Efficient Pattern-Matching with Don't Cares

Adam Kalai*

Abstract

We present a randomized algorithm for the *string matching with don't cares* problem. Based on the simple fingerprint method of Karp and Rabin for ordinary string matching [4], our algorithm runs in time $O(n \log m)$ for a text of length n and a pattern of length m and is simpler and slightly faster than the previous algorithms [3, 5, 1].

1 Introduction.

We extend the simple randomized fingerprinting algorithm of Karp and Rabin [4] to the problem of string matching with don't cares. Our algorithm uses a single, simple convolution. This is optimal in the sense that the string matching with don't cares problem is at least as hard as the boolean convolution problem [6]. Thus, to improve our run time of $O(n \log m)$ on text of length n and pattern of length m, one would have to improve on the Fast Fourier Transform.

Fischer and Paterson's algorithm [1] runs in time $O(n \log m \log |\Sigma|)$[1]. Since their deterministic algorithm in 1974, the only improvements were by Muthukrishan and Palem [5], who reduced the constant factor, and Indyk [3], who gave a randomized algorithm that also involved convolutions, running in time $O(n \log n)$. In addition to the small time improvement[2] ($O(n \log m)$ over $O(n \log n)$), we hope the conceptually simple algorithm is also of interest. Determining the complexity of this problem was on a list of Galil's [2] as an open problem.

2 Problem and Solution

Let text $X = x_1 x_2 \cdots x_n$, with $x_i \in \Sigma = \{1, 2, \ldots, s\}$, and a pattern $Y = y_1 y_2 \cdots y_m$, with $y_i \in \Sigma \cup \{*\}$. We think of $*$ as a "don't care." Then, we say that $X(j) = x_j x_{j+1} \cdots x_{j-1+m}$ *matches* Y if

$$x_{j-1+i} = y_i \text{ or } y_i = *, \text{ for } 1 \le i \le m.$$

The following algorithm finds all positions j where $X(j)$ matches Y.

Input: Text $X = x_1 \ldots x_n$, pattern $Y = y_1 \ldots y_m$, and parameter N.

1. For $1 \le i \le m$, set
$$r_i = \begin{cases} 0 \text{ if } y_i = * \\ \text{random from } \{1, 2, \ldots, N\} \text{ otherwise.} \end{cases}$$

2. Compute $t = \sum_{i=1}^{m} y_i r_i$.

3. For $1 \le j \le n - m$, compute $s(j) = \sum_{i=1}^{m} x_{j-1+i} r_i$ using FFT.

4. Output MATCH for those j's where $s(j) = t$.

If $X(j)$ matches Y, then the above algorithm will certainly output MATCH. However, if $X(j)$ does not match Y, then for some i, $y_i \neq x_{j-1+i}$ and $y_i \neq *$. Fixing the remaining variables, the equation $s(j) = t$ then has a unique solution for r_i, which we will choose with probability at most $1/N$. For say $N = n^2$, we would have at most a probability $1/n$ of having any false matches.

Runtime is $O(n \log m)$ assuming we can do calculations of the size $\log(N \Sigma n)$ in constant time, since Fast Fourier Transforms can be done in time $O(n \log m)$. All calculations could also be done modulo a prime $p \ge N$.

3 Don't Cares in Pattern and Text

The above algorithm can be extended to handle the case of don't cares in the text X as well. In this case, a * in the text matches any symbol in the pattern. We use a second convolution to compute what $s(j)$ should be for a match, which depends on j,

$$t(j) = \sum_{1 \le i \le m | x_{j-1+i} \neq *} y_i r_i.$$

This is an FFT of the vectors $\langle y_i r_i \rangle_{i=1}^{m}$ with $\langle \delta_{x_j \neq *} \rangle_{j=1}^{n}$. ($\delta_{x_j \neq *}$ is 1 if $x_j \neq *$ and 0 if $x_j = *$.) If we replace $*$' s with 0 's for step 3, then simple calculation shows we will certainly have $s(j) = t(j)$ if $X(j)$ matches Y. And for the same reason as above, we will err with probability at most $1/N$.

Acknowledgements. I thank Piotr Indyk and the anonymous SODA referees for helpful comments.

*M.I.T. (akalai@mit.edu)

[1] Like Indyk [3], we use the RAM model. The run time given by Fischer and Patterson [1], $O(n \log^2 m \log \log m \log |\Sigma|)$, is slightly higher because they do not use the RAM model.

[2] In personal communications, Indyk has indicated that his algorithm may also be improved to $O(n \log m)$.

References

[1] M.J. Fischer and M.S. Paterson, *String Matching and other Products*, in Complexity of Computation, R.M. Karp (editor), SIAM AMS Proceedings, 7, pp. 113-125, 1974.

[2] Z. Galil, *Open problems in stringology*, Combinatorial problems on words, NATO ASI Series, Vol. F12, pp. 1-8, 1985.

[3] P. Indyk, *Faster algorithms for string matching problems: matching the convolution bound*, in Proceedings of the 39th Symposium on Foundations of Computer Science, 1998.

[4] R.M. Karp and M.O. Rabin, *Efficient randomized pattern-matching algorithms*, IBM Journal of Research and Development, vol. 31, no 2., pp. 249-260, 1987.

[5] S. Muthukrishan and K. Palem, *Non-standard Stringology: Algorithms and Complexity*, in Proceedings of the 26th Symposium on the Theory of Computing, 1994.

[6] S. Muthukrishnan, H. Ramesh, *String Matching under a General Matching Relation*, Information and Computation, vol. 122, no. 1, pp. 140-148, Oct 1995.