1st Partial Project

Computational Mathematics

September 5, 2016

1 Introduction

String-matching is an important problem in the domain of text processing. String-matching algorithms are basic components used in implementations of practical software existing in most operating systems. They also play an important role in theoretical computer science by providing challenging problems.

Many string-matching algorithms build a finite automaton simple machine for processing information that scans the text string T for all occurrences of pattern P. These string-matching automata are very efficient: they examine each text character exactly once, taking constant time per text character.

One of these algorithms builds the minimal deterministic automaton[1] recognizing the language given by the pattern. Next, the text string is scanned using such an automaton, obtaining the number of times the pattern is found in the text [1].

2 Algorithm Description

This algorithm consists of two steps: (1) Building a DFA using P and (2) Input the text T into such a DFA to find the number occurrences in the DFA. In what follows, P is denoted by $x = x[1 \dots m]$ with length equal to m. T is denoted by $y = y[1 \dots n]$ with length equal to n. Both strings are built over the alphabet Σ.

2.1 Building the DFA

Searching the pattern x with an automaton consists first in building the minimal Deterministic Finite Automaton (DFA) $A(x)$ that recognizes the language $\Sigma^* x$. The DFA $A(x) = (Q, \delta, q_0, F)$ is defined as follows:

- $Q = \{q_0, q_1, \dots, q_m\}$
- $q_0 = q_0$
- $F = \{q_m\}$

[1]DFA minimization is the task of transforming a given deterministic finite automaton (DFA) into an equivalent DFA that has a minimum number of states

- δ is build using the following algorithm.

Listing 1: Algorithm 1

```
COMPUTE−TRANSITION−FUNCTION(P,Σ)
m = P.length
for q = 0 to m (for each state)
  for each character a ∈ Σ
     k = min(m+1, q+2)
   repeat
     k = k−1 (where 1 ≤ k ≤ m+1)
   until P_k includes P_q a as suffix
   δ(q, a) = k
return δ
```

2.2 DFA Matcher

Once the DFA $A(x)$ is built, searching for a word x in a text y consists in parsing the text with the DFA beginning with the initial state q_0. Each time the terminal state is found an occurrence of x is reported.

Listing 2: Algorithm 2

```
FINITE−AUTOMATON−MATCHER (T, δ, m)
n = T.length
q = 0
for i = 1 to n
  q = δ(q,T[i])
  if q == m
   print ‘‘Pattern occurs ’’
```

3 Example

Lets build A(x) for the pattern $P = abba$. The table below shows the execution of Algorithm 1.

q	symbol	k	p_k	p_q	p_q	p_k	p_k is suffix of p_q?
0	a	1	p_1	p_0a	ϵa	a	δ(0,a)=1
0	b	1	p_1	p_0b	ϵb	a	NO
		0	p_0	p_0b	ϵb	ϵ	δ(0,b)=0
1	a	2	p_2	p_1a	aa	ab	NO
		1	p_1	p_1a	aa	a	δ(1,a)=1
1	b	2	p_2	p_1b	ab	ab	δ(1,b)=2
2	a	3	p_3	p_2a	aba	abb	NO
		2	p_2	p_2a	aba	ab	NO
		1	p_1	p_2a	aba	a	δ(2,a)=1
2	b	3	p_3	p_2b	abb	abb	δ(2,b)=3
3	a	4	p_4	p_3a	abba	abba	δ(3,a)=4
3	b	4	p_4	p_3b	abbb	abba	NO
		3	p_3	p_3b	abbb	abb	NO
		2	p_2	p_3b	abbb	ab	NO
		1	p_1	p_3b	abbb	a	NO
		0	p_0	p_3b	abbb	ϵ	δ(3,b)=0
4	a	4	p_4	p_4a	abbaa	abba	NO
		3	p_3	p_4a	abbaa	abb	NO
		2	p_2	p_4a	abbaa	ab	NO
		1	p_1	p_4a	abbaa	a	δ(4,a)=1
4	b	4	p_4	p_4b	abbab	abba	NO
		3	p_3	p_4b	abbab	abb	NO
		2	p_2	p_4b	abbab	ab	δ(4,b)=2

After this process, DFA is ready and will look like Figure 1.

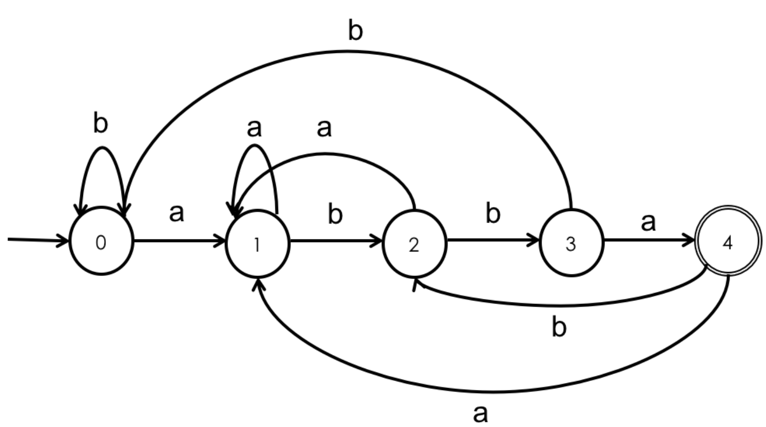

Figure 1: DFA that recognizes the *abba* pattern

With the DFA $A(x)$, we can now verify how many times the pattern P occurs in the text $T = baabbabbaaba$ by registering the times the final state of $A(x)$ is visited.

4 Deliverables

As part of this project, you and your team have to submit the following:

1. The executable file of your implementation using the programming language you like.

2. The source code of such an implementation.

3. The following sets of testing cases:

   - Given pattern $P = abba$ find how many times is present in the text $T = baabbabbaaba$
   - Given pattern $P = aabab$ find how many times is present in the text $T = aaababaabaababaab$

If you want more information about how the algorithm works, you can watch the following video https://youtu.be/nNb9lu5Hvio

The solution has to be presented by the complete team at a date and time previously agreed on. Without this, your project cannot be marked. You should present before **September 14th**.

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. String matching. In *Introduction to Algorithms*, chapter 32, pages 985–1013. MIT Press, Cambridge, Massachusetts, 1990.