqsubset z$ and $y \sqsupset z$, then:*

(i) if $|x| \leq |y|$, $x \sqsubset y$.

(ii) if $|x| \geq |y|$, $y \sqsupset x$.

(iii) if $|x| = |y|$, $x = y$.

*Department of Computer and Information Sciences, The Ohio State University, Columbus, Ohio, USA.

3 Rabin-Karp Algorithm

Rabin and Karp have proposed a string-matching algorithm that performs well in practice and that also generalizes to other algorithms for related problems, such as two-dimensional pattern matching. The Rabin-Karp algorithm requires a $\Theta(m)$ preprocessing time and a $O(mn)$ worst-case running time.

3.1 Description

Rabin-Karp algorithm maps alphabet to numbers, such that $|Alphabet| = |ThebaseofNumber|$. Suppose $|Alphabet| = 10$, we then compute the value of P as:

$$p = P[m] + 10(P[m - 1] + 10(P[m - 2] + \dots + 10(P[2] + 10P[1]) \dots)).$$

This can be done in $\Theta(m)$ time. Suppose we have t_s , which is the value of $T[s + 1]T[s + 2] \dots T[s + m + 1]$, we can compute t_{s+1} as:

$$t_{s+1} = 10(t_s - 10^{m-1}T[s + 1]) + T[s + m + 1].$$

If the pattern is short, t_{s+1} can be computed in constant time from t_s . However, if the pattern is very long, this assumption will become unreasonable. Fortunately, there is a simple cure for this problem: compute p and the t_s 's modulo a suitable modulus q. In this case, t_{s+1} will be computed as:

$$t_{s+1} = (d(t_s - d^{m-1} \pmod{q} T[s + 1]) + T[s + m + 1]) \pmod{q}$$

If at some point, $t_s \pmod{q} = p \pmod{q}$, the algorithm will go and check each bit to make sure they are all equal.

3.2 Algorithm

```
RABIN-KARP( $T, P, d, q$ )
 $n \leftarrow |T|$ 
 $m \leftarrow |P|$ 
 $h \leftarrow d^{m-1} \bmod q$ 
 $p \leftarrow 0$ 
 $t_0 \leftarrow 0$ 
for  $i = 1$  to  $m$ 
     $p \leftarrow (dp + P[i]) \bmod q$ 
     $t_0 \leftarrow (dt_0 + T[i]) \bmod q$ 
endfor
for  $s = 0$  to  $n - m$ 
    if  $p = t_s$  then:
        if  $p[1 \dots m] = T[s + 1 \dots s + m]$  then:
            output "matching with shift  $s$ "
        endif
    endif
    if  $s < n - m$  then:
         $t_{s+1} \leftarrow (d(t_s - T[s + 1]h) + T[s + m + 1]) \bmod q$ 
    endif
endfor
```

4 Finite Automaton

4.1 Definition

A finite automaton M is a 5-tuple $(Q, q_0, A, \Sigma, \delta)$, where

- Q is a finite set of states,
- $q_0 \in Q$ is the start state,
- $A \subseteq Q$ is a distinguished set of accepting states,
- Σ is a finite input alphabet,
- δ is a function from $Q \times \Sigma$ into Q , called the transition function of M .

A finite automaton M induces a function ϕ , called the final-state function, from Σ^* to Q such that $\phi(w)$ is the state in M where the machine ends up after scanning the string w . Thus, M accepts a string w if and only if ϕ is defined by the recursive relation:

$$\phi(\varepsilon) = q_0,$$

$$\phi(wa) = \delta(\phi(w), a) \text{ for } w \in \Sigma^*, a \in \Sigma.$$

We also define $\sigma(x) = \max\{k | P_k \sqsupseteq x\}$ as a suffix function, which is a mapping from Σ^* to $0, 1, \dots, m$ such that $\sigma(x)$ is the length of the longest prefix of P that is a suffix of x . For instance, let $p = (abab)$, then: $\sigma(aabab) = 4$, $\sigma(abaab) = 2$, $\sigma(abaaa) = 1$, $\sigma(ababb) = 0$.

Then the string-matching automaton corresponds to a given pattern $P[1 \dots m]$ as follows:

- The state set Q is $0, 1, \dots, m$. The start state q_0 is state 0, and state m is the only accepting state.
- The transition function δ is defined by $\delta(q, a) = \sigma(P_q a)$.

Theorem 2. $\phi(T_i) = \sigma(T_i)$

Proof. The proof is by induction on i . For $i = 0$, the theorem is trivially true, since $T_0 = \varepsilon$. Thus, $\phi(T_0) = 0 = \sigma(T_0)$.

Now, assume that $\phi(T_i) = \sigma(T_i)$. let q denote $\phi(T_i)$, and let a denote $T[i + 1]$. then,

$$\phi(T_{i+1}) = \phi(T_i a) = \delta(\phi(T_i), a) = \delta(q, a) = \sigma(T_i a) = \sigma(T_{i+1})$$

□

4.2 Algorithm

```

FINITE-AUTOMATON( $T, \delta, m$ )
 $n \leftarrow |T|$ 
 $q \leftarrow 0$ 
for  $i = 1$  to  $n$ , do
     $q \leftarrow \delta(q, T[i])$ 
    if  $q = m$  then:
        output "matching with shift  $i=m$ "
    endif
endfor

```