

Exact pattern matching

Knuth-Morris-Pratt and Boyer-Moore

x=*abbac***bb***ababacab***bbba**



The diagram shows the string **x** = *abbac***bb***ababacab***bbba**. The substrings **bb** and **bbba** are highlighted in red. Above the first **bb** is the number 6 with a red arrow pointing to the first 'b'. Above the **bbba** is the number 17 with a red arrow pointing to the first 'b' of that substring.

Exact pattern matching

Given string $\mathbf{x} = \text{abbacbbbababacabbba}$ and pattern $\mathbf{p} = \text{bbba}$
find all occurrences of \mathbf{p} in \mathbf{x}

$\mathbf{x} = \text{abbac}$ ⁶
 ↓
bbba bababacab ¹⁷
 ↓
bbba

Simple algorithm

Scan along x

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

Try to match p

Example

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

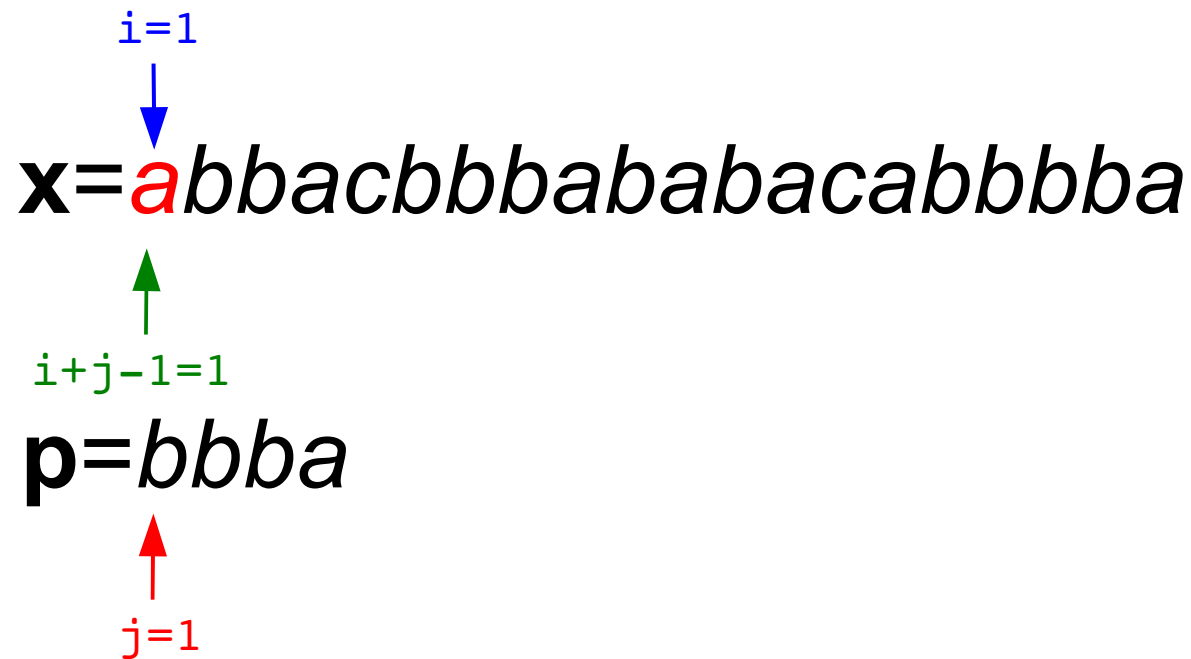
$i=1$
↓
x=abbacbbbabacabbba

p=bbba

↑
 $j=1$

Example

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



Example

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

i=2
↓
x=abbacbbbabacabbba

p=bbba

Example

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

x=*abbacbbbabacabbbba*



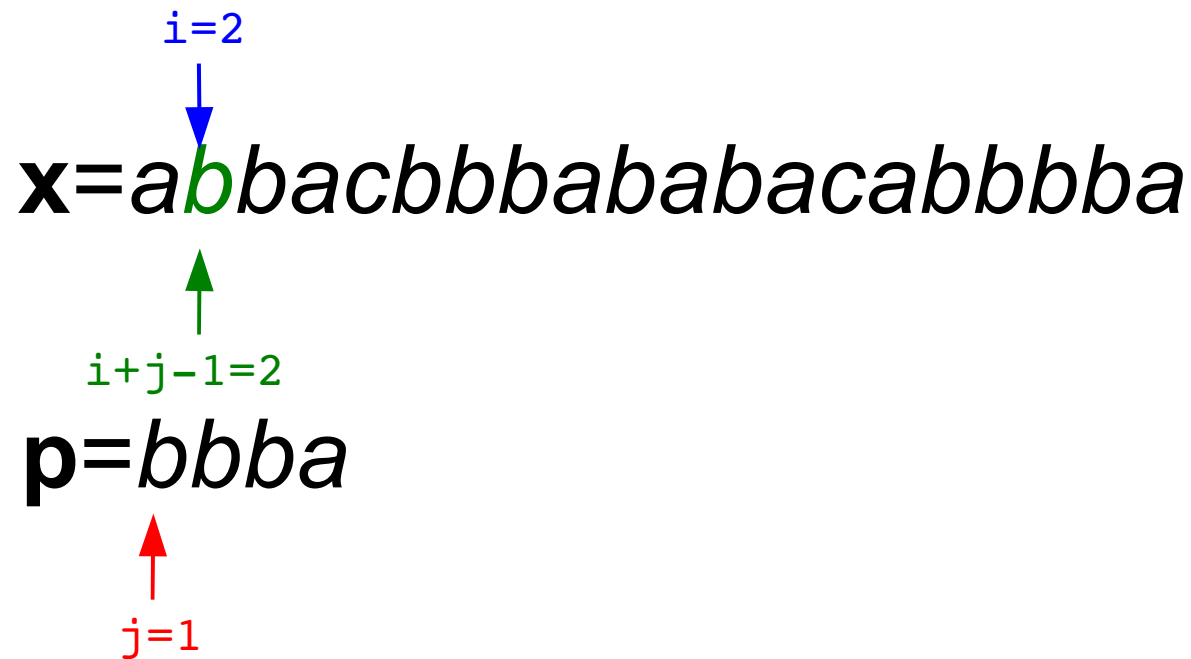
p=*bbba*

j=1



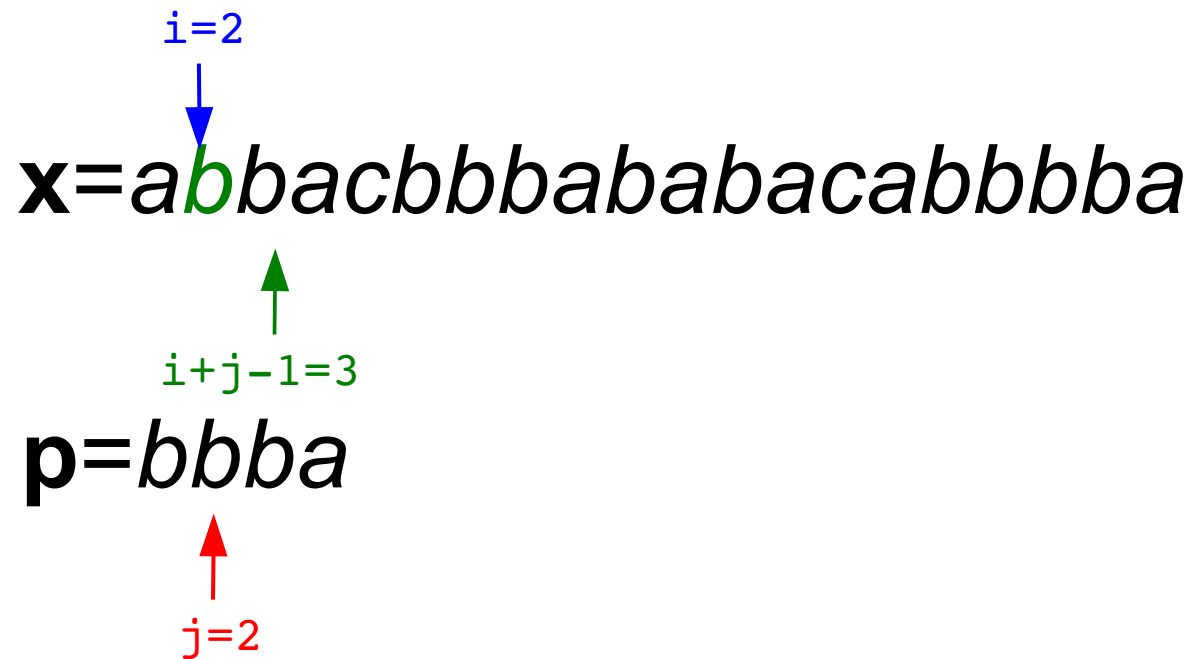
Example

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



Example

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



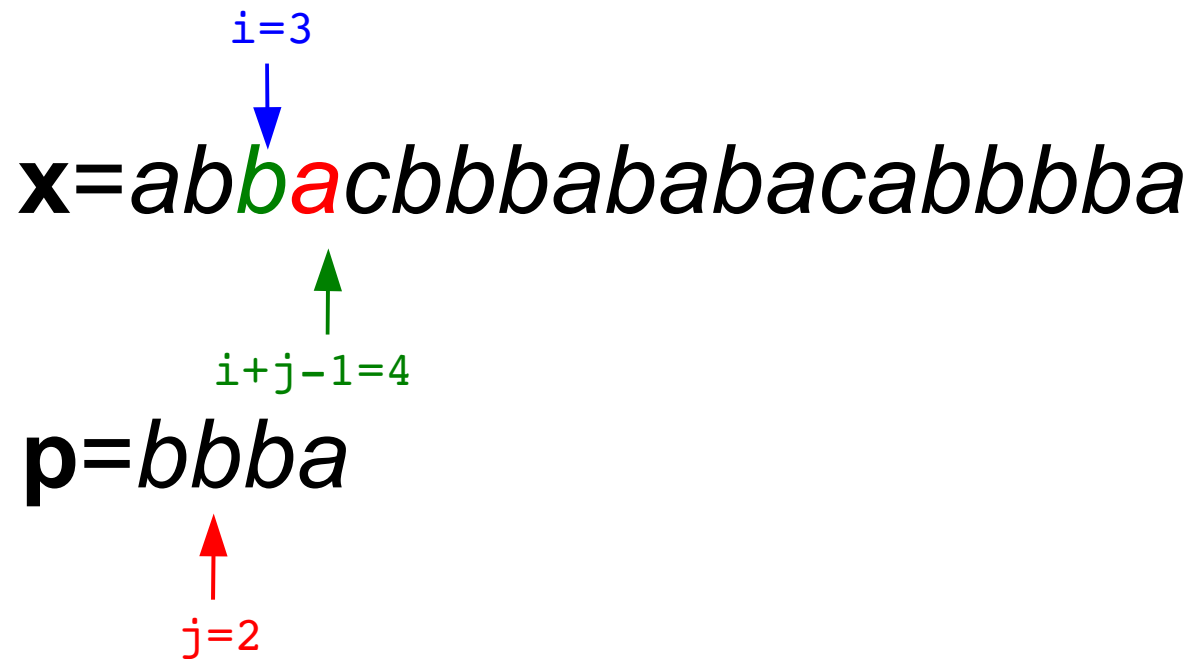
Example

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

$i=2$
↓
x=*ab***a**cbbbababacabbba
↑
 $i+j-1=4$
p=bbba
↑
 $j=3$

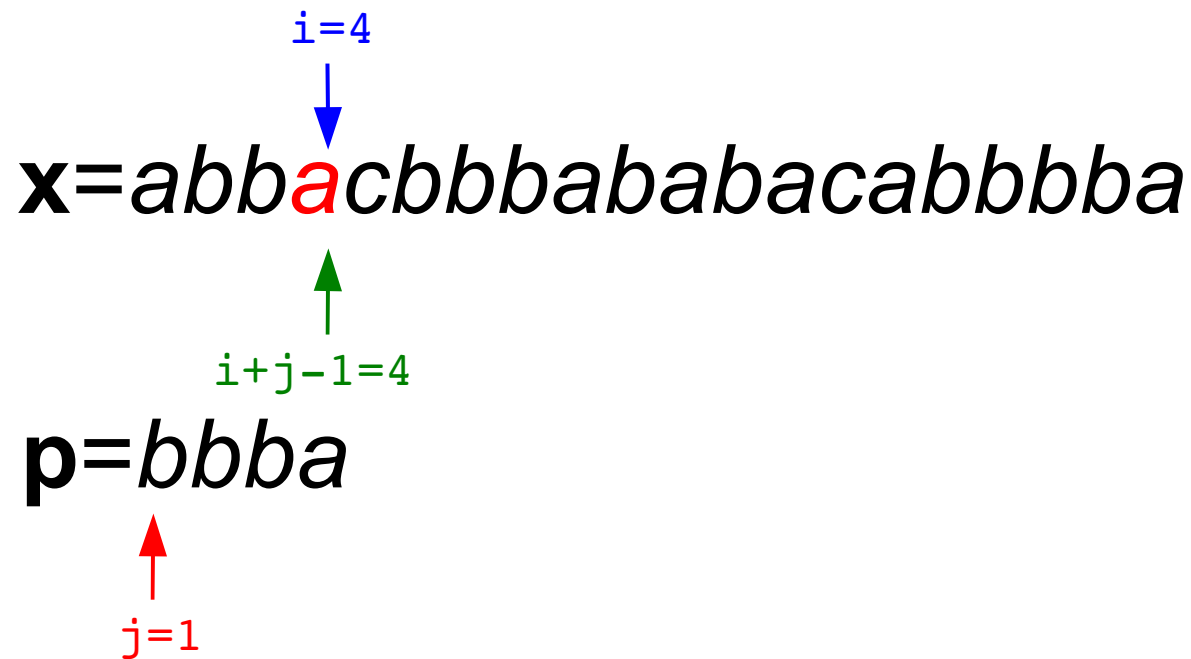
Example

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



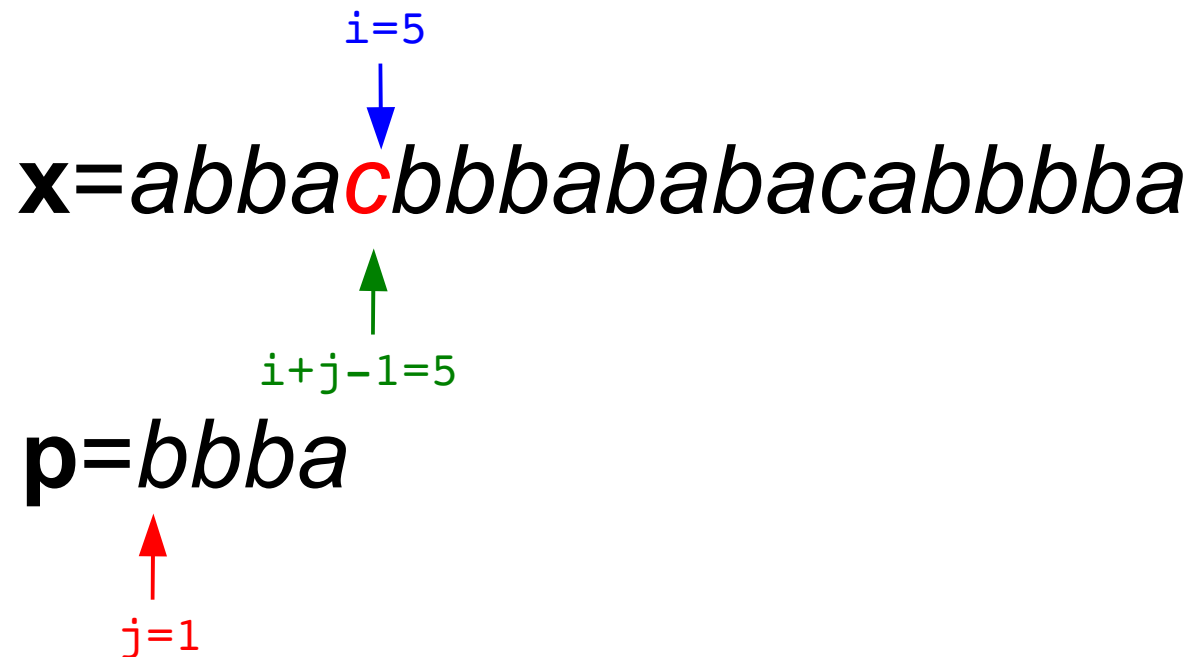
Example

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



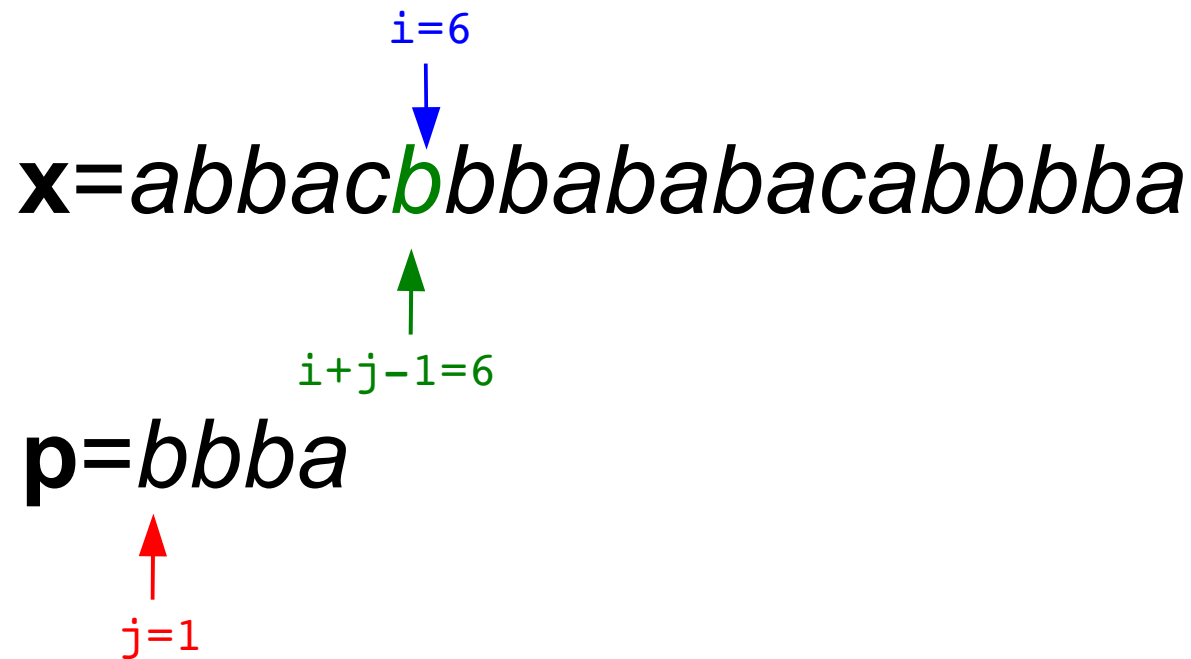
Example

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



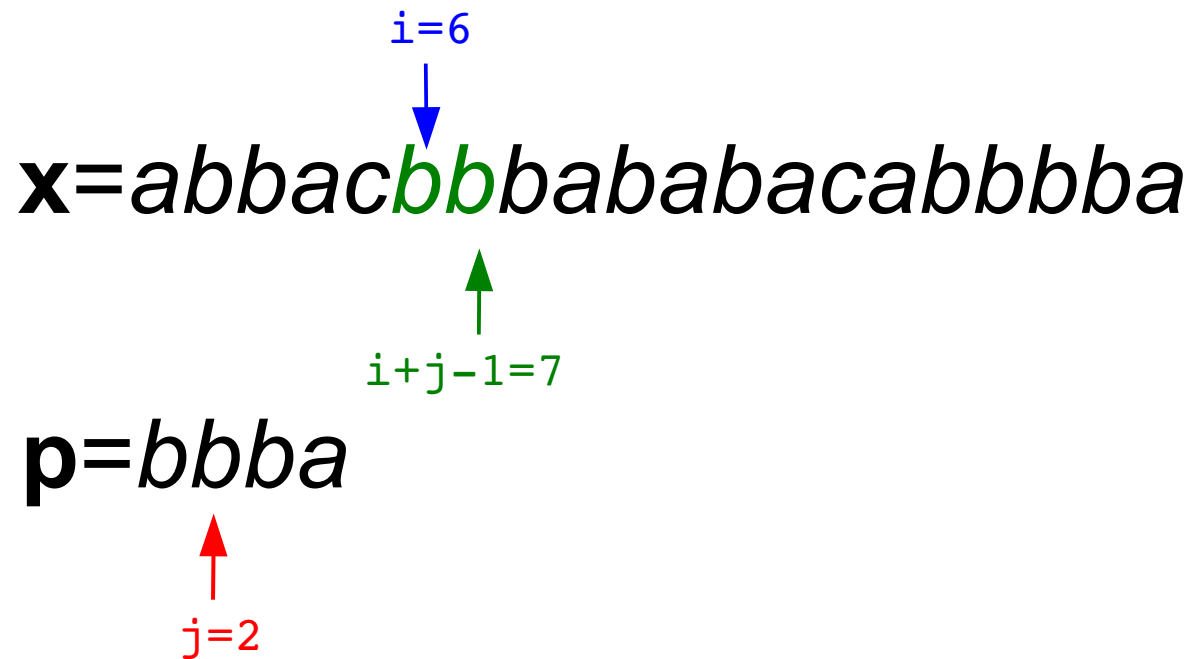
Example

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



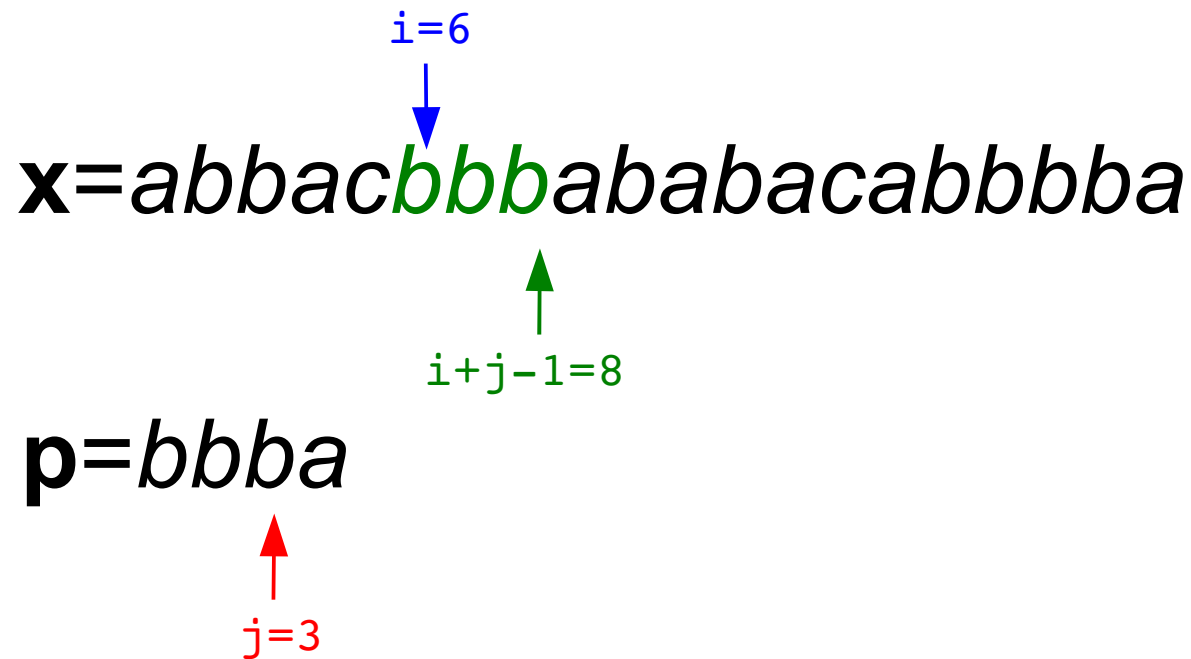
Example

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



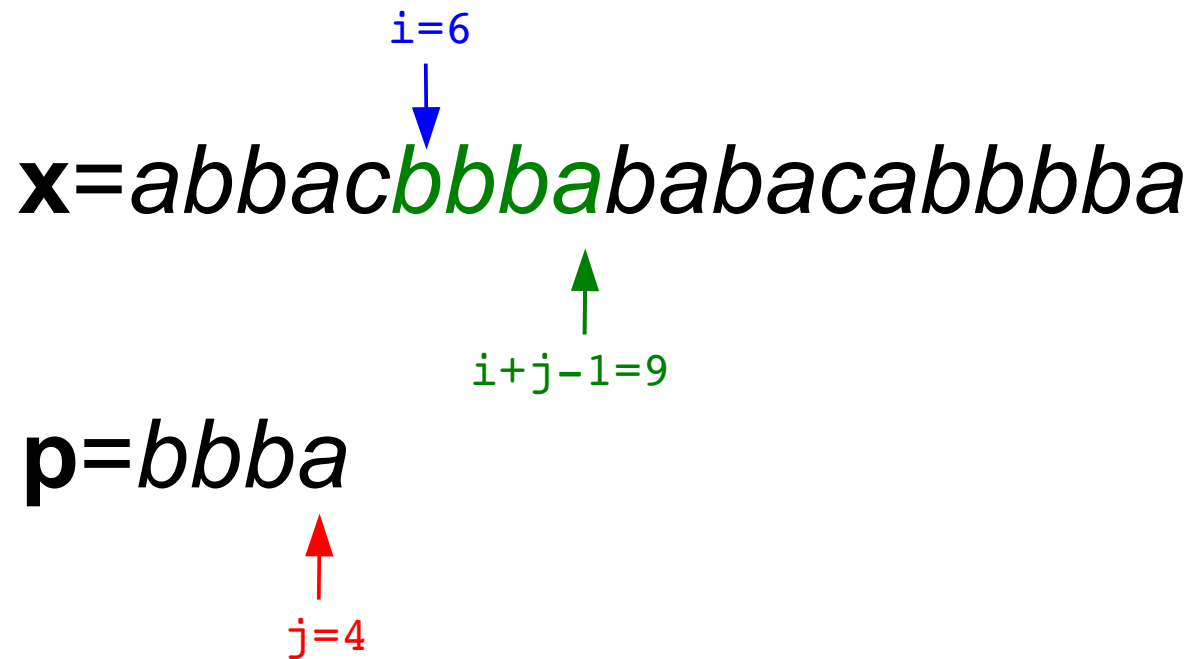
Example

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



Example

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



Example

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

$i=6$
↓
x=abbac**bbba**abacabbba

p=bbba

↑
 $i+j-1=10$

↑
 $j=5$

↖
Match at i=6

Running time

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

If **p** matches here



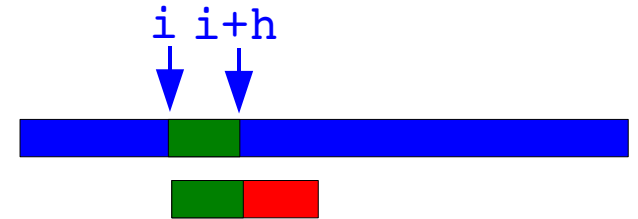
x=*abbac**bbb**ababacabbba*



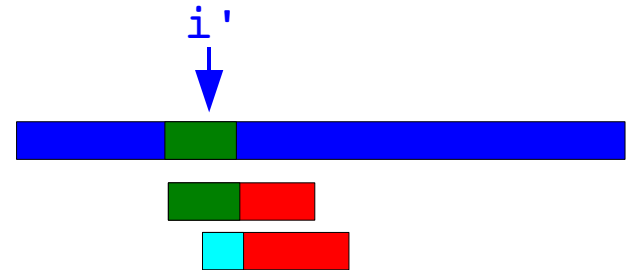
It cannot match here

Observation

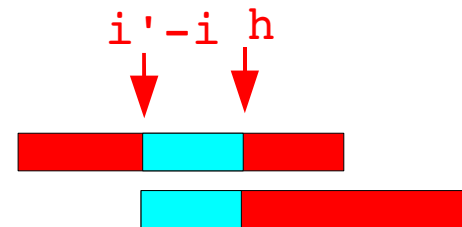
If a prefix of length h of \mathbf{p} matches at index i



then \mathbf{p} cannot match at i' in $[i, i+h]$



unless $\mathbf{p}[1..h-(i'-i)]$ equals $\mathbf{p}[i'-i..h]$
i.e. $\mathbf{p}[1..h-(i'-i)]$ is a *border* of $\mathbf{p}[1..h]$



Trick: Using borders

If mismatch at $x[i+h+1]:p[h+1]$:

$x = \text{gcggc} \overbrace{\text{acttaact}}^h \text{gattagacagtaagac...}$
 $p = \text{acttaactcgc}$

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

$x = \text{gcggcacttaactgattagacagtaagac...}$
 $p = \text{acttaactcgc}$
 $p = \text{acttaactcgc}$

Trick: Using borders

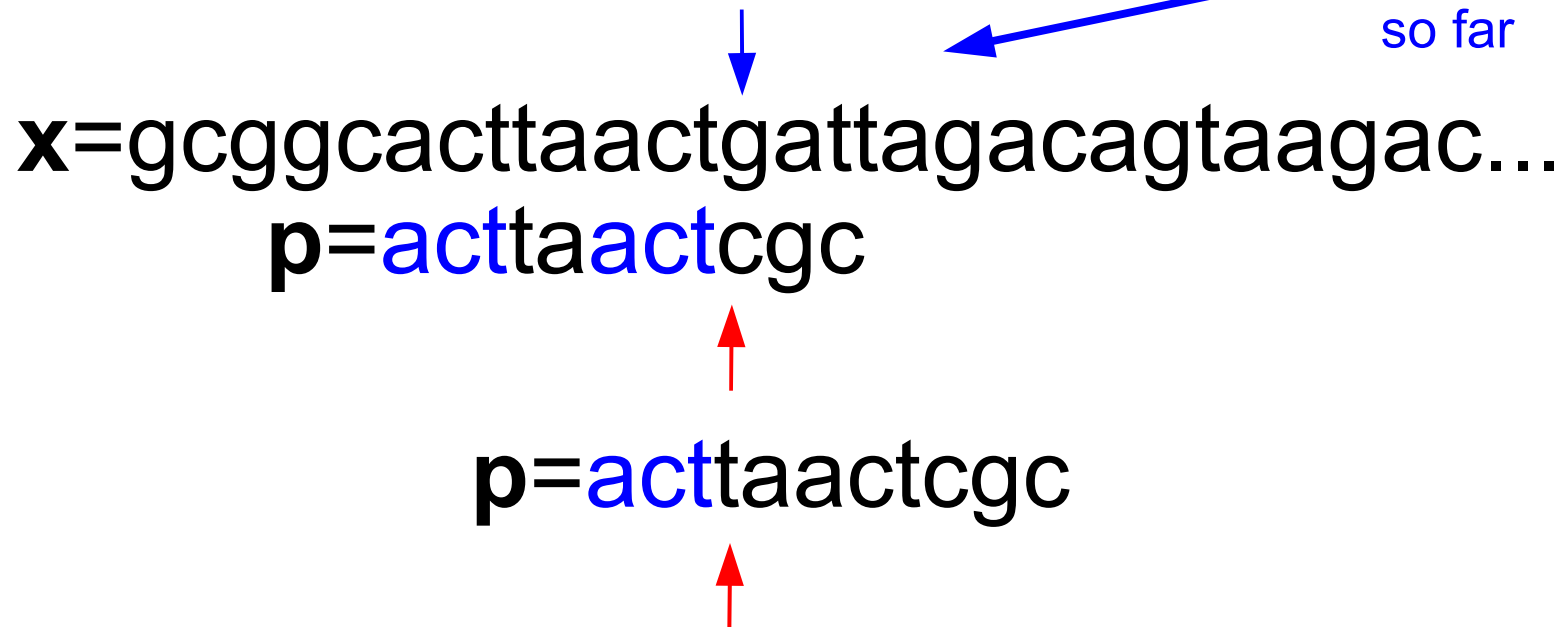
If we don't move the index into \mathbf{x} , the conceptual shift of \mathbf{p} is just moving the index in \mathbf{p} to $\beta[h]+1$



Trick: Using borders

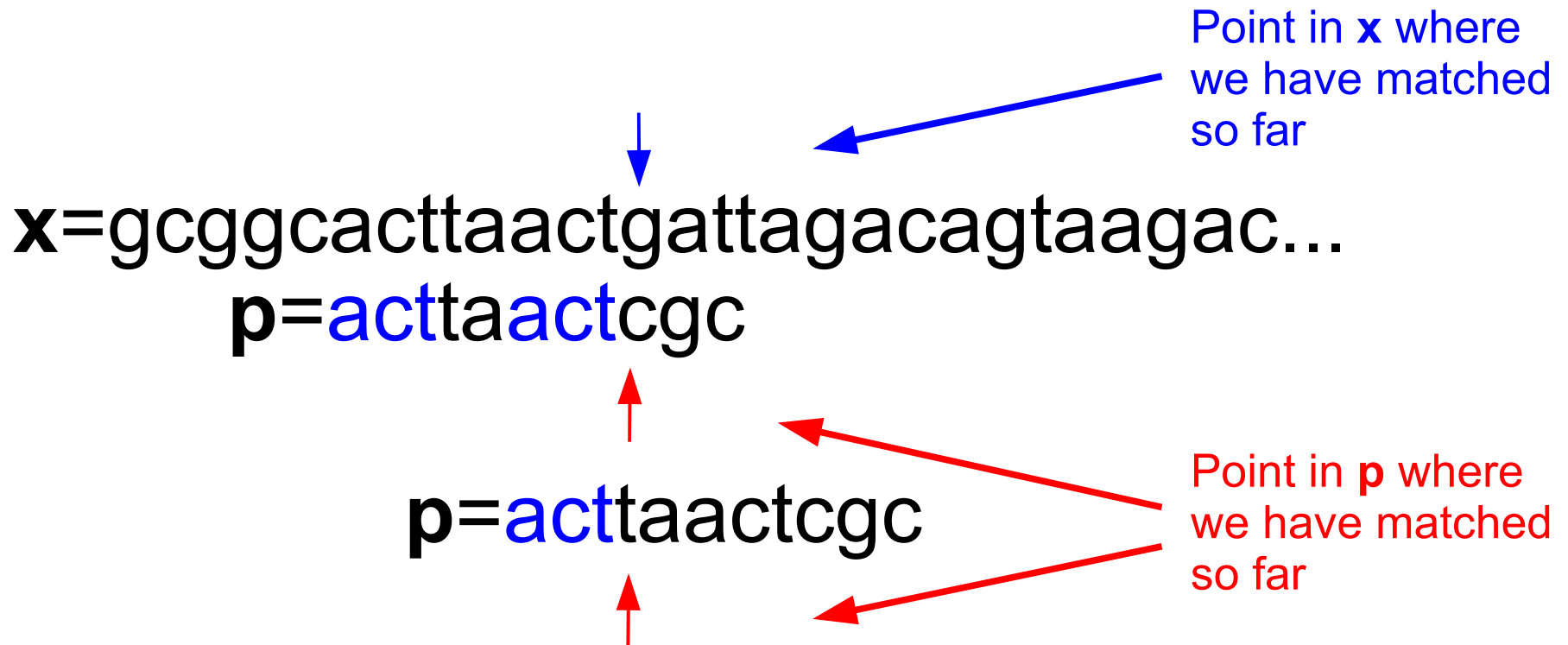
If we don't move the index into \mathbf{x} , the conceptual shift of \mathbf{p} is just moving the index in \mathbf{p} to $\beta[h]+1$

Point in \mathbf{x} where
we have matched
so far



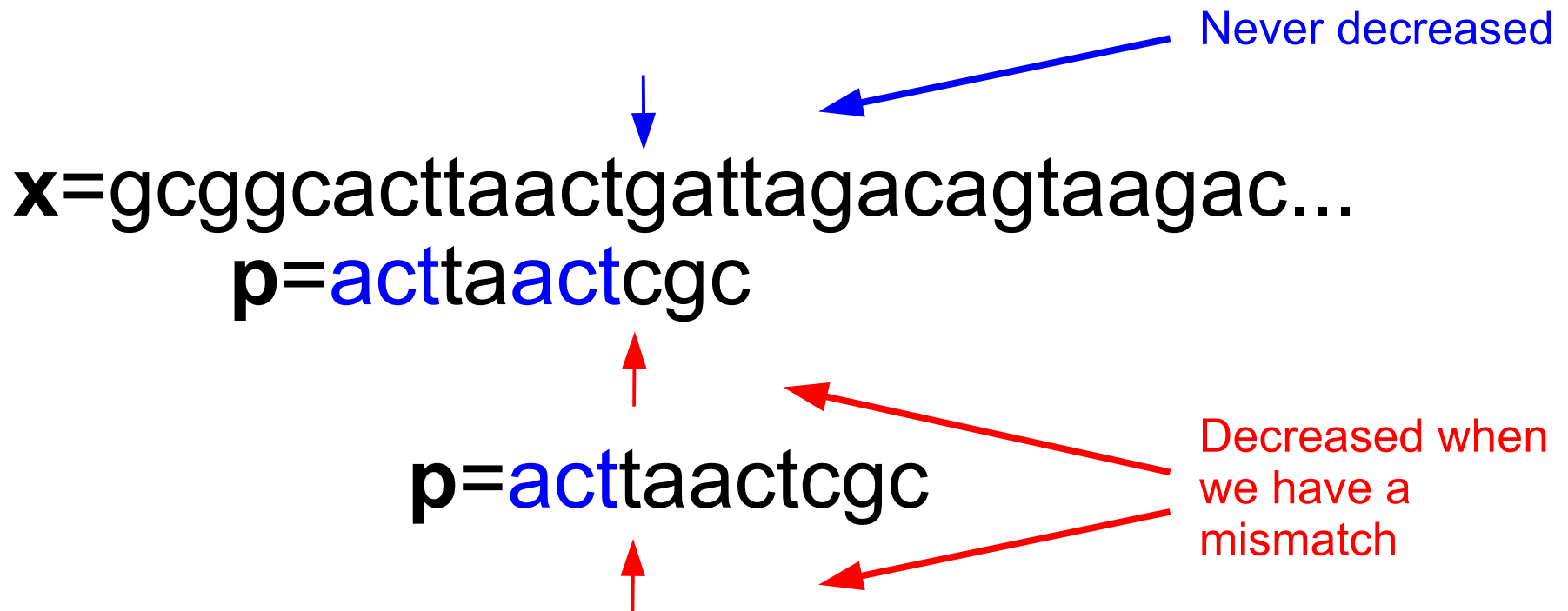
Trick: Using borders

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



Trick: Using borders

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



Knuth-Morris-Pratt

Preprocessing:

build border array B

$B'[1] = 0$

for $j=2..|p|+1$:

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

Main:

$i = 1; j = 1$

while $i \leq |x| - |p| - j$:

$i, j = \text{match}(i, j, |p|)$

if $j == |p| + 1$: report match at $i - |p|$

if $j == 1$: $i = i + 1$

else: $j = B'[j]$

Help routine:

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

while $x[i] == p[j]$

and $j \leq m$:

$i = i + 1$

$j = j + 1$

return i, j

Example – preprocessing

Preprocessing:

build border array B

$B'[1] = 0$

for $j=2..|p|+1$:

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

$p = bbba$

$B = 0120$ (β)

$B' = 01231$

$B' = 01231$

Example

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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=abbacbbbabacabbba
p=bbba
↑
 $j=1$

Example

$B' = 01231$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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=*abbacbbbababacabbba*
p=*bbba*
↑
 $j=1$

Example

$B' = 01231$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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=*abbacbbbabacabbba*
p=*bbba*
↑
 $j=1$

Example

$B' = 01231$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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
```

$i=2$
↓
x=abbacbbbabacabbba
p=bbba
↑
 $j=1$

Example

$B' = 01231$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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=*abb***a***cbbbababacabbbba*
p=*bb***b***a*
↑
 $j=3$

Example

$B' = 01231$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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=*abb***a***cbbbababacabbbba*
p=*bb***b***a*
↑
 $j=3$

Example

$B' = 01\underline{2}31$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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=*ab***b**acbbababacabbba
p=*b***b**ba
↑
 $j=2$

Example

$B' = 01231$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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=abb**a**cbbbababacabbba
p=bb**a**
↑
 $j=2$

Example

$B' = 0\underline{1}231$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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=abbacbbbabacabbba
p=bbba
↑
 $j=1$

Example

$B' = 01231$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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=abb**a**cbbbababacabbba
p=**b**bba
↑
 $j=1$

Example

$B' = 01231$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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
```

$i=5$
↓
x=abbacbbbababacabbba
p=bbba
↑
 $j=1$

Example

$B' = 01231$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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
```

$i=5$
↓
x=abbacbbbababacabbba
p=bbba
↑
 $j=1$

Example

$B' = 01231$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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
```

$i=6$
↓
x=abbacbbbababacabbba
p=bbba
↑
 $j=1$

Example

$B' = 01231$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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
```

$i=10$
↓
x=abbacbbba**babacabbba**
p=bbba****
↑
 $j=5$

Example

$B' = 01231$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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
```

$i=10$
↓
x=abbac**bbba**babacabbba
p=**bbba**
↑
 $j=5$

Match at $i=6$

Example

$B' = 0123\underline{1}$

```
i = 1; j = 1
while i <= |x| - |p| - j:
    i, j = match(i, j, |p|)
    if j == |p| + 1:
        report match at i - |p|
    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
```

$i=10$
↓
x=abbacbbbabacabbba
p=bbba
↑
 $j=1$

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

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

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

x=abbacbbbababacabbba

p=bbba

Variation on the simple algorithm

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

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

i=4
↓
x=abbacbbbababacabbba
p=bbba

Variation on the simple algorithm

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

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

$i=1$
↓
x=*ab***b***ac***b***bb***a***ba***a***ba***c**a***bb***b***ba***

p=*b***b***b***a**

↑
 $j=1$

Variation on the simple algorithm

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

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

i=5
↓
x=abbacbbbababacabbba
p=bbba

Variation on the simple algorithm

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

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

x=abbac**bbb**ababacabb**bba**

p=bbb**a**

i=5
↓
j=4
↑

Variation on the simple algorithm

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

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

i=6
↓
x=abbacbbbababacabbba
p=bbba

Variation on the simple algorithm

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

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

x=abbac**b**bbababacabbba
p=bb**b**a

i=6
↓
j=4
↑

Variation on the simple algorithm

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

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

i=7
↓
x=abbacbbbababacabbba
p=bbba

Variation on the simple algorithm

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

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

$i=7$
↓
x=abbacb*bbababacabbba*

p=bbb*a*
↑
 $j=4$

Variation on the simple algorithm

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

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

$i=8$
↓
x=abbacbbbababacabbba
p=bbba

Variation on the simple algorithm

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

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

$i=8$
↓
x=*abbacbbbababacabbba*
p=*bbb***a**
↑
 $j=4$

Variation on the simple algorithm

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

Variation on the simple algorithm

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

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

x=abbac**bbba**babacabbba

p=**bbba**

i=5

j=0

Variation on the simple algorithm

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

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

$i=5$
↓
x=abbacbb**ababacabbba**

p=bb**ba**

↑
 $j=0$

Match at $i=6$

Variation on the simple algorithm

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

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

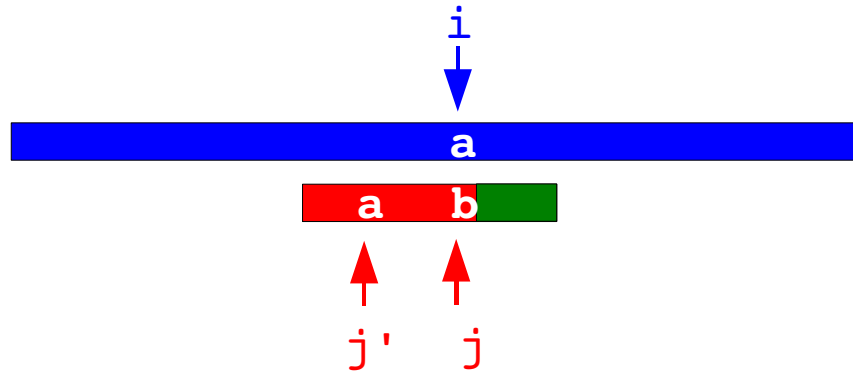
i=10
↓
x=abbacbbbababacabbba
p=bbba

Running time

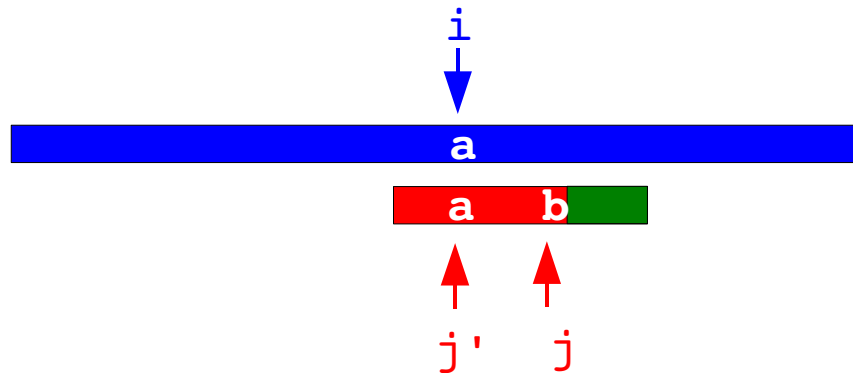
- Works as good (or bad) as the simple algorithm
 - Time and space in $O(|\mathbf{x}||\mathbf{p}|)=O(n^2)$

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 \mathbf{p} to the rightmost occurrence of a in \mathbf{p} , or $|\mathbf{p}|$ if a is not in \mathbf{p}

```
for a in  $\alpha$ :  
     $\mathbf{R}[a] = |\mathbf{p}|$   
for j = 1.. $|\mathbf{p}|$ :  
     $\mathbf{R}[\mathbf{p}[j]] = |\mathbf{p}| - j$ 
```

$\mathbf{p} = bbba$
 $\alpha = \{a, b, c\}$

$\mathbf{R}[a] = 0$ $\mathbf{p} = bbba$
 $\mathbf{R}[b] = 1$ $\mathbf{p} = bbba$
 $\mathbf{R}[c] = 4$ $\mathbf{p} = bbba$

“Rightmost” array

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

```
for a in  $\alpha$ :  
     $\mathbf{R}[a] = |\mathbf{p}|$   
for j = 1.. $|\mathbf{p}|$ :  
     $\mathbf{R}[\mathbf{p}[j]] = |\mathbf{p}| - j$ 
```

$\mathbf{p} = bbba$
 $\alpha = \{a, b, c\}$

$\mathbf{R}[a] = 0$ $\mathbf{p} = bbba$
 $\mathbf{R}[b] = 1$ $\mathbf{p} = bbba$
 $\mathbf{R}[c] = 4$ $\mathbf{p} = bbba$

NB: \mathbf{R} is called δ_1 in the book

Updated algorithm

```
i = |p|
while i <= |x|:
    i, j = hctam(i, |p|)
    if j==0: report i+1 as match
    i = i + max{|p|-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=abbacbbbababacabbba

p=bbba

Updated algorithm

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

$i=4$
↓
x=abbacbbbababacabbba
p=bbba

Updated algorithm

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

$i=1$
↓
x=*ab***b***ac***b***bb***a***ba***a***ba***c**a***bb***b***ba***

p=*b***b***b***a**

↑
 $j=1$

Updated algorithm

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

$i=1$
↓
x=*ab***b***ac***b***bb***a***ba***a***ba***c**a***bb***b***ba***

p=*b***b***b***a**

↑
 $j=1$

$|p| - j + 1 = 4$
and $R[x[1]] = R[a] = 0$

Updated algorithm

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

$i=5$
↓
x=abbacbbbababacabbba
p=bbba

$|p| - j + 1 = 4$
and $R[x[1]] = R[a] = 0$

Updated algorithm

```
i = |p|
while i <= |x|:
    i, j = hctam(i, |p|)
    if j==0: report i+1 as match
    i = i + max{|p|-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=abbac**bbb**ababacabb**bb**a

p=bbba

i=5
↓

↑
j=4

Updated algorithm

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

$i=5$
↓
x=abbacbbbababacabbba
p=bbba
↑
 $j=4$

$|p| - j + 1 = 1$
 $R[x[5]] = R[c] = 4$

Updated algorithm

```
i = |p|
while i <= |x|:
    i, j = hctam(i, |p|)
    if j==0: report i+1 as match
    i = i + max{|p| - 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=abbacbbbababacabbba
p=bbba

$|p| - j + 1 = 1$
 $R[x[5]] = R[c] = 4$

Updated algorithm

```
i = |p|
while i <= |x|:
    i, j = hctam(i, |p|)
    if j==0: report i+1 as match
    i = i + max{|p| - 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=abbacbbbababacabbba
p=bbba

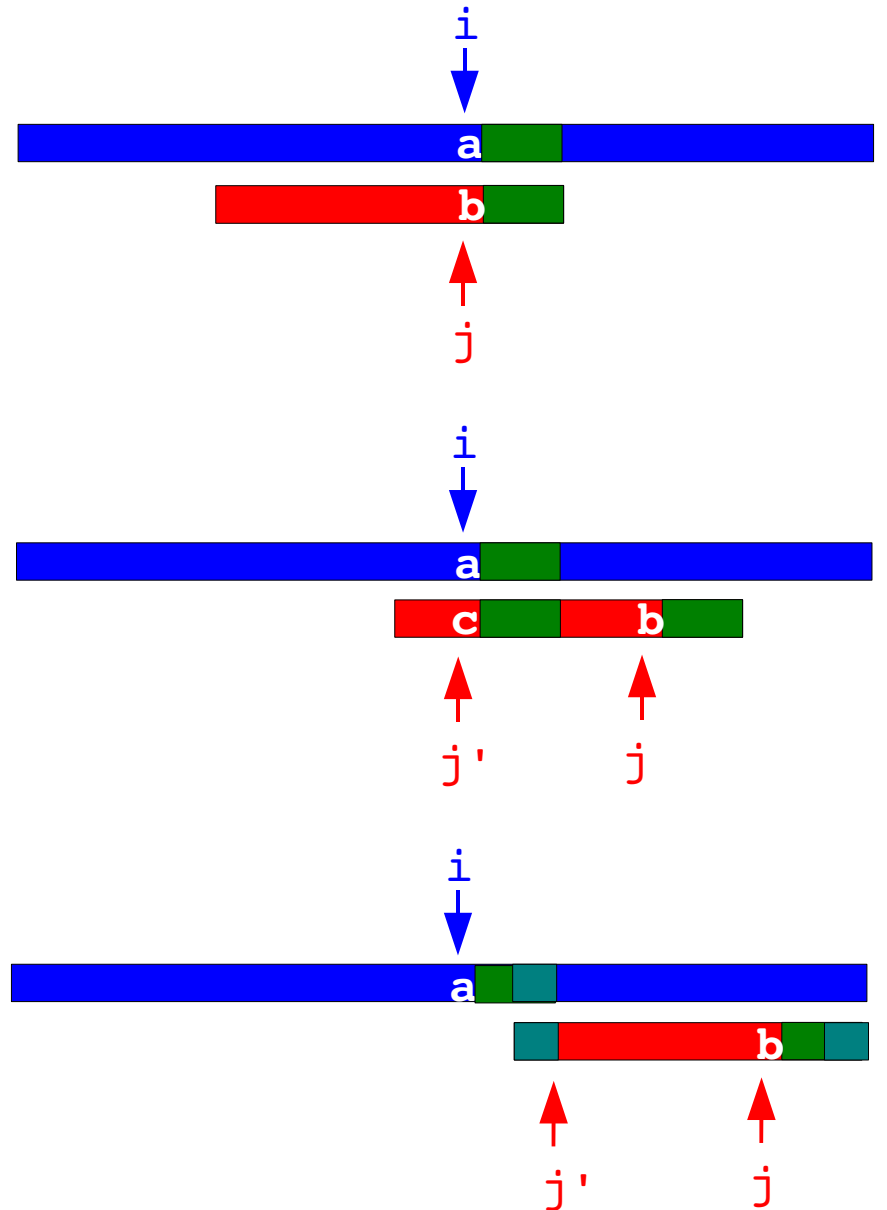
A shortcut of 4 character here!

Observation

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

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

or rightmost $j' < |p|$ where $p[1..j']$ is a suffix of $p[j..|p|]$:

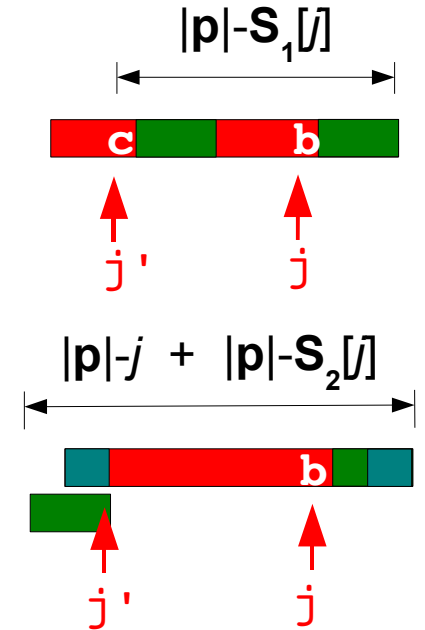


Trick 2: match suffixes...

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 $\mathbf{S}[j] = \min(|\mathbf{p}| - \mathbf{S}_1[j], |\mathbf{p}| - j + |\mathbf{p}| - \mathbf{S}_2[j])$ for $j = 1..|\mathbf{p}|$, and
 $\mathbf{S}[0] = 2|\mathbf{p}|$ - “longest border of \mathbf{p} ” (special case of case 2)

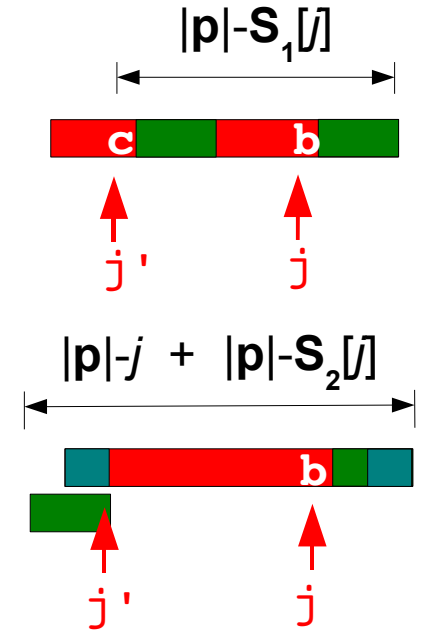


Trick 2: match suffixes...

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 $\mathbf{S}[j] = \min(|\mathbf{p}| - \mathbf{S}_1[j], |\mathbf{p}| - j + |\mathbf{p}| - \mathbf{S}_2[j])$ for $j = 1..|\mathbf{p}|$, and
 $\mathbf{S}[0] = 2|\mathbf{p}|$ - “longest border of \mathbf{p} ” (special case of case 2)



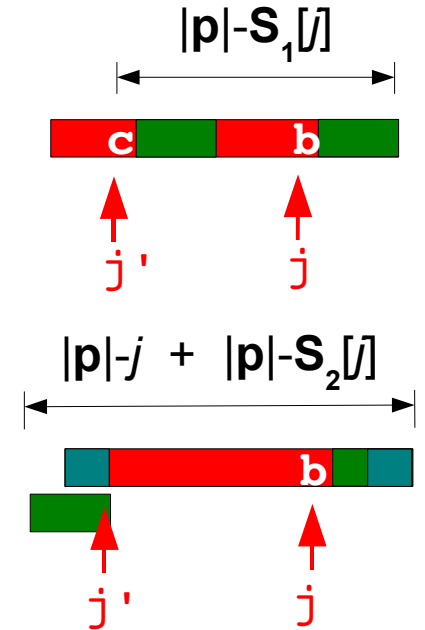
**NB: S (for “suffix”)
is called δ_2 in the book**

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

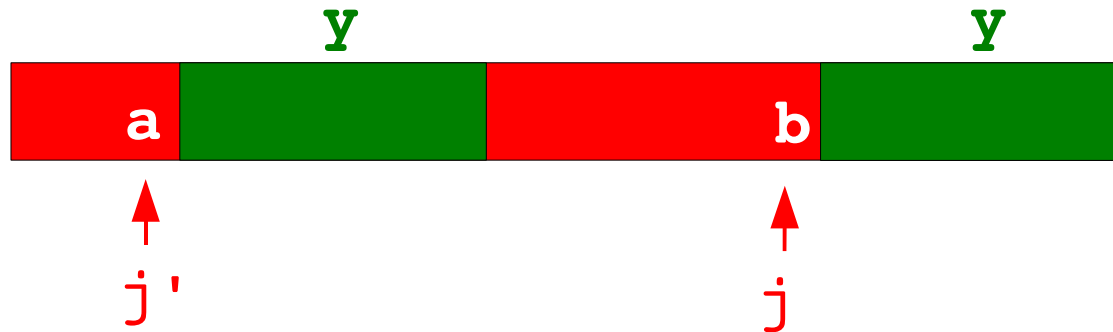
Define $\mathbf{S}[j] = \min(|\mathbf{p}| - \mathbf{S}_1[j], |\mathbf{p}| - j + |\mathbf{p}| - \mathbf{S}_2[j])$ for $j = 1..|\mathbf{p}|$, and
 $\mathbf{S}[0] = 2|\mathbf{p}|$ - “longest border of \mathbf{p} ” (special case of case 2)



$\mathbf{S}[j]$ is the amount we should increase i to get a point where the string matches the suffix seen so far

Computing S_1

For each j , j' is the rightmost index where y is a border of $p[j'+1..m]$ ($m=|p|$) and $p[j']y$ is not a border of $p[j'..m]$.

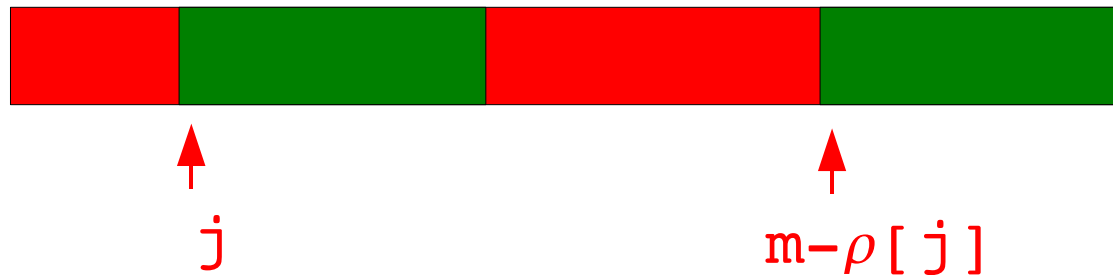


Computing S_1

Let ρ be the “reverse” border array, i.e.:

$\rho[j]$ is the length of the longest border of $\mathbf{p}[j..m]$

(ρ can be calculated similarly to the border array).



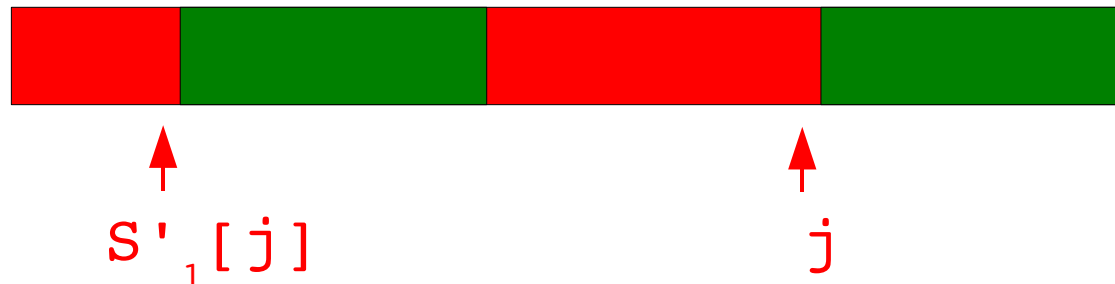
Computing S_1

If we calculate S_1 by:

for $i=1..m$:

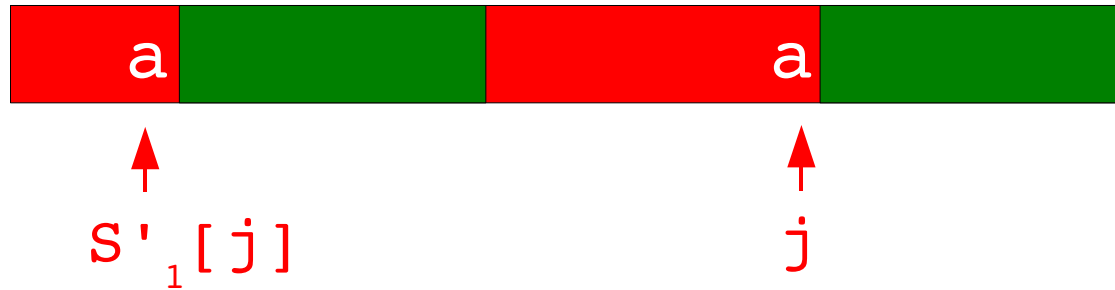
$$S_1[m-\rho[i]-1] = i-1$$

then $S_1[j]$ is the rightmost index j' such that $p[j'+1..j'+m-\rho[j]+1]$ is a border of $p[j'+1..m]$.



Computing S_1

Array S'_1 is almost, but not quite, what we need...



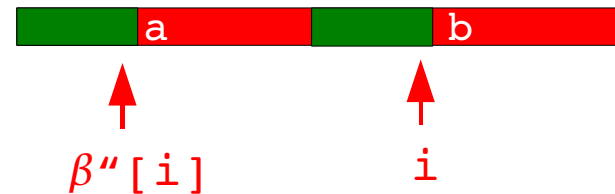
To get S_1 we must make sure that $p[j'] \neq p[j]$

Computing S_1

We can build an array β'' similarly to β , but where:

$\beta''[i]$ is the length of the longest border of $p[1..i]$ such that $p[\beta''[i]+1] \neq p[i]$

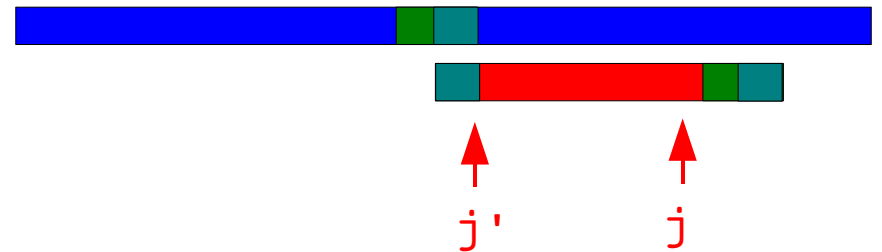
(See exercise 7.1.9)



Using the corresponding ρ'' instead of ρ
we can build the correct S_1 array

Computing S_2

For each j , the corresponding j' is the length of the longest border of \mathbf{p} shorter than $|\mathbf{p}|-j$.



Given the border array β , the borders of \mathbf{p} are (in decreasing length):

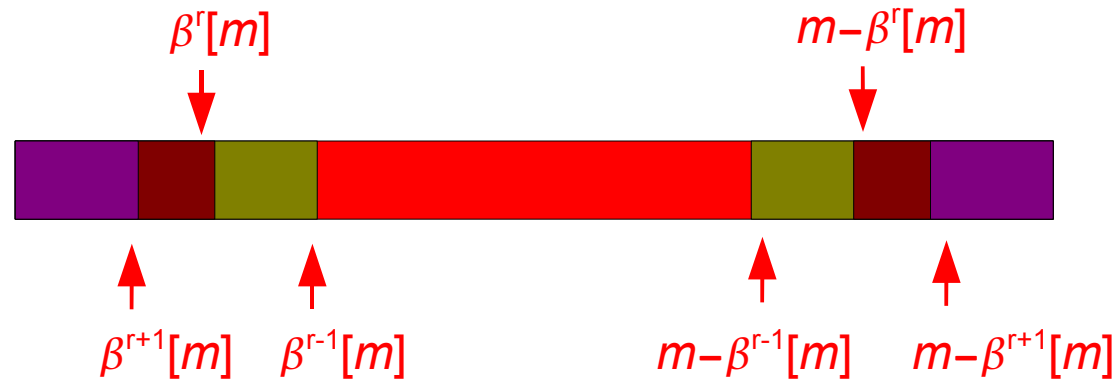
$$\beta[m], \beta^1[m], \beta^2[m], \dots, \beta^k[m]=0$$

for $m=|\mathbf{p}|$ and some k .

Given the border array, this sequence of borders of \mathbf{p} can be calculated in linear time.

Computing S_2

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



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

Below $m - \beta^{r-1}[m]$, $\beta^{r-1}[m]$ is a longer border less than $m - j$.
Above $m - \beta^r[m]$, $\beta^r[m]$ is longer than $m - j$.

Computing S_2

```
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:

$i = |p|$

while $i \leq |x|$:

$i, j = \text{hctam}(i, |p|)$

if $j == 0$: report match at $i+1$

$i = i + \max(\mathbf{S}[j], \max\{|p| - j + 1, \mathbf{R}[x[i]]\})$

The Boyer-Moore Algorithm

Preprocessing:

Calculate **R** and **S**

Main:

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

Safe since $\mathbf{S}[j] \geq |p|-j+1$
(because $\mathbf{S}_2[j] < |p|$)

Boyer-Moore: Example



$\mathbf{R}[a] = 0, \mathbf{R}[b] = 1, \mathbf{R}[c] = 5$

$\mathbf{S}_1[1] = 0, \mathbf{S}_1[2] = 0, \mathbf{S}_1[3] = 0, \mathbf{S}_1[4] = 1, \mathbf{S}_1[5] = 3, \mathbf{S}_1[6] = 5$

$\mathbf{S}_2[1] = 0, \mathbf{S}_2[2] = 0, \mathbf{S}_2[3] = 0, \mathbf{S}_2[4] = 0, \mathbf{S}_2[5] = 0, \mathbf{S}_2[6] = 0$

$\mathbf{S}[0] = 12, \mathbf{S}[1] = 6, \mathbf{S}[2] = 6, \mathbf{S}[3] = 6, \mathbf{S}[4] = 5, \mathbf{S}[5] = 3, \mathbf{S}[6] = 1$

$\mathbf{p} = cbaaba$

Boyer-Moore: Example

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

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

i=6
↓

x=*abbacbaabababacabbbba*
p=*cbaaba*

Boyer-Moore: Example

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

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

x = *abbac***b***aabababacabbba*

p = *cbaab***a**

i = 6

j = 6

Boyer-Moore: Example

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

	a	b	c		0	1	2	3	4	5	6	
R:	0	1	5		S:	12	6	6	6	5	3	1
		↑								↑		
		x[i]=b								j=6		

x = *abbac***b***aabababacabbba*

p = *cbaab***a**

i = 6 ↓

↑ *j* = 6

Boyer-Moore: Example

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

	a	b	c		0	1	2	3	4	5	6	
R:	0	1	5		S:	12	6	6	6	5	3	1
		↑								↑		
		x[i]=b								j=6		

i=7
↓

x=*abbacbaabababacabbba*

p=*cbaaba*

Boyer-Moore: Example

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

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

x = abba**c**baabababacabbba
p = cba**a**ba

i = 5
↓
j = 4
↑

Boyer-Moore: Example

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

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

i=4
↓
x=abbacbaababacabbbba
p=cbaaba
↑
j=0

Boyer-Moore: Example

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

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

i=4
↓
x=*abbacbaababacabbba*

p=*cbaaba*

↑
j=0

Match at *i=5*

Boyer-Moore: Example

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

	a	b	c		0	1	2	3	4	5	6	
R:	0	1	5		S:	12	6	6	6	5	3	1
		↑				↑						
		x[i]=a				j=0						

i=4
↓
x=abbacbaababacabbba
p=cbaaba
↑
j=0

Boyer-Moore: Example

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

	a	b	c		0	1	2	3	4	5	6
R:	0	1	5	S:	12	6	6	6	5	3	1
		↑			↑						
		x[i]=a			j=0						

i=16
↓
x=abbacbaabababacabbba
p=cbaaba

Boyer-Moore: Example

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

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

$i=15$
↓
x=abbacbaababababacabbbba
p=cbaaba
↑
 $j=5$

Boyer-Moore: Example

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

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

x = *abbacbaabababacabbba*
p = *cbaaba*

i = 20
↓
j = 6
↑

Boyer-Moore: Example

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

	a	b	c		0	1	2	3	4	5	6	
R:	0	1	5		S:	12	6	6	6	5	3	1
		↑								↑		
		x[i]=b								j=6		

x = *abbacbaabababacabbba*
p = *cbaaba*

i = 20 ↓
j = 6 ↑

Boyer-Moore: Example

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

	a	b	c		0	1	2	3	4	5	6	
R:	0	1	5		S:	12	6	6	6	5	3	1
		↑								↑		
		x[i]=b								j=6		

x = *abbacbaabababacabbba*
p = *cbaaba*

i = 21 ↓

Boyer-Moore: Example

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

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

x = *abbacbaabababacabb***b***ba*

p = *cba***a***ba*

i = 19

j = 4

Boyer-Moore: Example

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

	a	b	c		0	1	2	3	4	5	6	
R:	0	1	5		S:	12	6	6	6	5	3	1
		↑							↑			
		x[i]=b							j=4			

x=*abbacbaabababacabbba*

p=*cbaaba*

i=24
↓

Number of comparisons

$x = abbacbaabababacabbbba$

000023211100001100121=16

$|x| = 21$

We found all occurrences of p in x
in *sub linear* time!

Runtime for Boyer-Moore

- The worst-case time complexity is still as the simple algorithm: $O(|\mathbf{x}||\mathbf{p}|)$
 - Consider searching for $\mathbf{p}=a^m$ in the string $\mathbf{x}=a^n$
 - 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