Searching patterns: combinatorics and probability

Mireille Régnier
INRIA-Rocquencourt

July 8, 1996

[summary by Pierre Nicodème]

Abstract

We formally define a class of sequential pattern matching algorithms that includes all variations of the Morris-Pratt algorithm. We prove for the worst case and the average case the existence of a complexity bound which is a linear function of the text string length for the Morris-Pratt algorithm, using the *Subadditive Ergodic Theorem*. We establish some structural property of Morris-Pratt-like algorithms, proving the existence of "unavoidable positions" where the algorithm must stop to compare. We compute also the complexity of the Boyer-Moore algorithm.

1. Sequential pattern matching algorithms

1.1. Basic Definitions. Throughout we write $\mathbf{p}$ and $\mathbf{t}$ for the pattern and the text which are of lengths m and n, respectively. The ith character of the pattern $\mathbf{p}$ (text $\mathbf{t}$) is denoted as $\mathbf{p}[i]$ ($\mathbf{t}[i]$), and by $\mathbf{t}_i^j$ we denote the substring of $\mathbf{t}$ starting at position i and ending at position j, that is $\mathbf{t}_i^j = \mathbf{t}[i]\mathbf{t}[i+1]\cdots\mathbf{t}[j]$. We also assume that the length m of a given pattern $\mathbf{p}$ does not vary with the text length n.

We want to investigate the complexity of string matching algorithms [2]. We define it formally as follows.

DEFINITION 1 (COMPLEXITY).

(1) For any string matching algorithm that runs on a given text $\mathbf{t}$ and a given pattern $\mathbf{p}$, let $M(l,k) = 1$ if the lth symbol $\mathbf{t}[l]$ of the text is compared by the algorithm to the kth symbol $\mathbf{p}[k]$ of the pattern. We assume in the following that this comparison is performed at most once.

(2) For a given pattern matching algorithm, a partial complexity function $c_{r,s}$ is defined as

$$c_{r,s}(\mathbf{t},\mathbf{p}) = \sum_{l\in[r,s],k\in[1,m]} M[l,k]$$

where $1 \le r < s \le n$. For $r = 1$ and $s = n$ the function $c_{1,n} := c_n$ is simply called the *complexity* of the algorithm. If either the pattern or the text is a realization of a random sequence, then we denote the complexity by a capital letter, that is, we write C_n instead of c_n.

An Alignment Position AP is a position of the text which is aligned with the first character of the pattern during the processing of the algorithm, and such that, with the corresponding alignment, at least one character of the pattern is compared with the text.

DEFINITION 2. A string searching algorithm is said:

(1) *semi-sequential* if the text is scanned from left to right;
(2) *strongly semi-sequential* if the order of text-pattern comparisons actually performed by the algorithm defines a non-decreasing sequence of text positions (l_i) and if the sequence of alignment positions is non-decreasing.
(3) *sequential* (respectively *strongly sequential*) if they satisfy, additionally for any $k > 1$

$$M[l,k] = 1 \Rightarrow \mathbf{t}_{l-(k-1)}^{l-1} = \mathbf{p}_1^{k-1}.$$

Note that condition (3) forbids unnecessary comparisons.

EXAMPLE (NAIVE OR BRUTE FORCE ALGORITHM). The simplest string searching algorithm is the naive one. All text positions are alignment positions. For a given one, say AP, the text is scanned until the pattern is found or a mismatch occurs. Then, $AP + 1$ is chosen as the next alignment position and the process is repeated.

This algorithm is sequential but not strongly sequential. Condition (2) is violated after any mismatch on a alignment position l with parameter $k \geq 3$, as comparison $(l+1, 1)$ occurs after $(l+1, 2)$ and $(l+2, 3)$.

EXAMPLE (MORRIS-PRATT-LIKE ALGORITHMS [3]). Morris-Pratt like algorithms are strongly sequential; when a mismatch is found, they shift the pattern by the largest periodicity of the prefix of the pattern examined at the corresponding alignment position. The Knuth-Morris-Pratt variant remembers the last question concerning the mismatch position of the text and does not ask it again; the Simon variant remembers all the questions at the mismatch position, and does not ask them again. The efficiency of these algorithms is slightly better as the number of remembered questions increases.

It was already noted [3] that after a mismatch occurs when comparing $\mathbf{t}[l]$ with $\mathbf{p}[k]$, some alignment positions in $[l+1, \ldots, l+k-1]$ can be disregarded without further text-pattern comparisons. Namely, the ones that satisfy $\mathbf{t}_{l+i}^{l+k-1} \neq \mathbf{p}_1^{k-i}$, or, equivalently, $\mathbf{p}_{1+i}^{k} \neq \mathbf{p}_1^{k-i}$, and the set of such i can be known by a preprocessing of $\mathbf{p}$. Other i define the "surviving candidates", and choosing the next alignment position among the surviving candidates is enough to *ensure* that condition (2) in Definition 2 holds.

EXAMPLE (ILLUSTRATION TO DEFINITION 2). Let $\mathbf{p} = abacabacabab$ and $\mathbf{t} = abacabacabaaa$. The first mismatch occurs for $M(12,12)$. The comparisons performed from that point are:

1. *Morris-Pratt variant:* $(12,12);(12,8);(12,4);(12,2);(12,1);(13,2);(13,1)$, where the text character is compared in turn with pattern characters (b,c,c,b,a,b,a) with the alignment positions $(1,5,9,11,12,12,13)$.
2. *Knuth-Morris-Pratt variant:* $(12,12);(12,8);(12,2);(12,1);(13,2);(13,1)$, where the text character is compared in turn with pattern characters (b,c,b,a,b,a) with the alignment positions $(1,5,11,12,12,13)$.
3. *Simon variant:* $(12,12);(12,8);(12,1);(13,2);(13,1)$, where the text character is compared in turn with pattern characters (b,c,a,b,a) with the alignment positions $(1,5,12,12,13)$.

Positions 1, 5 and 12 are unavoidable for all these Morris-Pratt-like algorithms.

DEFINITION 3. For a given a pattern $\mathbf{p}$, a position i in the text $\mathbf{t}$ is an *unavoidable alignment position* for an algorithm if for any r, l such that $r \leq i$ and $l \geq i + m$, the position i is an alignment position when the algorithm is run on $\mathbf{t}_r^l$.

THEOREM 1. [7] *Given a pattern* $\mathbf{p}$ *and a text* $\mathbf{t}$*, all strongly sequential algorithms have the same set of unavoidable alignment positions* $U = \bigcup_{l=1}^{n} \{U_l\}$*, where*

$$U_l = \min\{\min_{1 \le k \le l} \{\mathbf{t}_k^l \preceq \mathbf{p}\}, l+1\}$$

and $\mathbf{t}_k^l \preceq \mathbf{p}$ *means that the substring* $\mathbf{t}_k^l$ *is a prefix of the pattern* $\mathbf{p}$*.*

1.2. Analysis. In the "average case analysis" we indicate that under assumption of *Stationary Model* (both strings $\mathbf{p}$ and $\mathbf{t}$ are random realizations of a *stationary* and *ergodic* sequence), the average complexity C_n may be computed by a direct application of an extension of Kingman's *Subadditive Ergodic Theorem* due to Derriennic [4] . See also [5].

LEMMA 1. [7] *A strongly semi-sequential algorithm satisfies the following basic inequality for all* r *such that* $1 \le r \le n$*:*

$$|c_{1,n} - (c_{1,r} + c_{r,n})| \le 2m^2,$$

provided any comparison is done only once.

We get also:

THEOREM 2. *With* $\mathbf{p}$ *a pattern of size* m*,* $\mathbf{t}$ *a text of size* n*, and a strongly-sequential algorithm, the number of comparisons is given by:*

(a) *worst case:* $\lim_{n\to\infty} \max_t c_n(t,p)/n = \alpha_1(p)$*,*
(b) $\mathbf{p}$ *given,* $\mathbf{t}$ *random:* $C_n(p)/n \overset{p.s.}{\to} \alpha_2(p)$ *(on the average),*
(c) $\mathbf{p}, \mathbf{t}$ *random:* $\lim_{n\to\infty} E_{t,p} C_n/n = \alpha_3 \ge 1$*.*

In the Boyer-Moore algorithm [1], a window of size equal to the size of the pattern is moved from left to right, with shifts depending of the text and pattern contents; inside the window, scanning is performed from right to left; the Boyer-Moore algorithm gives a counterexample to the preceding theorem, inside the class of pattern-matching algorithms: given the text $\mathbf{t} = \{\cdots y^{10} a z^4 (bazbzz)^n \cdots\}$, and a pattern $\mathbf{p} = \{x^4 a x^2 b x^2 a\}$, it is impossible to find a set of unavoidable positions for the Boyer-Moore algorithm.

2. Boyer-Moore algorithm

For the Boyer-Moore algorithm, a *head* is the rightmost position of the text in the window after a shift; let H_n be the number of heads in a text of length n. We show by a Laplace transform method the convergence of H_n to a variable with normal distribution.

Both expectation and variance of H_n are functions of the *shift polynomial*, defined as $f_p(z) = \sum_a q_a z^{d(a)}$, where $d(a)$ is the shift of the first occurrence of letter a from the right extremity of the pattern and q_a is the probability of occurrence of letter a. With this definition, the shift polynomial of the pattern 10001 is $\frac{1}{2}(z + z^4)$, with uniform distribution for letters 0 and 1.

When considering the complexity $C_n^{[P]}$ of the algorithm for a fixed pattern P and a text of length n, we define X_i as the number of comparisons done for an alignment at position i, and $Z_j = 1$ when j is a head, 0 otherwise. We have

$$C_n^{[P]} = \sum_{i=m}^{n} X_i Z_i.$$

After an algebraic manipulation, we take the expectation:

$$E\left[\frac{C_n^{[P]}}{n}\right] = \frac{1}{n}\sum_{i=m}^{n} E[X_i Z_i] - \frac{1}{n}\sum_{i=m}^{n} E[X_j(1 - Z_j)].$$

From this decomposition, we show that $E\left[\frac{1}{n}C_n^{[P]}\right] \to c_P$, and give an expression for c_P. We show also that the fourth moment is bounded.

With these results for moments, we apply a central limit theorem for dependant variables [5], where the strong mixing condition is equivalent to independence of positions sufficiently distant. This proves the convergence of $C_n^{[P]}$ to a variable with normal distribution.

Unavoidable positions. Almost surely, for a random text, there exists one unavoidable position; formally, we say that Z_k is *determined* by $t_{j+1}\cdots t_{k-1}$ if this string is sufficient to tell whether $Z_k = 0$ or 1. We denote the indicator of this event by

$$\xi_k^{(j)} = 1_{\{Z_k \text{ determined by } t_{j+1}\cdots t_{k-1}\}};$$

we have then:

LEMMA 2. *$E[1-\xi_k^{(j)}] \le \rho^{k-j-2}$, where $\rho < 1$, for $k - j \ge 2m$.*

PROOF. [Sketch] If $\xi_k^{(j)} = 0$, then p_{m-1} does not occur $m-1$ times consecutively in $t_{j+1}\cdots t_{k-1}$. Given a fixed set of $m-1$ consecutive characters, the probability that not all of them are equal to p_{m-1} is A, with $A < 1$. The probability of no string of $m-1$ consecutive occurrences of p_{m-1} is at most $A^{\lfloor (k-j-2)(m-1)\rfloor}$; take $\rho = A^{1/(2m)}$. □

3. Number of occurrences of a word

We extended the classical result of Guibas and Odlyzko [6] to the Markovian case, giving all moments. This is done by constructing language expressions that characterize both models, and by analysis on the corresponding generating functions.

Bibliography

[1] Boyer (R.) and Moore (J.). – A fast string searching algorithm. *Communications of the ACM*, vol. 20, 1977, pp. 762–772.

[2] Crochemore (M.) and Rytter (W.). – *Text Algorithms*. – Oxford University Press, 1994.

[3] D. E. Knuth (J. Morris) and Pratt (V.). – Fast pattern matching in strings. *SIAM Journal on Computing*, vol. 6, 1977, pp. 189–195.

[4] Derriennic (Y.). – Un théorème ergodique presque sous additif. *The Annals of Probability*, vol. 11, 1983, pp. 669–677.

[5] Durrett (R.). – *Probability: Theory and Examples*. – Wadsworth & Brooks/Cole Books, Pacific Grove, California, 1991.

[6] Guibas (L. J.) and Odlyzko (A. M.). – Strings overlaps, pattern matching and non-transitive games. *Journal of Combinatorial Theory, Series A*, vol. 30, 1981, pp. 183–208.

[7] Regnier (M.) and Szpankowski (W.). – *Complexity of Sequential Pattern Matching Algorithms*. – Research Report n° 2549, Institut National de Recherche en Informatique et en Automatique, 1995.