Exact pattern matching
Knuth-Morris-Pratt and Boyer-Moore

\[ x = \text{abbac} \text{bba} \text{babacabbbba} \]

6 \quad 17
What is a border?
What is a border?

A border of a string $x$ is any *proper prefix* of $x$ that equals a suffix of $x$. 

abaabbbbaabaab

ab         ab

abaab       ababaab
What is a border array?
What is a border array

In the border array $B[1..n]$ of $x[1..n]$, entry $B[i]$ is the length of the longest border of $x[1..i]$. 

![Diagram showing the concept of border array]

$B[i]$  

$i$  

---

**Diagram:**

- A horizontal line represents the string $x[1..n]$.
- Two vertical lines mark the positions of $B[i]$ and $i$.
- The green and blue sections indicate the longest border of $x[1..i]$. 

---
Exact pattern matching

Given string $x = abbacbbbababacabbbba$ and pattern $p = bbba$
find all occurrences of $p$ in $x$

$x = abbacbbbababacabbbba$

$m = |p|$
Simple algorithm

for $i=1..|x|:
  for $j=1..m+1$:
    if $x[i+j-1]! = p[j]$: break
  if $j==m+1$: report $i$ as match

Scan along $x$                      Try to match $p$
Example

\[
\text{for } i=1..|x|:
\text{for } j=1..m+1:
\quad \text{if } x[i+j-1] \neq p[j]: \text{ break}
\text{if } j==m+1: \text{ report } i \text{ as match}
\]

\[
\begin{align*}
\text{x} &= \text{abbacbbbababacabbbba} \\
\text{p} &= \text{bbba}
\end{align*}
\]
Example

for $i=1..|x|$
    for $j=1..m+1$:
        if $x[i+j-1] \neq p[j]$: break
    if $j==m+1$: report $i$ as match

$x=abbbacbbbababacabbbabbaa$

$p=bbba$

i=1

i+j-1=1

j=1
for i=1..|x|:
    for j=1..m+1:
        if x[i+j-1]!=p[j]: break
    if j==m+1: report i as match

x=abbacbbbabababacabbbba

i=2

p=bbba
Example

for \( i = 1 \ldots |x| \):
    for \( j = 1 \ldots m+1 \):
        if \( x[i+j-1] \neq p[j] \): break
    if \( j == m+1 \): report \( i \) as match

\( x = abbacbbbababacabbbba \)

\( p = bbba \)
for i=1..|x|:
  for j=1..m+1:
    if x[i+j−1]!=p[j]: break
  if j==m+1: report i as match

x=abbbacbbbabababacabbbba
p=bbba

i=2
i+j−1=2
j=1
Example

for i = 1..|x|:
    for j = 1..m+1:
        if x[i+j-1] != p[j]: break
    if j == m+1: report i as match

x = abbacbbbababacababbba
p = bbba

i = 2
i + j - 1 = 3
j = 2
for $i=1..|x|:$
    for $j=1..m+1:$
        if $x[i+j-1] \neq p[j]$: break
    if $j==m+1$: report $i$ as match

$x=abba|cbbbababacabbbba$

$p=bbba$
for i=1..|x|:
    for j=1..m+1:
        if x[i+j-1]!=p[j]: break
    if j==m+1: report i as match

\begin{equation}
x=abba\textcolor{red}{c}bba\textcolor{green}{b}ababa\textcolor{red}{b}acabb\textcolor{green}{b}bbaa
\end{equation}

\begin{equation}
p=bb\textcolor{green}{b}a
\end{equation}
Example

for $i=1..|x|$: 
  for $j=1..m+1$: 
    if $x[i+j-1]! = p[j]$: break 
  if $j==m+1$: report $i$ as match

$x=abba|cbbbabababacabbbba$

$p=bbba$

i=4

i+j-1=4

j=1
Example

For i = 1..|x|:
   for j = 1..m+1:
      if x[i+j-1] != p[j]: break
   if j == m+1: report i as match

x = abbaacbabbababacabbbba
p = bbba

i = 5
i+j-1 = 5
j = 1
Example

```
for i=1..|x|:
    for j=1..m+1:
        if x[i+j-1]! = p[j]: break
        if j==m+1: report i as match
```

\[ x = abbacbbbababacabbbba \]
\[ p = bbba \]
Example

for i=1..|x|:
    for j=1..m+1:
        if x[i+j-1]!=p[j]: break
    if j==m+1: report i as match

\[x=abbac\textcolor{green}{bbbababacabbbba}\]
\[p=bbba\]
Example

for i=1..|x|:
  for j=1..m+1:
    if x[i+j-1]!=p[j]: break
  if j==m+1: report i as match

x=abbacbbbababacabbbba

i=6

i+j-1=8

p=bbba

j=3
for $i=1..|x|:\n  \text{for } j=1..m+1:\n    \text{if } x[i+j-1] \neq p[j]: \text{ break}\n  \text{if } j==m+1: \text{ report } i \text{ as match}$

$x=abbacbbbababacabbbbba$

$p=bbba$

$i=6$

$i+j-1=9$

$j=4$
for $i=1..|x|$:  
  for $j=1..m+1$:  
    if $x[i+j-1] \neq p[j]$: break  
    if $j==m+1$: report $i$ as match

$x=\text{abbacbbbababacabbbba}$

$p=bbbba$

$\text{Match at } i=6$
Running time

- The running time is $O(|x|m)$
- But notice that we compare strings we *know* do not match

If $p$ matches here

$x=abbacbbbababacabbbba$

It cannot match here

$p=bbba$
Observation

If a prefix of length $h$ of $p$ matches at index $i$

then $p$ cannot match at $i'$ in $[i,i+h]$

unless $p[1..h-(i'-i)]$ equals $p[i'-i..h]$
i.e. $p[1..h-(i'-i)]$ is a border of $p[1..h]$
Trick: Using borders

If mismatch at \( x[i+h+1]:p[h+1] \):

\[
x = gcggcacttaactgattagacagtaagac... \\
p = acttaactcgc
\]

Then shift \( p \) to align the “prefix” border with the previous “suffix” border of \( p[1..h] \):

\[
x = gcggcacttaactgattagacagtaagac... \\
p = acttaactcgc \\
p = acttaactcgc
\]
Trick: Using borders

If we don't move the index into $x$, the conceptual shift of $p$ is just moving the index in $p$ to $B[h]+1$

\[ x = \text{gcggcacttaactgattagacagtaagac...} \]
\[ p = \text{acttaactcgc} \]

\[ p = \text{acttaactcgc} \]
Trick: Using borders

If we don't move the index into $x$, the conceptual shift of $p$ is just moving the index in $p$ to $B[h]+1$.

$x = gcggcacttaactgattagacagtaagac...$

$p = acttaactcgc$

Point in $x$ where we have matched so far

$p = acttaactcgc$

$p = acttaactcgc$
Trick: Using borders

If we don't move the index into $x$, the conceptual shift of $p$ is just moving the index in $p$ to $B[h]+1$

$x = gcggcacttaactgattagacagtaagac...$

$p = acttaactcgc$

Point in $x$ where we have matched so far

Point in $p$ where we have matched so far
Trick: Using borders

If we don't move the index into \( x \), the conceptual shift of \( p \) is just moving the index in \( p \) to \( B[h]+1 \)

\[
x = \text{gcttgactgtagacagtaagac...} \\
p = \text{acttaacctcgc} \\
p = \text{acttaacctcgc}
\]

Never decreased

Decreased when we have a mismatch
Knuth-Morris-Pratt

Preprocessing:
build border array $B$
$B'[1] = 0$
for $j=2..m+1$:
  $B'[j] = B[j-1]+1$

Main:
i = 1; j = 1
while $i <= |x|-m+j$:
  $i,j = \text{match}(i,j,m)$
  if $j==m+1$: report match at $i-m$
  if $j==1$: $i = i + 1$
  else: $j = B'[j]$

Help routine:
\[
\text{match}(i,j,m):
  \text{while } x[i]==p[j] \text{ and } j <= m:
    i = i + 1
    j = j + 1
  \text{return } i,j
\]
**Example – preprocessing**

**Preprocessing:**

build border array $B$

$B'[1] = 0$

for $j=2..m+1$:

$B'[j] = B[j-1]+1$

$p = bbba$

$B = 0120$

$B' = 01231$
Example

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

while \( i \leq |x| - m + j \):
  \[ i, j = \text{match}(i, j, m) \]
  \[
  \text{if } j==m+1:
    \text{report match at } i-m
  \]
  \[
  \text{if } j==1: \ i = i + 1
  \]
  else:
    j = \text{B}'[j]

B' = 01231

\[
x = \text{abbacbbbababacabbbba}
\]

\[
p = \text{bbba}
\]

\[
i = 1
\]

\[
j = 1
\]

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

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

\[
\text{and } j <= m:
  \]

\[
i = i + 1
\]

\[
j = j + 1
\]

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

```
i = 1; j = 1
while i <= |x| - m + j:
    i, j = match(i, j, m)
    if j == m + 1:
        report match at i - m
    if j == 1: i = i + 1
    else:
        j = B'[j]
match(i, j, m):
    while x[i] == p[j]
        and j <= m:
            i = i + 1
            j = j + 1
    return i, j

i=1
\[x=abbacbbbababacabbbba\]
p=\[bbba\]
```
Example

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

**Algorithm**

```
while i <= |x| - m + j:
    i, j = match(i, j, m)
    if j == m + 1:
        report match at i - m
    if j == 1: i = i + 1
    else: j = B'[j]
```

```
def match(i, j, m):
    while x[i] == p[j] and j <= m:
        i = i + 1
        j = j + 1
    return i, j
```

**Input**

\[
x = \text{abbacbbbababacabbbba}
\]

\[
p = \text{bbba}
\]

**Output**

\[
B' = 01231
\]
Example

$$B' = 01231$$

```python
i = 1; j = 1
while i <= |x| - m + j:
    i, j = match(i, j, m)
    if j == m + 1:
        report match at i - m
    if j == 1: i = i + 1
    else: j = B'[j]
```

```python
match(i, j, m):
    while x[i] == p[j] and j <= m:
        i = i + 1
        j = j + 1
    return i, j
```

```
x = abbacbbbababacabbbba
p = bbba
```
Example

B' = 01231

i = 1; j = 1
while i <= |x|-m+j:
    i,j = match(i,j,m)
if j==m+1:
    report match at i-m
if j==1: i = i + 1
else:    j = B'[j]

match(i,j,m):
    while x[i]==p[j]
    and j <= m:
        i = i + 1
        j = j + 1
    return i,j
Example

\begin{align*}
i &= 1; \ j &= 1 \\
\textbf{while} \ i &\leq |x| - m + j: \\
&\quad i, j = \text{match}(i, j, m) \\
&\quad \textbf{if} \ j == m + 1: \\
&\quad\quad \text{report match at } i-m \\
&\quad \textbf{if} \ j == 1: \ i = i + 1 \\
&\textbf{else:} \ \ j = B'[j] \\
\end{align*}

\begin{align*}
\text{match}(i, j, m): \\
&\quad \textbf{while} \ x[i] == p[j] \\
&\quad\quad \textbf{and} \ j \leq m: \\
&\quad\quad\quad i = i + 1 \\
&\quad\quad\quad j = j + 1 \\
&\quad \textbf{return} \ i, j \\
\end{align*}

\[x=abba\textcolor{red}{bc}bbababacabbbba\]
\[p=bb\textcolor{red}{ba}\]

\[B' = 01231\]
Example

\( i = 1; \ j = 1 \)
\[
\text{while } i \leq |x| - m + j:
\quad i, j = \text{match}(i, j, m)
\quad \text{if } j == m + 1:
\quad \quad \text{report match at } i - m
\quad \text{if } j == 1: \ i = i + 1
\quad \text{else: } \ j = B'[j]
\]

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

\[
B' = 01231
\]

match \((i, j, m)\): \\
\[
\quad \text{while } x[i] == p[j] \text{ and } j \leq m:
\quad \quad i = i + 1
\quad \quad j = j + 1
\quad \text{return } i, j
i = 1; j = 1
while i <= |x| - m + j:
    i, j = match(i, j, m)
    if j == m + 1:
        report match at i - m
    if j == 1: i = i + 1
else:
    j = B'[j]

match(i, j, m):
    while x[i] == p[j] and j <= m:
        i = i + 1
        j = j + 1
    return i, j

i = 4
x = a b b a c b b b b a b a b a c a b a b a b b b a a
p = b b b a
j = 2
B' = 0 1 2 3 1
Example

\[
\begin{align*}
    i &= 1; j = 1 \\
    \textbf{while} & \quad i \leq |x| - m + j: \\
        & \quad i, j = \text{match}(i, j, m) \\
        & \quad \textbf{if} \quad j = m + 1: \\
            & \quad \quad \text{report match at } i - m \\
        & \quad \textbf{if} \quad j = 1: \quad i = i + 1 \\
        & \quad \textbf{else:} \quad j = B'[j]
\end{align*}
\]

\[
\begin{align*}
    \text{match}(i, j, m): & \\
        & \quad \textbf{while} \quad x[i] = p[j] \\
            & \quad \quad \textbf{and} \quad j \leq m: \\
                & \quad \quad i = i + 1 \\
                & \quad \quad j = j + 1 \\
        & \quad \textbf{return} \quad i, j
\end{align*}
\]

\[
\begin{align*}
    x &= \text{abba}cbb\text{bababacabbbba} \\
    p &= \text{bbba}
\end{align*}
\]
Example

\[ B' = 01231 \]

\[
i = 1; \ j = 1 \\
\text{while } i \leq |x| - m + j: \\
i, j = \text{match}(i, j, m) \\
\text{if } j == m + 1: \\
    \text{report match at } i - m \\
\text{if } j == 1: \ i = i + 1 \\
\text{else: } \ j = B'[j]
\]

\[
\text{match}(i, j, m): \\
    \text{while } x[i] == p[j] \\
    \quad \text{and } j \leq m: \\
    \quad \ i = i + 1 \\
    \quad \ j = j + 1 \\
    \text{return } i, j
\]

\[
x = abba\textcolor{red}{c}b\textcolor{red}{b}b\textcolor{red}{b}aba\textcolor{red}{b}aba\textcolor{red}{b}ac\textcolor{red}{a}b\textcolor{red}{b}b\textcolor{red}{b}ba \\
p = bbba
\]

i=4
\[
\downarrow
\]
\[
x = abba\textcolor{red}{c}b\textcolor{red}{b}b\textcolor{red}{b}aba\textcolor{red}{b}aba\textcolor{red}{b}ac\textcolor{red}{a}b\textcolor{red}{b}b\textcolor{red}{b}ba \\
p = bbba
\]

j=1
Example

\[
i = 1; \ j = 1
\]
\[
\text{while } i \leq |x| - m + j:
\]
\[
i, j = \text{match}(i, j, m)
\]
\[
\text{if } j == m + 1:
\]
\[
\text{report match at } i-m
\]
\[
\text{if } j == 1: i = i + 1
\]
\[
\text{else: } j = B'[j]
\]

\[
machine(i, j, m):
\]
\[
\text{while } x[i] == p[j] \text{ and } j \leq m:
\]
\[
i = i + 1
\]
\[
j = j + 1
\]
\[
\text{return } i, j
\]

\[
\begin{align*}
B' &= \begin{array}{cccccc}
0 & 1 & 2 & 3 & 1 \\
\end{array}
\end{align*}
\]

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

\[
i = 5
\]
\[
j = 1
\]
i = 1; j = 1  
while i <= |x|−m+j:  
    i, j = match(i, j, m)  
    if j == m+1:  
        report match at i−m  
    if j == 1: i = i + 1  
else: j = B'[j]

match(i, j, m):
    while x[i] == p[j] and j <= m:
        i = i + 1
        j = j + 1
    return i, j

x = abbaaacbbbabababacabbbbaabba
p = bbba

B' = 01231
Example

\[
i = 1; \ j = 1
\]
\[
\text{while } i \leq |x| - m + j:
\]
\[
i, j = \text{match}(i, j, m)
\]
\[
\text{if } j == m + 1:
\]
\[
\text{report match at } i - m
\]
\[
\text{if } j == 1: \ i = i + 1
\]
\[
\text{else: } \ j = B'[j]
\]

\[
\begin{align*}
\text{x} &= \text{abbacbbbababacabbbba} \\
\text{p} &= \text{bbba}
\end{align*}
\]

\[
B' = 01231
\]

\[
\text{match}(i, j, m):
\]
\[
\text{while } x[i] == p[j] \text{ and } j \leq m:
\]
\[
\quad i = i + 1 \\
\quad j = j + 1
\]
\[
\text{return } i, j
\]

\[
i = 6
\]
\[
j = 1
\]
i = 1; j = 1
while i <= |x| - m + j:
    i, j = match(i, j, m)
    if j == m + 1:
        report match at i - m
    elif j == 1:
        i = i + 1
    else:
        j = B'[j]
match(i, j, m):
    while x[i] == p[j] and j <= m:
        i = i + 1
        j = j + 1
    return i, j

x=abbacbbababacabbbba
p=bbba

B' = 01231

i=10

j=5
Example

\begin{align*}
i &= 1; \ j &= 1 \\
\text{while } &i \leq |x| - m + j: \\
&\quad i, j = \text{match}(i, j, m) \\
&\quad \text{if } j = m + 1:\ \\
&\quad\quad \text{report match at } i-m \\
&\quad \text{if } j = 1: \ i = i + 1 \\
&\quad \text{else:} \ j = B'[j]
\end{align*}

\begin{align*}
\text{match}(i, j, m): \\
\quad &\text{while } x[i] = p[j] \\
&\quad \quad \text{and } j \leq m: \\
&\quad\quad i = i + 1 \\
&\quad\quad j = j + 1 \\
&\quad \text{return } i, j
\end{align*}

\[
x = \text{abbacbbababacabbbba} \quad p = \text{bbba}
\]

Match at \(i=6\)
\[
\begin{align*}
\text{i} &= 1; \text{j} = 1 \\
\text{while } \text{i} \leq |\text{x}|-\text{m}+\text{j}: \\
&\quad \text{i}, \text{j} = \text{match}(\text{i}, \text{j}, \text{m}) \\
&\quad \text{if } \text{j} == \text{m}+1: \\
&\quad\quad \text{report match at } \text{i} - \text{m} \\
&\quad \text{if } \text{j} == 1: \text{i} = \text{i} + 1 \\
&\quad \text{else: } \text{j} = \text{B}'[\text{j}] \\
\text{match}(\text{i}, \text{j}, \text{m}): \\
&\quad \text{while } \text{x}['i'] == \text{p}['j'] \\
&\quad \quad \text{and } \text{j} \leq \text{m}: \\
&\quad \quad \quad \text{i} = \text{i} + 1 \\
&\quad \quad \quad \text{j} = \text{j} + 1 \\
&\quad \text{return } \text{i}, \text{j}
\end{align*}
\]

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

• Time bounded by the sum of matches and mismatches
  - Why?
Running time

- Time bounded by the sum of matches and mismatches
  - Each match increases $i$
  - Each mismatch either increases $i$ or $p$’s position (conceptually)
Running time

• Time bounded by the sum of matches and mismatches
  - Each match increases i
  - Each mismatch either increases i or p’s position (conceptually)
  - Matches bounded by n
  - Mismatches bounded by n+(n-m)

• Runtime in O(n)
Variation on the simple algorithm

\begin{verbatim}
\textcolor{red}{i = m}
\textcolor{red}{
while } \textcolor{blue}{i} \textcolor{red}{\leq} \textcolor{blue}{|x|}:
\hspace{1em} \textcolor{blue}{i,j} = \textcolor{blue}{hctam(i,m)}
\hspace{1em} \textcolor{green}{\textbf{if } j==0: \text{ report } i+1 \text{ as match}}
\hspace{1em} \textcolor{blue}{i} = \textcolor{blue}{i + m - j + 1}
\hspace{1em}
\textcolor{blue}{hctam(i,j):}
\hspace{1em} \textcolor{red}{\textbf{while } x[i] == p[j] \textbf{ and } j \geq 0:}
\hspace{2em} \textcolor{blue}{i} = \textcolor{blue}{i - 1}
\hspace{2em} \textcolor{blue}{j} = \textcolor{blue}{j - 1}
\hspace{2em}
\hspace{1em} \textcolor{red}{\textbf{return } i,j}
\end{verbatim}

\textcolor{blue}{x=abbcabbbbababacabbbbaa}
\textcolor{green}{p=bbba}
Variation on the simple algorithm

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

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

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

\[
i = 1
\]
\[
\downarrow
\]
\[
x = abba c b b b a b a b a c a b b b a b a
\]
\[
p = bbba
\]
\[
\uparrow
\]
\[
j = 1
\]
Variation on the simple algorithm

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

\[
i = m
\]

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

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

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

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

\[
hctam(i, j):
\]

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

\[
i = i - 1
\]

\[
j = j - 1
\]

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

\[
i = 5
\]

\[
x = abba c bbbabababacabbbba
\]

\[
p = bbb\text{a}
\]

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

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

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

\[ i=6 \]

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

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

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

\[ x=\text{abbac}bbbababacabbbba \]
\[ p=\text{bbba} \]

\[ i=6 \]
\[ j=4 \]
Variation on the simple algorithm

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

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

\[
\begin{align*}
i &= 7 \\
x &= \text{abbacbbbababacabbba} \\
p &= \text{bbba}
\end{align*}
\]
Variation on the simple algorithm

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

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

\begin{align*}
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 \\
\end{align*}

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

\begin{align*}
i &= 8 \\
x &= \text{abbacbbbababacabbbba} \\
p &= \text{bbba} \\
\end{align*}
Variation on the simple algorithm

\[ i = m \]

while \( i \leq |x| \):
    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 \( j \geq 0 \):
        i = i - 1
        j = j - 1
    return i, j

\[ x = \text{abbacbbbababacabbbbba} \]
\[ p = \text{bbba} \]
\[ i = 8 \]
\[ j = 4 \]
Variation on the simple algorithm

```
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

i=9

x=abbacbbbababacabbbba
p=bbba
```
Variation on the simple algorithm

\[ i = m \]

\begin{itemize}
\item \textbf{while} \( i \leq |x| \):
  \item \( i, j = hctam(i, m) \)
  \item \textbf{if} \( j == 0 \): report \( i+1 \) as match
  \item \( i = i + m - j + 1 \)
\end{itemize}
Variation on the simple algorithm

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

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

\[ x=abbacbbbababacabbbba \]
\[ p=bbba \]

Match at i=6
Variation on the simple algorithm

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

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

\[
i=10 \\
\textbf{x=abbacbbbababacabbbba} \\
\textbf{p=bbba}
\]
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, \( \mathbf{R} \), such that for each letter \( a \), \( \mathbf{R}[a] \) is the distance from the right of \( p \) to the rightmost occurrence of \( a \) in \( p \), or \( m \) if \( a \) is not in \( p \)

\[
\text{for } a \text{ in } \alpha:\n\quad \mathbf{R}[a] = m
\]

\[
\text{for } j = 1..m:\n\quad \mathbf{R}[p[j]] = m-j
\]

\( \mathbf{p} = \text{bbba} \)
\( \alpha = \{a,b,c\} \)

\( \mathbf{R}[a] = 0 \quad \mathbf{p} = \text{bbba} \)
\( \mathbf{R}[b] = 1 \quad \mathbf{p} = \text{bbba} \)
\( \mathbf{R}[c] = 4 \quad \mathbf{p} = \text{bbba} \)
“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 $p$

\[
\text{for } a \text{ in } \alpha: \\
R[a] = m \\
\text{for } j = 1..m: \\
R[p[j]] = m-j
\]

$\alpha = \{a, b, c\}$
\begin{align*}
\text{p} &= bbba \\
R[a] &= 0 \\
R[b] &= 1 \\
R[c] &= 4
\end{align*}

NB: $R$ is called $\delta_1$ in the book
Updated algorithm

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

x=abbacbbbababacabbbba
p=bbba
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
**Updated algorithm**

\[ i = m \]

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

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

\[ i=1 \]
\[ x=abbaacbbaababacabbbba \]
\[ p=bbba \]
\[ j=1 \]
**Updated algorithm**

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

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

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

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

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

\[ i = 5 \]

\[ x = abbacbbbababacabbbba \]
\[ p = bbba \]

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

x = abbaabcbbbababacabbbba
p = bbbba
Updated algorithm

\[
i = m
\]

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

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

\[
\begin{align*}
  i &= 5 \\
  x &= abba cbbbababacabbbba \\
  p &= bbbba \\
  j &= 4 \\
  m-j+1 &= 1 \\
  R[x[5]] &= R[c] = 4
\end{align*}
\]
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

\[ i = 9 \]
\[ x = \text{abbacbbbababacabbbba} \]
\[ p = \text{bbba} \]

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

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

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

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

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 $S_1[j] = j'$ 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).
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.
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)

$S[j]$ 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

\[ p[j'+1..j'+(m-j)] \text{ is a border of } p[j'+1..m] \]

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; see exercise 1.3.10: If $B(p)$ is the border-array of $p$, then $\rho = (B(p^R))^R$, i.e. the reverse border-array of the reverse pattern.)
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$''$; see exercise 7.1.9: If $\rho[j'] = \rho[j'+1]+1$ then $\rho''[j'] = \rho''[m-\rho[j'+1]]$, otherwise $\rho''[j'] = \rho[j'+1])$.
Computing $S_1$

Let $LS[1..m-1]$ be defined by

$$LS[j' + \rho''[j']] = \rho''[j'] + 1$$
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 $p[j'] \neq p[j]$

To compute $S_1$ do:

for $k=1..m-1$:
  $j = m - LS[k] + 1$
  $j' = k - LS[k] + 1$
  $S_1[j] = j'$
Computing $S_1$

$$LS[j' + \rho''[j']] = \rho''[j'] + 1$$

To compute $LS$ do:

$$LS[1..m-1] = 1$$
for $j'=m-1..1$:

$$k = j' + \rho''[j']$$
$$LS[k] = \rho''[j'] + 1$$
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):

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

for $m = m$ 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 $S_2$ 

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

\[ j = 1; \quad r = 0 \]

\textbf{while} \ j \ \leq \ m:\n
\textbf{while} \ B^r[m] \ \leq \ m - j:\n
\[ S_2[j] = B^r[m] \]

\[ j \ += \ 1 \]

\[ r \ += \ 1 \]
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 $i \leq |x|:$
    $i,j = hctam(i,m)$
    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

\[ p = \text{cbaaba} \]

\[ R[a] = 0, R[b] = 1, R[c] = 5 \]
Boyer-Moore: Example

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

\[ i=6 \]
\[ x=abbacbaababababacabbbba \]
\[ p=cbaaba \]
Boyer-Moore: Example

\[
i = m
\]

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

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

\[
\begin{array}{c}
  x=abbacbaabababacabbbbba \\
p=cbaaba \\
\end{array}
\]
Boyer-Moore: Example

\[
i = m
\]

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

\( x = abbacbaababababacacabbbba \)

\( p = cbaaba \)
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]])
```

\[ i=7 \]

\[ x= abbacbaabababacabbbba \]

\[ p=cbaaba \]
Boyer-Moore: Example

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

\[ \text{x} = \text{abba}c\text{ba}ab\text{a}bab\text{a}ac\text{a}b\text{b}\text{b}\text{b}\text{b}\text{b} \]
\[ \text{p} = \text{cba}a\text{ba} \]

\[ \begin{array}{c}
\text{i} = 5 \\
\framebox{\text{x}} = \text{abba}c\text{ba}ab\text{a}bab\text{a}ac\text{a}b\text{b}\text{b}\text{b} \text{b} \\
\text{p} = \text{cba}a\text{ba} \\
\text{j} = 4
\end{array} \]
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]])
```

---

**Example: abbaabababa***

```
x = abba cba abababa babaca bbba
```

```
p = cba aba
```
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]])
```

**Example:**

```
x = "abbacbaabababacabbbba"
p = "cbaaba"
```

**Boyer-Moore:**

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

x[i] = c
j = 4
```

```
i = 10
```

```
i = 10
```

```
x[i] = c
j = 4
```

```
i = 10
```

```
x[i] = c
j = 4
```

```
i = 10
```

```
x[i] = c
j = 4
```

```
i = 10
```

```
x[i] = c
j = 4
```

```
i = 10
```

```
x[i] = c
j = 4
```

```
i = 10
```

```
x[i] = c
j = 4
```

```
i = 10
```

```
x[i] = c
j = 4
```

```
i = 10
```

```
x[i] = c
j = 4
```

```
i = 10
```

```
x[i] = c
j = 4
```

```
i = 10
```

```
x[i] = c
j = 4
```

```
i = 10
```

```
x[i] = c
j = 4
```

```
i = 10
```
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]])

\textbf{Boyer-Moore: Example}

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

\textbf{x=abba\textcolor{green}{c}baaba\textcolor{red}{ba}babacabbbba}

\textbf{p=\textcolor{red}{cbaaba}}

\textbf{i=4}

\textbf{j=0}
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

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

\[x=abba\overbrace{c}baabaababa\overbrace{c}abacabbbba\]

\[p=\overbrace{cbaa}ba\]

Match at i=5
Boyer-Moore: Example

\[ i = m \]

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

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

\[ x = abba \text{abcbaabababacabbbba} \]

\[ p = cbaaba \]
Boyer-Moore: Example

\[ i = m \]

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

\[ x = \text{abbacbaaabababacabbbba} \]
\[ p = \text{cbaaba} \]
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

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

x=abbacbaabababacabbbba
p=cbaaba

i=15
j=5
Boyer-Moore: Example

\[ i = m \]

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

\[ x = \text{abbacbaabababacabbbba} \]
\[ p = \text{cbaaaba} \]
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=abbacbaabaabababacabbbba
p=cbaaba
Boyer-Moore: Example

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

\[
x = abbacbaabababacabbbba\\n\]
\[
p = cbaaba
\]

\[
\begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
R: & 0 & 1 & 5 &   &   &   &   \\
S:  & 1 & 2 & 6 & 6 & 6 & 5 & 3 & 1
\end{array}
\]
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 = abbacbaababababacabbbba
p = cbaaba
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 = abbacbaabababacabbbba
p = cbaaba

$S = [12, 6, 6, 6, 5, 3, 1]$
$R = [0, 1, 5]$

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

i = 21, j = 6
Boyer-Moore: Example

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

\[ x = \text{abbacbaabababacabb}bba \]
\[ p = cbaaba \]
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 = `abbacbaabababacabb`

p = `cbaaba`

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

i = 19
j = 4

x[i] = b

j = 4
Boyer-Moore: Example

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

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

\[x=\text{abbacbaaabababacabbbba} \quad \text{p=}\text{cbaaba}\]
Number of comparisons

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

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
Project 3

http://cs.au.dk/~cstorm/courses/StrAlg_f12/project3.html

PROJECT 3 - STRING ALGORITHMS Q4/2012

This project is about exact pattern matching and the KMP and BM algorithms.

You should implement both the KMP and BM algorithms and compare their execution speed. Remember that the KMP algorithm has worst case running time $O(n+m)$ while the BM algorithm has worst case running time $O(nm)$ but is expected to run faster in many cases.

The project should be in groups of 2–3 students. It will not be graded.

Evaluation

The project is evaluated by an oral presentation/discussion in class on Thursday, May 24. All groups should consider the following questions, such that you can participate in the discussion:

- Insights you may have had while implementing and comparing the algorithms,
- Problems encountered, if any,
- How the type of strings affect the running time (repeats, random string, structured strings like English text, etc.).

As a minimum you should test your algorithms English text (where you can use the files you get for Project 1). Here we expect BM to run faster than KMP (but depending on the word we are searching for, of course). Try comparing the running time of the two algorithms with the suffix tree based exact pattern matching you implemented for Project 1.

For strings with a high number of repetitions, we expect the BM algorithm to be slow, and you can try generating such strings to test this. For

Remember to hand in your 2nd project TODAY!