A String Matching Algorithm Fast on the Average

Extended Abstract[x)]

by

Beate Commentz-Walter
FB 10 - Informatik
Universitaet des Saarlandes

currently:
IBM-Germany
Scientific Center Heidelberg
Tiergartenstrasse 15
D-6900 Heidelberg

0. Introduction

In many information retrieval and text-editing applications it is necessary to be able to locate quickly some or all occurrences of user-specified words or phrases in one or several arbitrary text strings. Specifically , we consider retrieval from unformatted data, for example, a libary data base where there is for each book a record containing the signature, title, and abstract of book. Each such record we call a document. A user of the data base specifies one or several words or phrases, so called keywords, describing the information sought. The answer will be the documents which contain all or some of the user specified keywords. It takes too much time to scan each document of the data base for every user seperately. Therefore, we introduce a sort of secondary index (compare Scheck lit /12/) containing keyword fragments. Searching the index with the user specified keywords yields a superset of the documents required. This

---

x) For detailed version compare lit (4). The work reported here was done at the Heidelberg Scientific Center of IBM-Germany. It is part of a project dealing with subjects like Automatic Indexing, Clustering, and retrieval structures of unformatted data base.

superset contains documents where the fragments match but the keywords do not. These documents we want to reject. Therefore, we scan the documents of the superset for the user specified keywords.

Aho, Corasick lit /2/ describe an efficient algorithm doing this job. Their algorithm first preprocesses the keywords in time linear in the "total lengths of the keywords i. e. in the sum of the length of the keywords. Then their algorithm searches for the keyword occurrences in the document in time linear in document length (worst case).

The idea of this algorithm is based on the ideas of the Knuth-Morris-Pratt algorithm lit /10/ and those of finite state machines.

If there is only one keyword to search for in some document, Boyer, Moore lit /3/ give an algorithm the preprocessing phase of which also runs linearly and the search phase of which is faster on the average than Aho-Corasick's algorithm.

In the case of large alphabets the Boyer-Moore algorithm takes time about |D|/|W| on the average to search for all occurrences of the keyword W in document D (where |S| denotes the length of string S).

With the modification due to Galil lit /7/ the search phase of the Boyer-Moore algorithm behaves linearly in the document length even in the worst case. This is proved by Knuth, Morris, Pratt lit /10/, Guibas, Odlyzko lit /8/, and Galil lit /7/.

We give an algorithm B for a set of keywords. Its search phase behaves similar to the Boyer-Moore algorithm, sublinear on the average. It does not maintain linear search time for the worst case. Modification to B yield algorithm B1, which does maintain linear search time. But for practical purposes algorithm B is more useful. The overhead of algorithm B1 is very high.