String Matching: Boyer-Moore Algorithm

Greg Plaxton
Theory in Programming Practice, Fall 2005
Department of Computer Science
University of Texas at Austin
Notation

• We abbreviate \( \min\{p - r \mid r \in R\} \) as \( \min(p - R) \)

• In general, if \( S \) is a set of strings and \( e(S) \) an expression that includes \( S \) as a term, then \( \min(e(S)) = \min\{e(i) \mid i \in S\} \), where \( e(i) \) is obtained from \( e \) by replacing \( S \) by \( i \)

• We adopt the convention that the minimum of the empty set is \( \infty \)
Basic Definitions

• Let \( R \) denote \( R' \cup R'' \), where \( R' \) is

\[
\{ r \text{ is a proper prefix of } p \land r \text{ is a suffix of } s \}
\]

and \( R'' \) is

\[
\{ r \text{ is a proper prefix of } p \land s \text{ is a suffix of } r \}
\]

• Recall that

\[
b(s) = \min \{ \overline{p} - \overline{r} \mid r \in R \}
\]

• Thus

\[
b(s) = \min(\min(\overline{p} - R'), \min(\overline{p} - R''))
\]
Properties of $b(s)$

- **P1:** $c(p) \in R$
- **P2:** $\min(\overline{p} - R') \geq \overline{p} - \overline{c(p)}$
- **P3:** If
  \[ V = \{ v \mid v \text{ is a suffix of } p \land c(v) = s \} \]
  then $\min(\overline{p} - R'') = \min(V - \overline{s})$
Proof of Property P1

• P1: \( c(p) \in R \)

• From the definition of core, \( c(p) \prec p \)

• Hence, \( c(p) \) is a proper prefix of \( p \)

• Also, \( c(p) \) is a suffix of \( p \), and, since \( s \) is a suffix of \( p \), they are totally ordered, i.e., either \( c(p) \) is a suffix of \( s \) or \( s \) is a suffix of \( c(p) \)

• Hence, \( c(p) \in R \)
Proof of Property P2

• P2: $\min(\bar{p} - R') \geq \bar{p} - c(p)$

• Consider any $r$ in $R'$

• Since $r$ is a suffix of $s$ and $s$ is a suffix of $p$, $r$ is a suffix of $p$

• Also, $r$ is a proper prefix of $p$, so $r \prec p$

• From the definition of core, $r \preceq c(p)$, and hence $\bar{p} - \bar{r} \geq \bar{p} - c(p)$ for every $r$ in $R'$
Proof of Property P3

• P3: If
  \[ V = \{ v \mid v \text{ is a suffix of } p \land c(v) = s \} \]
  then \( \min(\overline{p} - R'') = \min(V - \overline{s}) \)

• We split the proof into two parts:
  – First, we show that \( \min(\overline{p} - R'') \leq \min(V - \overline{s}) \)
  – Then, we show that \( \min(\overline{p} - R'') \geq \min(V - \overline{s}) \)
Proof that $\min(\overline{p} - R'') \leq \min(V - \overline{s})$

- If $V$ is empty, the inequality holds since the RHS is $\infty$; in what follows, assume that $V$ is nonempty and let $v$ be an arbitrary element of $V$.

- It is sufficient to exhibit an $r$ in $R''$ such that $\overline{p} - \overline{r} = \overline{v} - \overline{s}$.

- Let $r$ be the length-$(\overline{p} - \overline{v} + \overline{s})$ prefix of $p$.
  - Note that $r$ is a proper prefix of $p$ since $c(v) = s$ implies $\overline{v} > \overline{s}$.
  - Furthermore, $s$ is a suffix of $r$ since $c(v) = s$ implies that $s$ is a prefix of $v$.
  - So $r$ belongs to $R''$, as required.
Proof that \( \min(\overline{p} - R'') \geq \min(V - \overline{s}) \)

- If \( R'' \) is empty, the inequality holds since the LHS is \( \infty \); in what follows, assume that \( R'' \) is nonempty and let \( r \) be the string in \( R'' \) minimizing the LHS.

- It is sufficient to exhibit a \( v \) in \( V \) such that \( \overline{p} - \overline{r} = \overline{v} - \overline{s} \).

- Let \( v \) denote the length-\((\overline{p} - \overline{r} + \overline{s})\) suffix of \( p \):
  - Note that \( \overline{v} > \overline{s} \) since \( r \) is a proper prefix of \( p \).
  - Furthermore, \( s \prec v \), so \( s \preceq c(v) \).
  - If \( s \prec c(v) \), then we obtain a contradiction to the definition of \( r \) since the length-\( (\overline{r} + c(v) - \overline{s}) \) prefix \( r' \) of \( p \) also belongs to \( R'' \) and yields a smaller value for the LHS.
  - Thus \( s = c(v) \) and hence \( v \) belongs to \( V \), as required.
A Formula for $b(s)$

- We now derive a formula for $b(s)$, where

$$V = \{v \mid v \text{ is a suffix of } p \land c(v) = s\}$$

\[
\begin{align*}
b(s) & = \{\text{definition of } b(s)\} \\
& = \min(\bar{p} - R) \\
& = \{\text{from (P1): } c(p) \in R\} \\
& = \min(\bar{p} - c(p), \min(\bar{p} - R)) \\
& = \{R = R' \cup R''\} \\
& = \min(\bar{p} - c(p), \min(\bar{p} - R'), \min(\bar{p} - R'')) \\
& = \{\text{from (P2): } \min(\bar{p} - R') \geq \bar{p} - c(p)\} \\
& = \min(\bar{p} - c(p), \min(\bar{p} - R'')) \\
& = \{\text{from (P3): } \min(\bar{p} - R'') = \min(V - \bar{s})\} \\
& = \min(\bar{p} - c(p), \min(V - \bar{s}))
\end{align*}
\]
Computation of $b$: Towards An Abstract Program

- We now develop an abstract program to compute $b(s)$, for all suffixes $s$ of $p$
- We employ an array $b$ where $b[s]$ ultimately holds the value of $b(s)$, though it is assigned different values during the computation
- Initially, we set $b[s]$ to $\overline{p} - c(p)$
- Next, for each suffix $v$ of $p$ (in arbitrary order)
  - Let $s = c(v)$
  - Update $b[s]$ to $\min(b[s], \overline{v} - \overline{s})$
Computation of $b$: An Abstract Program

- Here is our abstract program for computing $b(s)$ for all suffixes $s$ of $p$
  assign $\overline{p} - c(p)$ to all elements of $b$;
  for all suffixes $v$ of $p$ do
    $s := c(v)$;
    if $b[s] > \overline{v} - \overline{s}$ then $b[s] := \overline{v} - \overline{s}$ endif
  endfor
Computation of $b$: Towards a Concrete Program

- The goal of the concrete program is to compute an array $e$, where $e[j]$ is the amount by which the pattern is to be shifted when the matched suffix is $p[j..p]$, $0 \leq j \leq p$
  - $e[j] = b[s]$, where $j + s = p$, or
  - $e[p - s] = b[s]$, for any suffix $s$ of $p$

- We have no need to keep explicit prefixes and suffixes; instead, we keep their lengths, $s$ in $i$ and $v$ in $j$

- Let array $f$ hold the lengths of the cores of all suffixes of $p$ suffixes $v$ of $p$, i.e., $f[\bar{v}] = c(v)$
Computation of $b$: A Concrete Program

• Here is our concrete program for computing $b(s)$ for all suffixes $s$ of $p$
  
  assign $\overline{p} - c(p)$ to all elements of $e$;
  for $j$, $0 \leq j \leq \overline{p}$, do
  $i := f[j]$;
  if $e[\overline{p} - i] > j - i$ then $e[\overline{p} - i] := j - i$ endif
  endfor

• It remains to compute $f$
Computation of $f$

- Here we are asked to compute the (length of the) core of every suffix of $p$

- Recall that the preprocessing phase of the KMP algorithm computes the core of every prefix of $p$ in $O(p)$ time

- A symmetric approach can be used to compute the core of every suffix of $p$ in $O(p)$ time
Computation of $b$: Time Complexity

- The computation of $b(s)$, for all suffixes $s$ of $p$, requires computing array $f$ and executing the concrete program presented earlier
  - Note that $c(p) = f[\overline{p}]$

- As we have indicated on the previous slide, the array $f$ can be computed in $O(p)$ time

- Given $f$, the concrete program runs in $O(\overline{p})$ time since the loop iterates $O(\overline{p})$ times, and each execution of the loop body takes constant time