Exact pattern matching
Boyer-Moore & SHIFT-and-OR

x=abbacbbbababacabbbba

6

17
Variation on the simple algorithm

\[
i = m \\
\textbf{while } i \leq |x|: \\
i, j = \text{hctam}(i, m) \\
\textbf{if } j==0: \text{ report } i+1 \text{ as match} \\
i = i + m - j + 1
\]

\[
x=\text{abbacbbbababacabbbba} \\
p=\text{bbba}
\]

\[
\text{hctam}(i,j): \\
\textbf{while } x[i] == p[j] \\
\textbf{and } j \geq 0: \\
i = i - 1 \\
j = j - 1 \\
\text{return } i, j
\]
Variation on the simple algorithm

\[ i = m \]
\[ \textbf{while} \ i \leq |x|:\]
\[ \quad i,j = \text{hctam}(i,m) \]
\[ \quad \textbf{if} \ j==0: \ \text{report} \ i+1 \ \text{as match} \]
\[ i = i + m - j + 1 \]

\[ \text{hctam}(i,j): \]
\[ \quad \textbf{while} \ x[i]==p[j] \quad \text{and} \ j \geq 0:\]
\[ \quad \quad i = i - 1 \]
\[ \quad \quad j = j - 1 \]
\[ \quad \text{return} \ i,j \]

\[ x=\text{abbacbbbababacabbbba} \]
\[ p=\text{bbba} \]
Variation on the simple algorithm

i = m

while i <= |x|:
    i, j = hctam(i, m)
    if j == 0: report i+1 as match
    i = i + m - j + 1

hctam(i, j):
    while x[i] == p[j] and j >= 0:
        i = i - 1
        j = j - 1
    return i, j

\[x = abba cbbbababacabbbba\]
\[p = bbba\]
Variation on the simple algorithm

\[
\begin{align*}
\text{i} &= \text{m} \\
\text{while } \text{i} \leq |\text{x}|: \\
\quad &\text{i, j = hctam(i, m)} \\
\quad &\text{if } \text{j} == 0: \text{ report } \text{i+1} \text{ as match} \\
\quad &\text{i} = \text{i} + \text{m} - \text{j} + 1 \\
\end{align*}
\]

\[
\text{hctam}(\text{i, j}): \\
\text{while } \text{x}[\text{i}] == \text{p}[\text{j}] \\
\quad \text{and } \text{j} \geq 0: \\
\quad \text{i} = \text{i} - 1 \\
\quad \text{j} = \text{j} - 1 \\
\text{return } \text{i, j}
\]

\[
\text{x=abbacbbbababacabbbba} \\
\text{p=bbba}
\]
Variation on the simple algorithm

\[ i = m \]

\[ \text{while } i \leq |x| : \]
\[ i,j = \text{hctam}(i,m) \]
\[ \text{if } j == 0: \text{ report i+1 as match} \]
\[ i = i + m - j + 1 \]

```
hctam(i,j):
    while x[i] == p[j] and j >= 0:
        i = i - 1
        j = j - 1
    return i, j
```

\[ x=abba\textbf{c}bbbabababacabbbba \]
\[ p=bbba \]
\[ i=5 \]
\[ j=4 \]
Variation on the simple algorithm

```python
i = m
while i <= |x|:
    i, j = hctam(i, m)
    if j == 0: report i+1 as match
    i = i + m - j + 1

hctam(i, j):
    while x[i] == p[j] and j >= 0:
        i = i - 1
        j = j - 1
    return i, j
```

\[ x = abbacbbbababacabbbba \]

\[ p = bbba \]
Variation on the simple algorithm

\[ i = m \]

\[ \textbf{while } i \leq |x| : \]

\[ i, j = \text{hctam}(i, m) \]

\[ \textbf{if } j==0: \text{ report } i+1 \text{ as match } \]

\[ i = i + m - j + 1 \]

\[ \text{hctam}(i, j): \]

\[ \textbf{while } x[i]==p[j] \]

\[ \text{and } j >= 0 : \]

\[ i = i - 1 \]

\[ j = j - 1 \]

\[ \text{return } i, j \]

\[ i=6 \]

\[ x=abbacbbbababacabbbba \]

\[ p=bbbaa \]

\[ j=4 \]
Variation on the simple algorithm

\[
i = m \\
\textbf{while} \ i \leq |x|: \\
i, j = \text{hctam}(i, m) \\
\textbf{if} \ j == 0: \text{report} \ i + 1 \text{ as match} \\
i = i + m - j + 1
\]

\[
\text{hctam}(i, j): \\
\textbf{while} \ x[i] == p[j] \\
\quad \text{and} \ j \geq 0: \\
\quad i = i - 1 \\
\quad j = j - 1 \\
\textbf{return} \ i, j
\]\n
\[
i=7 \
\]
\[
x = \text{abbacbbbababacabbbba} \\
p = \text{bbba}
\]
Variation on the simple algorithm

\[ hctam(i,j): \]
\[ \text{while } x[i] == p[j] \text{ and } j \geq 0: \]
\[ i = i - 1 \]
\[ j = j - 1 \]

\[ \text{return } i, j \]

\[ i = m \]
\[ \text{while } i \leq |x|: \]
\[ i, j = hctam(i,m) \]
\[ \text{if } j == 0: \text{ report } i+1 \text{ as match} \]
\[ i = i + m - j + 1 \]

\[ x = abbacbbbababacabbbba \]
\[ p = bbbaa \]

\[ i = 7 \]
\[ j = 4 \]
Variation on the simple algorithm

\[
i = m\\
\text{while } i \leq |x|: \\
\quad i,j = \text{hctam}(i,m) \\
\quad \text{if } j==0: \text{ report } i+1 \text{ as match} \\
\quad i = i + m - j + 1\\
\text{hctam}(i,j): \\
\quad \text{while } x[i]==p[j] \text{ and } j >= 0: \\
\quad \quad i = i - 1 \\
\quad \quad j = j - 1 \\
\quad \text{return } i,j
\]

\[x=abbacbbbababacabbbba \\
p=bbba\]
Variation on the simple algorithm

\[
i = m
\]

\[
\textbf{while }\ i \leq |x|:\n\]
\[
i, j = hctam(i, m)\]
\[
\textbf{if }\ j == 0: \text{ report } i+1 \text{ as match}\]
\[
i = i + m - j + 1\]

\[
hctam(i, j):\]
\[
\textbf{while }\ x[i] == p[j] \text{ and } j \geq 0:\n\]
\[
i = i - 1\]
\[
j = j - 1\]
\[
\textbf{return } i, j\]

\[
i=8\]
\[
\textbf{x}=abba\textbf{c}\textbf{b}b\textbf{b}babab\textbf{a}\textbf{c}\textbf{a}b\textbf{b}b\textbf{b}ba\]
\[
p=bbba\]
\[
j=4\]
Variation on the simple algorithm

\[ \text{i} = \text{m} \]

\[ \text{while } \text{i} \leq |\text{x}|: \]
\[ \text{i}, \text{j} = \text{hctam}(\text{i}, \text{m}) \]
\[ \text{if } \text{j} == 0: \text{ report } \text{i}+1 \text{ as match} \]
\[ \text{i} = \text{i} + \text{m} - \text{j} + 1 \]

hctam(i,j):
\[ \text{while } \text{x}[\text{i}] == \text{p}[\text{j}] \]
\[ \text{ and } \text{j} \geq 0: \]
\[ \text{i} = \text{i} - 1 \]
\[ \text{j} = \text{j} - 1 \]
\[ \text{return } \text{i, j} \]

\[ i=9 \]
\[ x=\text{abbacbbbababacabbbba} \]
\[ p=\text{bbba} \]
Variation on the simple algorithm

\[
\begin{align*}
i &= m \\
\textbf{while} \ i &\leq |x|: \\
&\quad i, j = \text{hctam}(i, m) \\
&\quad \textbf{if} \ j == 0: \text{ report } i + 1 \text{ as match} \\
&\quad i = i + m - j + 1 \\
\end{align*}
\]

\[
\text{hctam}(i, j):
\]
\[
\begin{align*}
&\quad \textbf{while} \ x[i] == p[j] \\
&\quad \quad \text{and} \ j \geq 0: \\
&\quad \quad \quad i = i - 1 \\
&\quad \quad \quad j = j - 1 \\
&\quad \quad \text{return } i, j
\end{align*}
\]

\begin{align*}
i &= 5 \\
x &= \text{abbac} \text{bbba} \text{ababacabbbba} \\
p &= \text{bbba} \\
j &= 0
\end{align*}
Variation on the simple algorithm

\[
\begin{align*}
i &= m \\
\text{while } i \leq |x|: \\
&\hspace{1em} i, j = \text{hctam}(i, m) \\
&\hspace{1em} \text{if } j == 0: \text{ report } i + 1 \text{ as match} \\
&\hspace{1em} i = i + m - j + 1
\end{align*}
\]

\[
\text{hctam}(i,j):
\begin{align*}
\text{while } &\ x[i] == p[j] \text{ and } j \geq 0: \\
&\hspace{1em} i = i - 1 \\
&\hspace{1em} j = j - 1 \\
&\hspace{1em} \text{return } i, j
\end{align*}
\]

\[x=abbacbbbababacabbbbaa\]

\[p=bbba\]

Match at \(i=6\)
Variation on the simple algorithm

\[ hctam(i, j): \]

\[ \text{while } x[i] == p[j] \text{ and } j \geq 0: \]
\[ i = i - 1 \]
\[ j = j - 1 \]
\[ \text{return } i, j \]

\[ i = m \]
\[ \text{while } i \leq |x|: \]
\[ i, j = hctam(i, m) \]
\[ \text{if } j == 0: \text{ report } i+1 \text{ as match} \]
\[ i = i + m - j + 1 \]

\[ x = abbacbbbabaababacabcabbbba \]
\[ p = bbba \]
Running time

- Works as good (or bad) as the simple algorithm
  - Time and space in $O(|x|m) = O(n^2)$
Trick: rightmost occurrence

If we have a mismatch at $p[j]:x[i]$, and $j' < j$ is the rightmost occurrence of $x[i]$ in $p$:

then align $j'$ with $i$:
“Rightmost” array

Define array, \( R \), such that for each letter \( a \), \( R[a] \) is the distance from the right of \( p \) to the rightmost occurrence of \( a \) in \( p \), or \( m \) if \( a \) is not in \( p \)

\[
\text{for } a \text{ in } \alpha: \quad R[a] = m \\quad \text{p=bbba} \\
\text{for } j = 1..m: \quad R[p[j]] = m-j \quad \alpha={a,b,c} \\
\]

\[
R[a] = 0 \quad \text{p=bbba} \\
R[b] = 1 \quad \text{p=bbba} \\
R[c] = 4 \quad \text{p=bbba} 
\]
“Rightmost” array

Define array, $R$, such that for each letter $a$, $R[a]$ is the distance from the right of $p$ to the rightmost occurrence of $a$ in $p$, or $m$ if $a$ is not in $p$

```
for a in α:
    R[a] = m
for j = 1..m:
    R[p[j]] = m-j
```

$p=bbba$
$α={a,b,c}$

$p=bbba$
$R[a] = 0$  
$R[b] = 1$  
$R[c] = 4$

NB: $R$ is called $δ_1$ in the book
Updated algorithm

\[ i = m \]

**while** \( i \leq |x| : \)

\[ i, j = hctam(i, m) \]

**if** \( j == 0: \) report \( i+1 \) as match

\[ i = i + \max\{m-j+1, R[x[i]]\} \]

\[ x=abbacbbbababacabbbba \]

\[ p=bbba \]

\[ hctam(i, j): \]

**while** \( x[i] == p[j] \)

**and** \( j \geq 0: \)

\[ i = i - 1 \]

\[ j = j - 1 \]

**return** \( i, j \)
Updated algorithm

\[
\text{hctam}(i,j): \\
\text{while } x[i] == p[j] \text{ and } j >= 0: \\
\quad i = i - 1 \\
\quad j = j - 1 \\
\text{return } i, j
\]

\[
i = m \\
\text{while } i <= |x|: \\
\quad i, j = \text{hctam}(i,m) \\
\quad \text{if } j == 0: \text{ report } i+1 \text{ as match} \\
\quad i = i + \max\{m-j+1, R[x[i]]\}
\]

\[
x=\text{abbacbbbababacabbbba} \\
p=\text{bbba}
\]

\[i=4\]
Updated algorithm

\[
\begin{align*}
i &= m \\
\textbf{while} & \ i \leq |x|: \\
& \quad i, j = \text{hctam}(i, m) \\
& \quad \textbf{if} \ j == 0: \ \text{report} \ i+1 \ \text{as match} \\
& \quad i = i + \max\{m-j+1, R[x[i]]\}
\end{align*}
\]

\[
\begin{align*}
hctam(i,j): \\
& \quad \textbf{while} \ x[i] == p[j] \\
& \quad \quad \text{and} \ j \geq 0: \\
& \quad \quad i = i - 1 \\
& \quad \quad j = j - 1 \\
& \quad \textbf{return} \ i, j
\end{align*}
\]

\[
\begin{align*}
i &= 1 \\
x &= \text{abba}cbbbaba\text{babacabbbba} \\
p &= \text{bbba}
\end{align*}
\]
**Updated algorithm**

\[ i = m \]

\[ \text{while } i \leq |x| : \]
\[ \quad i, j = \text{hctam}(i, m) \]
\[ \quad \text{if } j == 0: \text{ report } i+1 \text{ as match} \]
\[ \quad i = i + \max\{m-j+1, R[x[i]]\} \]

\[ \text{hctam}(i, j): \]
\[ \quad \text{while } x[i] == p[j] \]
\[ \quad \quad \text{and } j \geq 0: \]
\[ \quad \quad \quad i = i - 1 \]
\[ \quad \quad \quad j = j - 1 \]
\[ \quad \text{return } i, j \]

\[ x = abba\textcolor{green}{cbbbabababacabbbbba} \]

\[ p = \textcolor{green}{bbba} \]

\[ i=1 \]
\[ j=1 \]

\[ m-j+1 = 4 \]
\[ \text{and } R[x[1]] = R[a] = 0 \]
Updated algorithm

```python
i = m
while i <= |x|:
    i, j = hctam(i, m)
    if j == 0: report i+1 as match
    i = i + max{m-j+1, \text{R}[x[i]]}

hctam(i, j):
    while x[i] == p[j] and j >= 0:
        i = i - 1
        j = j - 1
    return i, j
```

\[ x = \text{abbacbbbababacabbbba} \]
\[ p = \text{bbba} \]

\[ m-j+1 = 4 \]
\[ \text{and } \text{R}[x[1]]=\text{R}[a]=0 \]
Updated algorithm

\[
\text{i = m}
\]

\[
\text{while } i \leq |x|:
\]

\[
\text{i, j = hctam(i, m)}
\]

\[
\text{if } j == 0: \text{ report } i+1 \text{ as match}
\]

\[
\text{i = i + max(m-j+1, R[x[i]])}
\]

\[
hctam(i, j):
\]

\[
\text{while } x[i] == p[j] \text{ and } j \geq 0:
\]

\[
i = i - 1
\]

\[
j = j - 1
\]

\[
\text{return } i, j
\]

\[
\begin{align*}
\text{x} &= \text{abba} \text{cbbbababacabbbba} \\
\text{p} &= \text{bbbaa}
\end{align*}
\]
i = m
while i <= |x|:
    i, j = hctam(i, m)
    if j == 0: report i+1 as match
    i = i + max{m−j+1, R[x[i]]}

hctam(i, j):
    while x[i] == p[j] and j >= 0:
        i = i − 1
        j = j − 1
    return i, j

\begin{align*}
\text{x} &= \text{abba}c\text{bbbababacabbbba} \\
\text{p} &= \text{bbba} \\
\text{m−j+1} &= 1 \\
R[x[5]] &= R[c] = 4
\end{align*}
Updated algorithm

\[
i = m
\]
\[
\textbf{while} \ i \leq |x|:\n\]
\[
i, j = \text{hctam}(i, m)\n\]
\[
\textbf{if} \ j == 0: \text{report} \ i + 1 \ \text{as match} \n\]
\[
i = i + \max\{m - j + 1, R[x[i]]\}\n\]

\[
x = \text{abbacbbbababacabbbbba} \n\]
\[
p = \text{bbba} \n\]

\[
i = 9 \n\]

\[
m - j + 1 = 1 \n\]
\[
R[x[5]] = R[c] = 4\n\]
**Updated algorithm**

\[
i = m
\]

\[
\textbf{while } i \leq |x|:\n\]

\[
i, j = \text{hctam}(i, m) \textbf{ if } j = 0: \text{ report } i + 1 \text{ as match} \]

\[
i = i + \max\{m - j + 1, R[x[i]]\}
\]

\[
hctam(i, j): \]

\[
\textbf{while } x[i] == p[j] \textbf{ and } j \geq 0:\n\]

\[
i = i - 1 \]

\[
j = j - 1 \]

\[
\textbf{return } i, j
\]

i=9

\[x=\text{abbacbbbababacabbbba}\]

\[p=\text{bbba}\]

A shortcut of 4 characters here!
Observation

If we have a mismatch at $p[j]:x[i]$:

then either there is rightmost $j'<j$, where $p[j'] \neq p[j]$ and $p[j'..h]=p[j..m]$

or rightmost $j'<m$ where $p[1..j']$ is a suffix of $p[j..m]$:
**Trick 2: match suffixes...**

Define $\mathbf{S}_1[j] = j'$ from if it exists, and $\mathbf{S}_1[j] = 0$ otherwise

Define $\mathbf{S}_2[j] = j'$ if it exists, or $\mathbf{S}_2[j] = 0$ otherwise

Define

$$\mathbf{S}[j] = \min(m - \mathbf{S}_1[j], m - j + m - \mathbf{S}_2[j])$$

for $j = 1..m$, and

$\mathbf{S}[0] = 2m$ - “longest border of $\mathbf{p}$” (special case of case 2)
Trick 2: match suffixes...

Define \( S_1[j] = j' \) from
if it exists, and \( S_1[j] = 0 \) otherwise

Define \( S_2[j] = j' \)
if it exists, or \( S_2[j] = 0 \) otherwise

Define \( S[j] = \min(m - S_1[j], m - j + m - S_2[j]) \) for \( j = 1..m \), and \( S[0] = 2m \) - “longest border of \( p \)” (special case of case 2)

**NB:** \( S \) (for “suffix”) is called \( \delta_2 \) in the book
Define $S_1[j] = j'$ from if it exists, and $S_1[j] = 0$ otherwise.

Define $S_2[j] = j'$ if it exists, or $S_2[j] = 0$ otherwise.

Define $S[j] = \min(m - S_1[j], m - j + m - S_2[j])$ for $j = 1..m$, and $S[0] = 2m$ - “longest border of $p$” (special case of case 2).

$S[j]$ is the amount we should increase $i$ to get a point where the string matches the suffix seen so far.
Computing $S_1$

$S_1[j]$ is the largest $j'$ such that

$$p[j'+1..j'+(m-j)]$$
is a border of $p[j'+1..m]$

and such that

$$p[j'] \neq p[j]$$
Computing $S_1$

Let $\rho$ be the suffix border array, i.e.:
$\rho[j']$ is the length of the longest border of $p[j'..m]$.

(\(\rho\) can be calculated similarly to the border array; Exercise 1.3.10).
Computing $S_1$

Let $\rho''[j']$ be the length of the longest border of $p[j'+1..m]$ such that $p[j'] \neq p[m-\rho''[j']-1]$.

($\rho''$ can be calculated similarly to $B''$; Exercise 7.1.9).
Computing $S_1$

A strategy that looks right, but doesn’t quite work…

for $j'=1..m$:
\[
  k = m - \rho''[j'] - 1 \\
  S[k] = j'
\]
Computing $S_1$

Problem: we set a pointer for each $j'$, but several borders can point to the same $j'$.
Computing $S_1$

We can get all the borders that can point to $j'$ from recursive calls to $\rho$
Computing $S_1$

For any $j''$, set pointers for all the borders that points to $j''$.

```plaintext
for j' = 1..m:
    b := ρ''[j']
while b != 0:
    S[m-b] := j'
    b := ρ''[b]
```
Computing $S_1$

For any $j''$, set pointers for all the borders that points to $j''$.

Exercise: Figure out the time complexity of this
Computing $S_2$

For each $j$, the corresponding $j'$ is the length of the longest border of $p$ shorter than $m-j$.

Given the border array $B$, the borders of $p$ are (in decreasing length):

$$B[m], B'[m], B^2[m], ..., B^k[m] = 0$$

for $m$ and some $k$.

Given the border array, this sequence of borders of $p$ can be calculated in linear time, since a border of a border is also a border (think about it!).
Computing $S_2$

Consider a border, $\beta^r[m]$, of $p$.

In the interval $m-\beta^r[m]+1..m-m-\beta^r[m]$, $\beta^r[m]$ is the longest border with length less than $m-j$. The next border, $m-\beta^r[m]$ is too long.

Below $m-\beta^r[m]$, $\beta^r[m]$ is a longer border less than $m-j$. Above $m-\beta^r[m]$, $\beta^r[m]$ is longer than $m-j$. 
A match to an index in this interval should point to border $r$.

In this interval to border $r+1$.
Computing $S_2$

$$S_2[j] = \max_{r} \{ j' \mid j' = B^r[m] \leq m - j \}$$

```
j = 1 ; r = 0
while j <= m:
    while j <= m - \beta^r[m]:
        S_2[j] = \beta^r[m]
        j += 1
    r += 1
```
The Boyer-Moore Algorithm

Preprocessing:
Calculate $R$ and $S$

Main:
i = m
while $i \leq |x|$
    $i, j = hctam(i, m)$
    if $j == 0$: report match at $i+1$
    $i = i + \max(S[j], \max\{m-j+1, R[x[i]]\})$
The Boyer-Moore Algorithm

Preprocessing:
Calculate $R$ and $S$

Main:
$i = m$
while $i \leq |x|:$
    $i,j = \text{hctam}(i,m)$
    if $j == 0$: report match at $i+1$
    $i = i + \max(S[j],R[x[i]])$

Safe since $S[j] \geq m-j+1$ (because $S_2[j]<m$)
Boyer-Moore: Example

$R[a] = 0, R[b] = 1, R[c] = 5$


$p = cbaaba$
Boyer-Moore: Example

i = m

while i <= |x|:
    i, j = hctam(i, m)
    if j == 0: report match at i+1
    i = i + max(S[j], R[x[i]])

\[ x = \text{abbacbaababababacabbbba} \]
\[ p = \text{cbaaba} \]
Boyer-Moore: Example

\[i = m\]
\[\text{while } i \leq |x|:\]
\[i, j = hctam(i, m)\]
\[\text{if } j == 0: \text{ report match at } i+1\]
\[i = i + \max(S[j], R[x[i]])\]
Boyer-Moore: Example

\[ i = m \]
while \( i \leq |x| : \)
\[ i, j = hctam(i, m) \]
\[ \text{if } j == 0: \text{ report match at } i+1 \]
\[ i = i + \max(S[j], R[x[i]]) \]

\[ x = \text{abbacbaabababacabbbba} \]
\[ p = \text{cbaababa} \]
\[ j = 6 \]
Boyer-Moore: Example

```
\[\begin{array}{c|cccccc}
\text{a} & \text{b} & \text{c} & 0 & 1 & 2 & 3 \\
\text{R:} & 0 & 1 & 5 \\
\text{S:} & 12 & 6 & 6 & 6 & 5 & 3 & 1
\end{array}\]

i = m
while i <= |x|:
    i, j = hctam(i, m)
    if j == 0: report match at i+1
    i = i + max(S[j], R[x[i]])
```

\[x=\text{abbacbaabababacabbbba}\]
\[p=\text{cbaabab}\]
Boyer-Moore: Example

\[ i = m \]
\[ \textbf{while} \ i \leq \ |x| : \]
\[ i, j = \text{hctam}(i, m) \]
\[ \textbf{if} \ j == 0: \text{ report match at } i + 1 \]
\[ i = i + \max(S[j], R[x[i]]) \]

---

\[ x = \text{abba}c\text{baa}bababacabbbba \]
\[ p = \text{cba}aaba \]
Boyer-Moore: Example

\[ i = m \]
\[ \text{while } i \leq |x|: \]
\[ i, j = hctam(i, m) \]
\[ \text{if } j == 0: \text{ report match at } i+1 \]
\[ i = i + \max(S[j], R[x[i]]) \]

\[ x = abba\text{cba}abababacabbb\text{ba} \]
\[ p = \text{cba}aba \]
\textbf{Boyer-Moore: Example}

\begin{verbatim}
\textnormal{i = m}
\textbf{while} \textnormal{i} \leq |\textnormal{x}|:
  \textnormal{i,j} = \textbf{hctam}(\textnormal{i,m})
  \textbf{if} \textnormal{j} == 0: report match at \textnormal{i+1}
  \textnormal{i} = \textnormal{i} + \max(\textnormal{S}[\textnormal{j}],\textnormal{R}[\textnormal{x}[\textnormal{i}]])
\end{verbatim}

\begin{verbatim}
\textbf{Boyer-Moore: Example}
\end{verbatim}

\begin{tabular}{c c c c c c c c}
\hline
\texttt{a} & \texttt{b} & \texttt{c} & \texttt{0} & \texttt{1} & \texttt{2} & \texttt{3} & \texttt{4} & \texttt{5} & \texttt{6} \\
\hline
\texttt{R:} & 0 & 1 & 5 & & & & & & \\
\texttt{S:} & 12 & 6 & 6 & 6 & 5 & 3 & 1 & & \\
\hline
\end{tabular}

\begin{align*}
\textbf{x} &= \texttt{abbacbaabababacabbbba} \\
\textbf{p} &= \texttt{cbaaba}
\end{align*}
Boyer-Moore: Example

```
while i <= |x|:
    i, j = hctam(i, m)
    if j == 0: report match at i+1
    i = i + max(S[j], R[x[i]])
```

\[p = \text{cbaaba} \]

\[x = \text{abba} \]

\[i = 4 \]

\[j = 0 \]
Boyer-Moore: Example

\[ i = m \]
\[
\text{while } i \leq |x|: \\
i, j = \text{hctam}(i, m) \\
\text{if } j == 0: \text{ report match at } i+1 \\
i = i + \max(S[j], R[x[i]])
\]

Match at \( i=5 \)

**Example:**

\( x = \text{abba} \text{cba} \text{aba} \text{bab} \text{abacab} \text{bbbba} \)

\( p = \text{cba} \text{aba} \text{ba} \text{a} \)

\( \begin{array}{ccccccc}
  & a & b & c & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
R: & 0 & 1 & 5 & & & & & & & \\
S: & 12 & 6 & 6 & 6 & 5 & 3 & 1 & & & \\
\end{array} \)
Boyer-Moore: Example

\[
\begin{align*}
i &= m \\
\text{while } i &\leq |x|: \\
i, j &= \text{hteam}(i, m) \\
\text{if } j &= 0: \text{report match at } i+1 \\
i &= i + \max(S[j], R[x[i]])
\end{align*}
\]

\[
\begin{array}{ccccccccc}
a & b & c & & & & & & \\
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
R: & 0 & 1 & 5 \\
S: & 12 & 6 & 6 & 6 & 5 & 3 & 1
\end{array}
\]

\[
\begin{align*}
\text{a b c} \\
\text{0 1 2 3 4 5 6} \\
\text{R: 0 1 5} \\
\text{S: 12 6 6 6 5 3 1} \\
\end{align*}
\]

\[
\begin{align*}
\text{x=i=4} \\
\text{a b c a b a a b a b a a b} \\
\text{p=c b a a b a a b a b a}
\end{align*}
\]
Boyer-Moore: Example

\[ \text{i} = m \]

\textbf{while} \ i \ \textless= \ |x|:\n
\textbf{i,j} = hctam(i,m) \\
\textbf{if} \ j == 0: \ \text{report match at} \ i+1 \\
\text{i} = \text{i} + \max(S[j],R[x[i]]) \\

\text{x}=abbacbaababababacabbbba \\
\text{p}=cbaaba
Boyer-Moore: Example

1 = m

while i <= |x|:
    i, j = hctam(i, m)
    if j == 0: report match at i+1
    i = i + max(S[j], R[x[i]])

x = abbacbaabababacabbbba
p = cbaaba

i = 15
j = 5
Boyer-Moore: Example

\[ i = m \]

\[
\text{while } i \leq |x|:\n\]
\[
i, j = hctam(i, m)\]
\[
\text{if } j == 0: \text{ report match at } i+1\]
\[
i = i + \max(S[j], R[x[i]])\]

\[ x = \text{abbacbaabababacabbbba} \]

\[ p = \text{cbaaba} \]
Boyer-Moore: Example

\[
i = m \\
\textbf{while} \  i \leq |x|:\ \\
\quad i,j = \text{htam}(i,m) \\
\quad \textbf{if} \  j = 0: \ \text{report match at} \ i+1 \\
\quad i = i + \max(S[j],R[x[i]])
\]

\(x=abbacbaabababacabbbba\)

\(p=cbaaba\)
Boyer-Moore: Example

\[ i = m \]
\[ \textbf{while } i \leq |x|: \]
\[ i, j = \text{hctam}(i, m) \]
\[ \textbf{if } j == 0: \text{ report match at } i+1 \]
\[ i = i + \text{max}(S[j], R[x[i]]) \]

\[ x = \text{abbacbaaabababacabbbba} \]
\[ p = \text{cbaababa} \]
\[ i = 20 \]
\[ j = 6 \]
Boyer-Moore: Example

\[ i = m \]

\textbf{while} \quad i \leq |x|:\n
\quad i, j = hctam(i, m)\n
\textbf{if} \quad j == 0: \text{ report match at } i+1\n
\quad i = i + \max(S[j], R[x[i]])\n
\begin{align*}
\text{x} &= \text{abbacbaababababacabbbba} \\
\text{p} &= \text{cbaababa} \\
\text{R:} &= \begin{array}{cccccccc} a & b & c & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\end{array} \\
\text{S:} &= \begin{array}{cccccccc} 12 & 6 & 6 & 6 & 5 & 3 & 1 \\
\end{array} \\
\end{align*}
Boyer-Moore: Example

\[i = m\]

\[
\text{while } i \leq |x|:\]

\[
i, j = hctam(i, m)\]

\[
\text{if } j == 0: \text{ report match at } i+1\]

\[
i = i + \max(S[j], R[x[i]])\]

\[x=\text{abbacbaabaabababacabbbba}\]

\[p=cbaababa\]

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

R: 0 1 5
S: 12 6 6 6 5 3 1

\[x[i]=b\]

\[j=6\]

\[i=21\]
Boyer-Moore: Example

```python
i = m
while i <= |x|:
    i, j = hctam(i, m)
    if j == 0: report match at i+1
    i = i + max(S[j], R[x[i]])
```

```plaintext
x=abbacbaabababacabbbba
p=cbaaba
```

```
a b c 0 1 2 3 4 5 6
R: 0 1 5
S: 12 6 6 6 5 3 1
```

```
i=19
j=4
```
Boyer-Moore: Example

\[ i = m \]

```
while i <= |x|:
    i, j = hctam(i, m)
    if j == 0: report match at i+1
    i = i + max(S[j], R[x[i]])
```

\[ x = abbacbaababababacabb \]

\[ p = cbaaba \]

\[ R: \]
\[
0 1 2 3 4 5 6
\]

\[ S: \]
\[
12 6 6 6 5 3 1
\]
i = m
while i <= |x|:
    i, j = hctam(i, m)
    if j == 0: report match at i+1
    i = i + max(S[j], R[x[i]])

Boyer-Moore: Example

\[
\begin{array}{ccc}
\text{i} & \text{c} & \text{b} & \text{a} & \text{c} & \text{b} & \text{a} & \text{b} & \text{a} & \text{a} & \text{b} & \text{a} \\
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\text{R:} & 0 & 1 & 5 \\
\text{S:} & 12 & 6 & 6 & 6 & 6 & 5 & 3 & 1
\end{array}
\]

\[
\begin{array}{c}
\text{x[24]} = \text{b} \\
\text{j=4}
\end{array}
\]

\[
\begin{array}{c}
\text{x=abbacbaabaabababacabbbba} \\
\text{p=cbaaba}
\end{array}
\]
Number of comparisons

\[ x = \text{abbacbaaababababacabbbba} \]
\[ 000023211100001100121 = 16 \]
\[ |x| = 21 \]

We found all occurrences of \( p \) in \( x \) in \textit{sub linear} time!
Runtime for Boyer-Moore

- The worst-case time complexity is still as the simple algorithm: $O(|x|m)$
  - Consider searching for $p=a^m$ in the string $x=a^n$
  - but see Chap. 8 for a linear time version
- Often sub-linear runtime on “average” data
Comparison of the algorithms

- **Knuth-Morris-Pratt:**
  - Always linear
  - Deals with repetitive strings as with other strings

- **Boyer-Moore:**
  - On “average” *sub-linear!*
  - Problems with repetitions in strings
  - Worst case $O(n^2)$
  - Fastest in practice for many applications
A bit-vector SHIFT-and-OR approach to exact pattern matching

\[ x = \texttt{bbba\textcolor{red}{c}bbba\textcolor{green}{ab}abacab\textcolor{green}{bb}ba} \]
“Sprint rather than plod!”

• An algorithm that does **not** attempt to reduce the number of comparisons
  - Essentially the “simple” $O(|p||x|)$ algorithm

• Instead the trick is to make each comparison very fast
Using a “state vector” $s$

We define a vector $s$ – the state of matching so far – by:

$$s[j] = 0 \text{ iff } x[i-j+1 .. i] = p[1 .. j]$$
Using a “state vector” $s$

Notice that $s$ holds information about more than one comparison!

Conceptually, $p$ is positioned $|p|$ places along $x$: $s$ tries to match $p$ at positions $i-|p|+1 .. i$
Using a “state vector” \( s \)

When \( s[|p|] == 0 \) we have an occurrence of \( p \) in \( x \) at \( i-|p|+1 \)
Example

\[ i = 0 \]

\[ \mathbf{p} = \text{bbba} \]

\[ \mathbf{p} = \text{bbba} \]

\[ \mathbf{p} = \text{bbba} \]

\[ \mathbf{p} = \text{bbba} \]

\[ \mathbf{x} = \text{bbbacbbbababacabbbba} \]

\[ s[1] = 1 \]

\[ s[2] = 1 \]

\[ s[3] = 1 \]

\[ s[4] = 1 \]

\[ s = 1111 \]
Example

\[ \begin{align*}
  i &= 1 \\
  x &= \text{bbbacbbbababacabbbba} \\
  p &= \text{bbba} \quad s[1] == 0 \\
  p &= \text{bbba} \quad s[2] == 1 \\
  p &= \text{bbba} \quad s[3] == 1 \\
  p &= \text{bbba} \quad s[4] == 1 \\
  s &= 0111
\end{align*} \]
Example

\[ x = \text{bbbacbbbababacabbbba} \]

\[ i = 2 \]

\[ p = \text{bbba} \]
\[ s[1] = 0 \]

\[ p = \text{bbba} \]
\[ s[2] = 0 \]

\[ p = \text{bbba} \]
\[ s[3] = 1 \]

\[ p = \text{bbba} \]
\[ s[4] = 1 \]

\[ s = 0011 \]
Example

$x = \text{bbbacbbbababacabbbbba}$

$p = \text{bbba}$

$s[1] == 0$

$p = \text{bbba}$

$s[2] == 0$

$p = \text{bbba}$

$s[3] == 0$

$p = \text{bbba}$

$s[4] == 1$

$s = 0001$
Example

\[ x = \text{bbbacbbbababacabbbba} \]

\[ p = \text{bbba} \]

\[ s[1] = 1 \]

\[ p = \text{bbba} \]

\[ s[2] = 1 \]

\[ p = \text{bbba} \]

\[ s[3] = 1 \]

\[ p = \text{bbba} \]

\[ s[4] = 0 \]

\[ s = 1110 \]

Match at \( i-4+1 = 1 \)
Example

\[ i = 5 \]

\[ x = \text{bbbacbbbababacabbbba} \]
\[ p = \text{bbba} \quad s[1] = 1 \]
\[ p = \text{bbba} \quad s[2] = 1 \]
\[ p = \text{bbba} \quad s[3] = 1 \]
\[ p = \text{bbba} \quad s[4] = 1 \]
\[ s = 1111 \]
Example

$$x = \text{bbbacbbbbababacabbbba}$$

$$i = 6$$

$$p = \text{bbba} \quad s[1] = 0$$

$$p = \text{bbba} \quad s[2] = 1$$

$$p = \text{bbba} \quad s[3] = 1$$

$$p = \text{bbba} \quad s[4] = 1$$

$$s = 0111$$
Example

\[ x = \text{bbbacbbbababacabbbbba} \]
\[ p = \text{bbba} \]
\[ s[1] = 0 \]
\[ p = \text{bbba} \]
\[ s[2] = 0 \]
\[ p = \text{bbba} \]
\[ s[3] = 1 \]
\[ p = \text{bbba} \]
\[ s[4] = 1 \]
\[ s = 0011 \]
Let $\mathbf{s}^i$ be the state vector in iteration $i$. Then $\mathbf{s}^i[j]=\mathbf{s}^{i-1}[j-1]$ OR $t$ where $t=0$ if $p[j]=x[i]$ and $t=1$ otherwise.
Let $s^i$ be the state vector in iteration $i$. Then $s^i[j] = s^{i-1}[j-1] \text{ OR } t$ where $t = 0$ if $p[j] = x[i]$ and $t = 1$ otherwise.
Special cases...

- For this to work:
  - $s^0 = 01|p|$
  - $s^i[0] = 0$ for all $i$

  No non-empty substring matches the empty prefix of $x$

  $s[1] == 0$ iff $p[1] == x[1]$, regardless of previous $s$
Bit-matrix for $t$

The bit $t$, where $t=0$ if $p[j]=x[i]$ and $t=1$ otherwise can be pre-calculated and stored in a bit-matrix:

$$t[h,j] = \begin{cases} 0 & \text{if } p[j] == h \\ 1 & \text{if } p[j] != h \end{cases}$$

with rows indexed by the alphabet and columns indexed by indices in $p$
Bit-matrix $\mathbf{t}$ for $p=\text{bbba}$

<table>
<thead>
<tr>
<th>column</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbf{t}[\text{'a'},]$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$\mathbf{t}[\text{'b'},]$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$\mathbf{t}[\text{'c'},]$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Example

\[ x = \text{bbbacbbbababacabbbba} \]

\[ p = \text{bbba} \]
\[ s^0[0] = 0 \]
\[ p = \text{bbba} \]
\[ s^0[1] = 1 \]
\[ p = \text{bbba} \]
\[ s^0[2] = 1 \]
\[ p = \text{bbba} \]
\[ s^0[3] = 1 \]
\[ p = \text{bbba} \]
\[ s^0[4] = 1 \]
\[ s = 011111 \]
Example

\[ x = \text{bbbacbbbababacabbbba} \]

\[ p = \text{bbba} \]
\[ s^1[0] == 0 \]
\[ s^1[1] == 0 \]
\[ s^1[2] == 1 \]
\[ s^1[3] == 1 \]
\[ s^1[4] == 1 \]

\[ s^0[0] == 0 \]
\[ s^0[1] == 1 \]
\[ s^0[2] == 1 \]
\[ s^0[3] == 1 \]
\[ s^0[4] == 1 \]

\[ s = 00111 \]
Example

\[
x = \text{bbbacbbbababacabbbba}
\]

\[
i = 2
\]

\[
p = \text{bbba}
\]

\[
p = \text{bbba}
\]

\[
p = \text{bbba}
\]

\[
p = \text{bbba}
\]

\[
t[b]
\]

\[
s^2[0] == 0
\]

\[
s^2[1] == 0
\]

\[
s^2[2] == 0
\]

\[
s^2[3] == 1
\]

\[
s^2[4] == 1
\]

\[
s^1[0] == 0
\]

\[
s^1[1] == 0
\]

\[
s^1[2] == 1
\]

\[
s^1[3] == 1
\]

\[
s^1[4] == 1
\]

\[
s = 00011
\]
Example

\[ x = \text{bbbacbbbababacabbbba} \]

\[ i = 3 \]

\[ t[b] \]

\[ p = \text{bbba} \]
\[ p = \text{bbba} \]
\[ p = \text{bbba} \]
\[ p = \text{bbba} \]
\[ s = 00001 \]
Example

\[ x = bbbacbbbababacacabbbba \]

\[ i = 4 \]

\[ p = bbbba \]
\[ p = bbba \]
\[ p = bbba \]
\[ p = bbba \]
\[ p = bbba \]

\[ s^4[0] == 0 \]
\[ s^4[1] == 1 \]
\[ s^4[2] == 1 \]
\[ s^4[3] == 1 \]
\[ s^4[4] == 0 \]

\[ s^3[0] == 0 \]
\[ s^3[1] == 0 \]
\[ s^3[2] == 0 \]
\[ s^3[3] == 0 \]
\[ s^3[4] == 1 \]

\[ t[0] \]
\[ t[1] \]
\[ t[2] \]
\[ t[3] \]
\[ t[4] \]

\[ s = 011110 \]
Example

\[ x = \text{bbbacbbbababacabbbba} \]

\[ i = 5 \]

\[ s^5[0] == 0 \]
\[ s^5[1] == 1 \]
\[ s^5[2] == 1 \]
\[ s^5[3] == 1 \]
\[ s^5[4] == 1 \]

\[ s^4[0] == 0 \]
\[ s^4[1] == 1 \]
\[ s^4[2] == 1 \]
\[ s^4[3] == 1 \]
\[ s^4[4] == 0 \]

\[ t[c] \]

\[ s = 011111 \]
Example

\[ i = 6 \]

\[ x = bbbacbbbababacabbbba \]

\[ p = bbbba \]
\[ p = bbba \]
\[ p = bbba \]
\[ p = bbba \]
\[ p = bbba \]

\[ t[b] \]
\[ s^6[0] = 0 \]
\[ s^6[1] = 0 \]
\[ s^6[2] = 1 \]
\[ s^6[3] = 1 \]
\[ s^6[4] = 1 \]
\[ s^5[0] = 0 \]
\[ s^5[1] = 1 \]
\[ s^5[2] = 1 \]
\[ s^5[3] = 1 \]
\[ s^5[4] = 1 \]

\[ s = 001111 \]
The SHIFT–and–OR Algorithm

Preprocessing:
for c in α and j=1..|p|:
    t[c,j] = 1
for j=1..|p|:
    t[p[j],j] = 0

Main:
  s = 01^{|p|}
for i=1..|x|:
  s = (s >> 1) | t[x[i]]
  if s[|p|]==0: report i−|p|+1 as match
Time usage

- Preprocessing takes time $O(|\alpha||p|)$
- Main search takes time $O(|x||p|)$
Bit-operations

• If the word size is $w$ we can usually do bit-operations on $w$ bits in constant time
  - shift $w$ bits in time $O(1)$
  - OR $w$ bits in time $O(1)$

• Manipulating larger bit-vectors can be broken down into $w$ sized chunks
  - initialize $|p|$ long bit-vector to all 1s in time $O(|p|/w)$
  - shift $|p|$ long bit-vector in time $O(|p|/w)$
  - OR $|p|$ long bit-vector in time $O(|p|/w)$
Time usage (redux)

• Preprocessing takes time $O(|\alpha||p|/w + |p|)$
• Main search takes time $O(|x||p|/w)$
• For small $|p|$ and $|\alpha|$ this approaches a search-time $O(|x|)$ with very little overhead
• For large $|p|$ and $|\alpha|$ the approach is not advisable