String matching with preprocessing of text and pattern

Moni Naor
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120, USA

Abstract

We consider the well known string matching problem where a text and pattern are given and the problem is to determine if the pattern appears as a substring in the text. The setting we investigate is where the pattern and the text are preprocessed separately at two different sites. At some point in time the two sites wish to determine if the pattern appears in the text (this is the on-line stage). We provide preprocessing and on-line algorithms such that the preprocessing algorithm runs in linear time and requires linear storage and the on-line complexity is logarithmic in the text. We also give an application of the algorithm to parallel data compression, and show how to implement the Lempel Ziv algorithm in logarithmic time with linear number of processors.

1 Introduction

The string matching problem is: given a text $S$ and a pattern $P$ over some alphabet, does $P$ appear as a substring of $S$. This is one of the most studied problems in computer science and a large body of literature is devoted to it. (See e.g. the survey by Aho [Ah] or the proceedings edited by Apostolico and Galil [AG].)

Most pattern matching algorithms fall into one of two categories: either the text is first preprocessed and then, given the pattern, the work is simple and proportional to the length of the pattern (for instance [We, Mc]); or, the pattern is first preprocessed and then, given
the text, it is simple to determine if the text contains the pattern as a substring (for instance [KMP]). In this work we combine the two approaches and show what can be done when both pattern and text are preprocessed.

Suppose that a system consists of two sites and one site is given a text and another is given a pattern. They have time to preprocess their inputs and store the results (this is the preprocessing stage). At some later time they wish to determine whether the pattern appears in the text (the on-line stage). What we are aiming for is a procedure for the preprocessing which is not more expensive than the traditional algorithms in terms of time and memory requirements in the preprocessing stage, but whose on-line complexity is considerably smaller than the length of the text or the pattern.

This problem is applicable in several scenarios. It fits naturally in a distributed environment where the text and the pattern reside in two different locations. Another scenario is where there are many texts and many patterns, and we wish to know for some pairs of (text, pattern) whether the pattern appears in the text. We need not know in advance which pattern should be searched for in which text. (Searching for all patterns in all texts might yield too much information, i.e. the length of the output might be larger than the complexity we are aiming for.) Using the methods described in this paper, after the initial preprocessing we can easily determine for any given pair of pattern and text whether the pattern appears in the text. Another application is outlined in Section 3, where we show how to use the solution of this problem in order to implement in parallel with a linear number of processors the data compression method of Lempel-Ziv [ZL1].

Compare the scenario considered in this paper with the communication complexity one: in a communication complexity setting only the number of bits exchanged between the parties are counted, and the complexity of their internal computation is completely disregarded. Nevertheless, impossibility results for our model can be derived from the communication complexity model. In particular, we can deduce that our algorithms must be probabilistic, if we insist that the on-line complexity be smaller than the length of the pattern: even the simpler string equality problem, i.e. in the case where the pattern and the text are of the same length, the deterministic communication complexity is linear [Yao].

In this paper we provide an algorithm where the preprocessing complexity is linear and the on-line complexity is logarithmic in the length of the text. The probability of error can be made inverse polynomially small within those time bounds. We assume that the system has a globally known short random string. The inputs and the shared random string are