A CORRECT PREPROCESSING ALGORITHM FOR BOYER-MOORE STRING-SEARCHING*

WOJCIECH RYTTER†

Abstract. We present the correction to Knuth's algorithm [2] for computing the table of pattern shifts later used in the Boyer–Moore algorithm for pattern matching.

Key words. algorithm, pattern-matching, string, overlap

The key to the Boyer–Moore algorithm for the fast pattern matching is the application of the table of pattern shifts which is denoted in [1] by $\Delta_2$ and in [2] by $dd'$. Let us denote this table by $D$.

Assume that the pattern is given by the array pattern $[1:n]$, so $D$ is given as an array $D[1:n]$. For every $1 \leq j \leq n$, $D[j]$ gives the minimum shift $d > 0$ such that the pattern with the right end placed at the position $k + d$ of the processing string is compatible with the part of string scanned before, where $k$ is the last scanned position in the string and $j$ is the last scanned position in the pattern.

The formal definition of $D$ given in [2] is:

$$D[j] = \text{MIN}\{s + n - j | s \geq 1 \text{ and } (s \geq j \text{ or pattern } [j-s] \neq \text{pattern } [j]) \text{ and } ((s \geq i \text{ or pattern } [i-s] = \text{pattern } [i]) \text{ for } j < i \leq n)\}.$$ 

Algorithm A given by Knuth is:

A1. for $k := 1$ step 1 until $n$ do $D[k] := 2^n n - k$;

A2. $j := n; t := n + 1$;

while $j > 0$ do

begin

$f[j] := t$;

while $t \leq n$ and pattern $[j] \neq \text{pattern } [t]$ do

begin

$$D[t] := \text{MIN}\left(D[t], n - j\right);$$

$$t := f[t];$$

end

$t := t - 1; j := j - 1$;

end;

A3. for $k := 1$ step 1 until $t$ do

$$D[k] := \text{MIN}\left(D[k], n + t - k\right);$$

Algorithm A computes also the auxiliary table $f[0:n]$, for $j < n$ defined as follows: $f[j] = \min\{i | j < i \leq n \text{ and pattern } [i+1] \cdots \text{pattern } [n] = \text{pattern } [j+1] \cdots \text{pattern } [n + j - i]\}$; the final value of $t$ corresponds to $f[0]$. $f[0]$ is the minimum non-zero shift of pattern on itself; let us denote this value by $\text{SHIFT}(\text{pattern})$.

* Received by the editor January 18, 1979, and in revised form May 25, 1979.

† Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México Apartado Postal 20-726, México 20, D.F. On leave of absence from Institute of Informatics, Department of Mathematics, Warsaw University, Warsaw, Poland.

509
Take as inputs to Algorithm A the following two strings: pattern 1 = aaaaaaaaaaa and pattern 2 = abaabaabaa. Denoting by DefD and $D'$ respectively the value of $D$ according to the definition and computed by Algorithm A we obtain the following results:

\[
\begin{align*}
  j & \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \\
  \text{pattern 1}[j] & = \quad a \quad a \quad a \quad a \quad a \quad a \quad a \quad a \\
  \text{DefD}[j] & = 10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10 \\
  D'[j] & = 10 \quad 18 \quad 17 \quad 16 \quad 15 \quad 14 \quad 13 \quad 12 \quad 11 \quad 10 \\
  \text{SHIFT(pattern 1)} &= 1. \\
  \text{pattern 2}[j] & = \quad a \quad b \quad a \quad a \quad b \quad a \quad a \quad a \\
  \text{DefD}[j] & = 12 \quad 11 \quad 10 \quad 12 \quad 11 \quad 10 \quad 12 \quad 11 \quad 2 \quad 2 \\
  D'[j] & = 12 \quad 11 \quad 10 \quad 16 \quad 15 \quad 14 \quad 13 \quad 12 \quad 2 \quad 2 \\
  \text{SHIFT(pattern 2)} &= 3.
\end{align*}
\]

The disagreement between DefD and $D'$ demonstrates explicitly that Knuth's algorithm is incorrect.

There are three cases which are considered in the design of Algorithm A for computing the value of $D[j]$:

- **Case (1).** $D[j] = 2^*n - j$. This is the most simple case computed in the part A1 of Algorithm A.
- **Case (2).** $D[j] < n$ and pattern $[l] \neq \text{pattern}[j]$, where $l = n - D[j]$. In this case $D[j]$ is computed in the part A2.
- **Case (3).** $n \leq D[j] < 2^*n - j$ and $j \leq \text{SHIFT(pattern)} = f[0] = t$. In this case $D[j]$ is computed in the part A3 of Algorithm A.

However, another case occurs which is not covered by Cases (1), (2) and (3):

- **Case (4).** $n < D[j] < 2^*n - j$ and $j > \text{SHIFT(pattern)}$. For example it occurs for pattern = pattern 2 and $j = 5$. To correct Algorithm A, we have to consider not only the minimal nonzero shift of the string on itself but all shifts, namely all $i$ such that $0 < i \leq n$ and pattern $[i + 1] \cdots \text{pattern} [n] = \text{pattern}[1] \cdots \text{pattern}[n - i]$. Let us denote the set of all such $i$ by ALLSHIFTS(pattern). Using the method of computing the failure function in the pattern-matching algorithm of Knuth, Morris and Pratt [2], we give below a correct version of the algorithm, where A1, A2 denote the corresponding parts of Algorithm A.

**Algorithm B.**

A1; A2;

\[
\begin{align*}
  q & := t; \quad t := n + 1 - q; \quad q1 := 1; \\
  B1. & \quad j1 := 1; \quad t1 := 0; \\
  & \quad \text{while } j1 \leq t1 \text{ do} \\
  & \quad \quad \text{begin} \\
  & \quad \quad \quad f1[j1] := t1; \\
  & \quad \quad \quad \text{while } t1 \leq 1 \text{ and } \text{pattern}[j1] \neq \text{pattern}[t1] \text{ do} \\
  & \quad \quad \quad \quad t1 := f1[t1]; \\
  & \quad \quad \quad t1 := t1 + 1; \quad j1 := j1 + 1; \\
  & \quad \quad \text{end};
\end{align*}
\]
BOYER-MOORE STRING-SEARCHING

B2. while $q < n$ do
   begin
      for $k := q + 1$ step 1 until $q$ do $D[k] := \text{Min} (D[k], n + q - k);$ 
      $q \leftarrow q + 1; q := q + t - f1[t];$
      $t \leftarrow f1[t];$ end;

The part B1 computes the auxiliary table $f1[1:t']$ where $t' = n + 1 - \text{SHIFT(pattern)}$, and the part B2 computes the values of $D[j]$ for both Cases (3) and (4).

The correctness of the part B2 follows from the following: If $\text{ALLSHIFTS(pattern)} = \{i_1, i_2, \ldots, i_k\}$ and $i_1 = \text{SHIFT(pattern)}$ and $i_1 < i_2 < \cdots < i_k$ and $t_1 = n + 1 - i_1$, $t_{p+1} = f1[t_p]$ for $p = 1, 2, \cdots, (k - 1)$ then $t_{p+1} = t_p + t_{p+1} - t_{p+1}$ for $p = 1, 2, \cdots, (k - 1)$.

Remark 1. The same table space can be used for $f$ and $f1$.

Remark 2. The tables $f$ and $f1$ are related in the following way: Let pattern' be the string resulting from reversing the string pattern and $f1$ be computed for the string pattern and $f$ be computed for pattern'.

Then

\[ f1[i] = n - f[n - i + 1] + 1 \quad \text{for } i = 1, 2, \cdots, (n + 1). \]

Remark 3. Denote $\text{OVR(pattern)} = n - \text{SHIFT(pattern)}$. So OVR(pattern) gives the maximum overlap of the pattern with itself. The difference in the time complexity of Algorithms A and B is proportional to OVR(pattern) which can be linear with respect to $n$. However, on the average it is very small for alphabets of the size greater than 1. Let $V(n, k)$ denotes the average value of OVR(pattern) taken over the set of all patterns of the length $n$ over the same alphabet of the size $k$.

The rounded values of $V(n, 2)$ for $n \leq 14$ computed on B6700 are shown in Table 1.

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
</tr>
<tr>
<td>$V(n, 2)$</td>
</tr>
<tr>
<td>$n$</td>
</tr>
<tr>
<td>$V(n, 2)$</td>
</tr>
</tbody>
</table>
LEMMA. 1. If $k > 1$ then $V(n, k) < k/(k - 1)^2$.
2. $V(n, 2) < 2$.
3. $V(n, k) < 1$ for $k > 2$.

Proof. Fix $n$ and $k$ and assume that $k > 1$. Let $a_j$ be the number of patterns such that $OVR(pattern) = j$ for $j = 1, 2, \ldots, (n - 1)$. Every pattern with $OVR(pattern) = j$ is determined by its prefix of the length $n - j$. So $a_j \leq k^{n-j}$. Hence $V(n, k) = (\sum_{j=1}^{n-1} a_j)/k^n \leq \sum_{j=1}^{n-1} j \cdot (1/k)^j \leq \sum_{j=1}^{\infty} j \cdot (1/k)^j = k/(k - 1)^2$. Parts 2 and 3 of the lemma follow from 1. This ends the proof.

REFERENCES