String Algorithms and Data Structures
Boyer-Moore

CS 199-225
Brad Solomon

September 26, 2022
Exact Pattern Matching \textit{w/} Z-algorithm

\begin{align*}
\text{Pattern, } P & \quad \text{Text, } T \\
\text{Naive} & \approx \theta(|P| \cdot |T|) \\
\text{Z-Algorithm} & \approx \theta(|P| + |T|) \\
\text{Find instances of } P \text{ in } T \\
\text{‘instances’: An exact, full length copy}
\end{align*}
Why continue?

The Z-algorithm is:

The Z-algorithm is: $O( |P| + |T| )$ time

An alphabet-independent solution

The Z-algorithm is less good at:

Searching for a set of patterns (Aho-Corasick)

Running in sub-linear* time (Boyer-Moore)

* — in practice, not theory
Exact pattern matching with Boyer-Moore

Boyer Moore preprocesses the pattern $P$

$$\text{Preprocess} \approx O(|P|)$$

$T$

Boyer-Moore $\approx O(|P| + |T|)$

Find instances of $P$ in $T$

‘instances’: An exact, full length copy
Boyer-Moore

**Intuition:** Learn from alignments to avoid others

\[ P: \text{cat} \]
\[ T: \text{carl carried the cat} \]
\[ \text{cat} \]
\[ \text{0 1 2 3 4 5 6 7 8 9} \ldots \]

What does this alignment tell us?
Intuition: Learn from alignments to avoid others

\[ P: \text{cat} \]

\[ T: \text{carl carried the cat} \]

\[ \text{cat} \quad \text{car} \quad \text{r} \quad \text{i} \quad \text{e} \quad \text{d} \quad \text{the} \quad \text{cat} \]

What does this alignment tell us?

1) Our pattern doesn’t match at this alignment

\[ \text{car} \]

There is no ‘r’ in ‘cat’!
Boyer-Moore

**Intuition:** Learn from alignments to avoid others

- \( P: \text{cat} \)
- \( T: \text{carl carried the cat} \)

\[
\begin{array}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \ldots \\
\hline
\text{car} & \text{l} & \text{car} & \text{r} & \text{i} & \text{e} & \text{d} & \text{the} & \text{cat} & & \\
\text{cat} & & & & & & & & & & \\
\end{array}
\]

What does this alignment tell us?

2) Our pattern doesn’t match at *later* alignments

There is no ‘r’ in ‘cat’!
Boyer-Moore

**Intuition:** Learn from alignments to avoid others

\[ P: \text{cat} \]

\[ T: \text{carl carried the cat} \]

\[ \text{cat} \]

What does this alignment tell us?

2) Our pattern doesn’t match at *later* alignments

There is no ‘r’ in ‘cat’!
Boyer-Moore

**Intuition:** Learn from alignments to avoid others

\[ P: \text{cat} \]

\[ T: \text{carried the cat} \]

\[ \text{ca} \quad \text{ta} \]

\[ \text{t} \quad \text{skip!} \quad \text{t} \quad \text{skip!} \quad \text{t} \]

**What does this alignment tell us?**

2) Our pattern doesn’t match at *later* alignments

\[ \text{car} \quad \text{There is no ‘r’ in ‘cat’!} \]

\[ \text{cat} \]
Boyer-Moore

**Intuition:** Learn from alignments to avoid others

\[ P: \text{word} \]

\[ T: \text{There would have been a } \ldots \]

\[ \underline{\text{word}} \]

\[ 0 1 2 3 4 5 6 7 8 9 \ldots \]
Boyer-Moore

**Intuition:** Learn from alignments to avoid others

\[ P: \text{word} \]

\[ T: \text{There would have been a } \ldots \]

0 1 2 3 4 5 6 7 8 9 ...

1) Our pattern doesn’t match at this alignment

\[ T: \text{ould} \]

\[ P: \text{word} \]
Boyer-Moore

**Intuition:** Learn from alignments to avoid others

\[ P: \text{word} \]
\[ T: \text{There would have been a ...} \]
\[ \quad \text{word} \quad \]
\[ 0 1 2 3 4 5 6 7 8 9 ... \]

How many alignments can we skip?

2) Our pattern doesn’t match at *later* alignments

\[ T: \text{woul} \]
\[ P: \text{word} \]

There is no ‘u’ in ‘word’!
Boyer-Moore

**Intuition:** Learn from alignments to avoid others

\[ P: \text{word} \]

\[ T: \text{There would have been a …} \]

\[ \cdots \text{word} \cdots \]

\[ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ … \]

How many alignments can we skip? 2

2) Our pattern doesn’t match at *later* alignments

\[ T: \text{woul} \]

\[ P: \text{word} \]

There is no ‘u’ in ‘word’!
Boyer-Moore

**Intuition:** Learn from alignments to avoid others

\[ P: \text{word} \]

\[ T: \text{There would have been a} \ ... \]

\[ \text{---word---} \]

word skip!

word skip!

word

How many alignments can we skip? \[ 2 \]

2) Our pattern doesn’t match at *later* alignments

\[ T: \text{woul} \]

\[ P: \text{word} \]

There is no ‘u’ in ‘word’!
Boyer-Moore

**Intuition:** Learn from alignments to avoid others

\[ P: \text{T A G A C} \]
\[ T: \text{G T A G A T G G C T G A T C G A G T A G C G G C G} \]

How many alignments can we skip? 3

There IS a T in ‘TAGAC’!
**Boyer-Moore**

**Intuition:** Learn from alignments to avoid others

\[ P: \text{TAGAC} \]

\[ T: \text{GTAGATGGCTGATCGAGTAGCGGCG} \]

3

How many alignments can we skip? 3

There IS a T in ‘TAGAC’!
Boyer-Moore

**Intuition:** Learn from alignments to avoid others

- **P:** A A B B B
- **T:** A A A B A B A A A A A A A A A A A A A A A A A A A A A A A A A A A A

How many alignments can we skip? 1

AABAB

There IS an A in ‘AAABB’!
Boyer-Moore

**Intuition:** Learn from alignments to avoid others

\[ P: A A B B B \]
\[ T: A A A B A B A A A A A A A A A A A A A A A A A A A A A A A A A \]

- A A B B B  
  A A B B B  skip!
  A A B B B  the *first* match we encounter!

How many alignments can we skip? \( \frac{1}{5} \)

AABAB

There IS an A in ‘AAABB’!

AABB
Boyer-Moore: Bad Character rule

Upon mismatch, skip alignments until (a) mismatch becomes a match, or (b) $P$ moves past mismatched character. (c) If there was no mismatch, don't skip.

Step 1:

$$T: \text{CCTTCTGC} \text{TACCTTTGCGCGCGCGCGGAA}$$

$$P: \text{CCTTTTGC}$$

Case (a)

Step 2:

$$T: \text{CCTTCTGC} \text{TACCTTTGCGCGCGCGCGGAA}$$

$$P: \text{CCTTTTGC}$$

Case (b)

Step 3:

$$T: \text{CCTTCTGC} \text{TACCTTTGCGCGCGCGCGGAA}$$

$$P: \text{CCTTTTGC}$$

Case (b)

(etc)

Step 7:

$$T: \text{CCTTCTGC} \text{TACCTTTGCGCGCGCGCGGAA}$$

$$P: \text{CCTTTTGC}$$

Case (c)
Boyer-Moore: Bad Character rule

Step 1:
\[ T: \text{CCTTCTGCTACCTTTTGCGCGCGCGCGGAA} \]
\[ P: \text{CCTTTTGC} \]
\[ \text{skip!} \]

Step 2:
\[ T: \text{CCTCTGCTACCTTTTGCGCGCGCGCGGAA} \]
\[ P: \text{CCCTTTTGCG} \]
\[ \text{skip!} \]

Step 3:
\[ T: \text{CCTCTGCTACCTTTTGCGCGCGCGCGGAA} \]
\[ P: \text{CCCTTTTGCG} \]
\[ \text{skip!} \]

We skipped three alignments

Can we do anything to make this better?
Boyer-Moore: Bad Character rule

Which of the following alignments skips the most?

A) $T$: TATAT...  
   $P$: TAGAC

B) $T$: TTGAT...  
   $P$: TAGAC

C) $T$: TAGAT...  
   $P$: TAGAC

D) $T$: TAGTT...  
   $P$: TAGAC
Boyer-Moore: Bad Character rule improvement

Continue to test alignment from left-to-right

... but compare *characters* from right to left.

\[ \begin{align*}
P: & \quad \text{T A G A C} \\
T: & \quad \text{G T A G A T G G C T G A T C G A G T A G C G G C G} \\
    & \quad \text{--- T A G A C ---} \\
    & \quad \text{--- --- --- ---} \\
\end{align*} \]
Right-to-left-scanning w/ BC Rule

\[ P: \text{word} \]

\[ T: \text{There would have been a ... word} \]

There is no ‘l’ in ‘word’!

How many alignments do we skip?
Right-to-left-scanning w/ BC Rule

P: word

T: There would have been a ...

word

word

word

How many alignments do we skip? 3
Right-to-left-scanning w/ BC Rule

Upon mismatch, skip alignments until (a) mismatch becomes a match, or (b) $P$ moves past mismatched character. (c) If there was no mismatch, don't skip.

Step 1:

$T$: CCTTCGCTACCTTTTGCGCGCGCGGAA
$P$: CCTTGGC

Case (a)

Step 2:

$T$: CCTTCGCTACCTTTTGCGCGCGCGGAA
$P$: CCTTGGC

Case (b)

Step 3:

$T$: CCTTCGCTACCTTTTGCGCGCGCGGAA
$P$: CCTTGGC

Case (c)

Step 4:

$T$: CCTTCGCTACCTTTTGCGCGCGCGGAA
$P$: CCTTGGC

Case (a)

(etc)
Right-to-left-scanning w/ BC Rule

Step 1:
T: C C T T C T G C T A C C T T T T G C G G C G C G C G G A A
P: C C T T T T G C

Step 2:
T: C C T T C T G C T A C C C T T T T G C G G C G C G C G G A A
P: C C T T T T G C

Step 3:
T: C C T T C T G C T A C C C T T T T G C G G C G C G C G G A A
P: C C T T T T G C

Up to step 3, we skipped 8 alignments

5 characters in T were never looked at
Right-to-left-scanning w/ BC Rule

Learn from character comparisons to skip pointless alignments

1. When we hit a mismatch \( c \), move \( P \) along until \( c \) becomes a match (or \( P \) moves past \( c \))

2. Try alignments in one direction, but do character comparisons in opposite direction

How do we put the first two rules in practice?

Exact pattern matching \textbf{w/ Boyer-Moore}

Boyer Moore \textbf{preprocesses} the pattern

\[ \text{Preprocess} \approx O(|P|) \]

\[ \text{Boyer-Moore} \approx O(|P| + |T|) \]

Find instances of \( P \) in \( T \)

‘instances’: An exact, full length copy
Boyer-Moore: BC rule preprocessing

Preprocessing requires two args: \( P: \text{T C G C} \quad \Sigma: \text{A C G T} \)

The goal is to produce a table which tracks *skips*

\[
\begin{array}{cccc}
\text{T} & \text{C} & \text{G} & \text{C} \\
\hline
\text{A} & & & \\
\text{C} & & & \\
\text{G} & & & \\
\text{T} & & & \\
\end{array}
\]
Preprocessing requires two args:

$P: \text{T C G C}$

$\Sigma: \text{A C G T}$

The goal is to produce a table which tracks *skips*.

<table>
<thead>
<tr>
<th>$\Sigma$</th>
<th>$P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>T</td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>
Boyer-Moore: BC rule preprocessing

Preprocessing requires two args: $P: \text{T C G C}$ $\Sigma: \text{A C G T}$

The goal is to produce a table which tracks *skips*

The table is as follows:

<table>
<thead>
<tr>
<th>$\Sigma$</th>
<th>$T$</th>
<th>$C$</th>
<th>$G$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$G$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T$</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

$T: \ ? \ ? \ ? \ T \ ? \ ? \ ? \ ? \ ? \ ? \ ?$ $P: \ ? \ ? \ ? \ ? \ ? \ ? \ ? \ ? \ ? \ ? \ ? \ \text{Preprocessing}$

$P: \text{T C G C}$
Boyer-Moore: BC rule preprocessing

Preprocessing requires two args: \( P : T \ C \ G \ C \) \( \Sigma : A \ C \ G \ T \)

The goal is to produce a table which tracks \textit{skips}
Boyer-Moore: BC rule preprocessing

Preprocessing requires two args: $P \colon T \ C \ G \ C \ \\ 
\Sigma \colon A \ C \ G \ T$

The goal is to produce a table which tracks *skips*

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>C</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

$T: \ ? \ ? \ ? \ A \ ? \ ? \ ? \ ? \ ? \ ?$

$P: \ T \ C \ G \ C$

$P$ is the pattern, $\Sigma$ is the alphabet.
Boyer-Moore: BC rule preprocessing

Preprocessing requires two args: \( P: T \ C \ G \ C \) \( \Sigma: A \ C \ G \ T \)

The goal is to produce a table which tracks *skips*

\[
\begin{array}{c|cccc}
\Sigma & T & C & G & C \\
\hline
A & 0 & 1 & 2 & 3 \\
C & 0 & - & 0 & - \\
G & 0 & 1 & - & 0 \\
T & - & 0 & 1 & 2 \\
\end{array}
\]

\[
\begin{array}{c|cccc}
P & T & C & G & C \\
\hline
\end{array}
\]
Boyer-Moore: BC rule preprocessing

Preprocessing requires two args: $P$: B A B A A A B $\Sigma$: A B

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>A</th>
<th>B</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Sigma$</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Boyer-Moore: BC rule preprocessing

Preprocessing requires two args: $P: \text{B A B A A A B}$ \hspace{1cm} $\Sigma: \text{A B}$

For each character $p$ in pattern $P$

For each character $c$ in alphabet $\Sigma$

Find the closest previous instance of $p$ (to the left of $c$).

<table>
<thead>
<tr>
<th>$\Sigma$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Boyer-Moore: BC rule preprocessing

Preprocessing requires two args:  \( P: B A B A A A B \)  \( \Sigma: A B \)

For each character \( p \) in pattern \( P \)

For each character \( c \) in alphabet \( \Sigma \)

Find the closest previous instance of \( p \) (to the left of \( c \)).

Pattern

\[
\begin{array}{cccccccc}
A & 0 & 1 & 0 & 1 \\
B & 0 & 0 & 1 & 0 \\
\end{array}
\]

\( \Sigma \)
Boyer-Moore: BC rule preprocessing

Preprocessing requires two args: \( P: \text{BABAAAB} \) \( \Sigma: \text{AB} \)

For each character \( p \) in pattern \( P \)

For each character \( c \) in alphabet \( \Sigma \)

Find the closest previous instance of \( p \) (to the left of \( c \)).

\[
\begin{array}{cccccccc}
\hline
\hline
A & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
B & 0 & 0 & 1 & 0 & 1 & 2 & 3 \\
\hline
\end{array}
\]
Assignment 4: a_bmoore

Learning Objective:

Implement preprocessing of patterns with Boyer-Moore*

Observe Boyer-Moore* efficiency as a heuristic

Consider: Optimal preprocessing is \( \theta(|P| \cdot |\Sigma|) \). Can you code it?
Boyer-Moore: Using the BC Table

Try alignments from left-to-right and match characters from right-to-left

When we encounter a mismatch, skip the calculated number of alignments
Boyer-Moore: Using the BC Table

Try alignments from left-to-right and match characters from right-to-left

When we encounter a mismatch, skip the calculated number of alignments

\[
\begin{array}{cccc}
T & C & G & C \\
A & 0 & 1 & 2 & 3 \\
C & 0 & - & 0 & - \\
G & 0 & 1 & - & 0 \\
T & - & 0 & 1 & 2 \\
\end{array}
\]

\( T: \text{GGGGGGGGGGGGG} \)

\( P: \text{TCCG} \)
Boyer-Moore: Using the BC Table

Try alignments from left-to-right and match characters from right-to-left

When we encounter a mismatch, skip the calculated number of alignments

<table>
<thead>
<tr>
<th>Σ</th>
<th>T</th>
<th>C</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ \Sigma \]

\[ P \]

\[ T: A A T C A A T A G C \]

\[ P: T C G C \]
Boyer-Moore: Tracking total skips

\[ P \]
\[
\begin{array}{cc}
A & A \\
A & 0 & 0 \\
B & 0 & 1 \\
\end{array}
\]

\[ \Sigma \]

\[ T: \text{B B B B} \]

\[ T: \text{B B B B B} \]

\[ T: \text{B B B B B B} \]
Boyer-Moore: Tracking total skips

<table>
<thead>
<tr>
<th>Σ</th>
<th>P</th>
<th>T: B B B B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Assignment 4: a_bmoore

Learning Objective:

Implement preprocessing of patterns with Boyer-Moore*

Observe Boyer-Moore* efficiency as a heuristic

Consider: Our Boyer-Moore is theoretically slower than Z-algorithm.

But is it slower in practice? What is our total character comparisons?
A complete bonus lecture!
A better Boyer-Moore

Learn from character comparisons to skip pointless alignments

1. When we hit a mismatch $c$, move $P$ along until $c$ becomes a match (or $P$ moves past $c$) “Bad character rule”

2. Try alignments in one direction, but do character comparisons in \textit{opposite} direction “Right-to-left scanning”

Is this $O(|P| + |T|)$?
Worst-Case Bad Character rule

Upon mismatch, skip alignments until (a) mismatch becomes a match, or (b) $P$ moves past mismatched character. (c) If there was no mismatch, don't skip

Step 1:

$T$: A A A A A A A A A A A A A A A A A A A A A A A A A A

$P$: A A A

Case (c)

Step 2:

$T$: A A A A A A A A A A A A A A A A A A A A A A A A A A

$P$: A A A

Case (c)

Step 3:

$T$: A A A A A A A A A A A A A A A A A A A A A A A A A A

$P$: A A A

Case (c)

Step 4:

$T$: A A A A A A A A A A A A A A A A A A A A A A A A A A

$P$: A A A

Case (c)

(etc)

Using just bad character, $O(|P| \times |T|)$
A better Boyer-Moore

The complete Boyer-Moore algorithm, *with all refinements*, is $O(|P| + |T|)$.

Refinements include:
- "strong" good suffix rule
- Galil rule

We will be covering the ‘weak’ good suffix rule

If interested in refinements, see Gusfield textbook (syllabus) or contact me for details
**Intuition:** Learn from alignments to avoid others

\[ P: \text{A C A T A C} \]

\[ T: \text{A C A G A C A T A C A T G A C A G T G A C C A} \]

\[ \text{``A C A T A C''} \]

What does this alignment tell us?
“Weak” Good Suffix rule

**Intuition:** Learn from alignments to avoid others

\[ P: \quad A \ C \ A \ T \ A \ C \]

\[ T: \quad T \ A \ C \ A \ G \ A \ C \ A \ T \ A \ C \ A \ T \ G \ A \ C \ A \ G \ T \ G \ A \ C \ C \ A \]

We only want to look at alignments that are **at least as good** as our current alignment
“Weak” Good Suffix rule

**Intuition:** Learn from alignments to avoid others

\[ P: \text{A C A T A C} \]

\[ T: \text{T A C A GACATACATGACAGTGACCA} \]

\[ \text{‘A C A T A C} \]

What does partial match (the suffix ‘AC’) tell us?

Any alignment that overlaps this region of the text must match the suffix! So we can look for another ‘AC’ somewhere in the pattern!
“Weak” Good Suffix rule

**Intuition:** Learn from alignments to avoid others

\[ P: \text{TACATAC} \]

\[ T: \text{TACAGACATACATGACAGTGACCA} \]

Any alignment that overlaps this region of the text must match the suffix! So we can look for another ‘AC’ somewhere in the pattern!
“Weak” Good Suffix rule

**Intuition:** Learn from alignments to avoid others

\[
P: \text{ACATAC}
\]

\[
T: \text{TAGACATACATGACAGTGACCA}
\]

Any alignment that overlaps this region of the text must match the suffix! So we can look for another ‘AC’ somewhere in the pattern!

How many alignments do we skip? 3
“Weak” Good Suffix rule

**Intuition:** Learn from alignments to avoid others

\[ P: \text{ATC} \]
\[ T: \text{AGTAGCAGCACTAGA} \]

Any alignment that overlaps this region of the text must match the suffix! So we can look for another _______ somewhere in the pattern!

How many alignments do we skip?
"Weak" Good Suffix rule

**Intuition:** Learn from alignments to avoid others

\[
P: \ A \ T \ C
\]
\[
T: \ \textcolor{red}{A} \ G \ T \ A \ G \ \textcolor{red}{C} \ A \ G \ C \ A \ C \ A \ G \ T \ A \ G \ C \ A \ G \ C \ T \ A \ G \ A
\]

Any alignment that overlaps this region of the text must match the suffix! So we can look for another \textcolor{red}{C} somewhere in the pattern!

How many alignments do we skip? 2
“Weak” Good Suffix rule

**Intuition:** Learn from alignments to avoid others

\[ P: \text{G C A G C} \]

\[ T: \text{A G T} \boxed{\text{A G C}} \text{A G C A C A G T A G C A G C T A G A} \]

\[ \text{G C A G C} \]

Any alignment that overlaps this region of the text must match the suffix! So we can look for another _______ somewhere in the pattern!

How many alignments do we skip?
"Weak" Good Suffix rule

**Intuition:** Learn from alignments to avoid others

\[ P: \quad G \ C \ A \ G \ C \]

\[ T: \quad A \ G \ T \quad \text{AGC} \quad A \ G \ C \quad A \ C \ A \ G \ T \quad A \ G \ C \quad A \ G \ C \quad T \quad A \ G \ A \]

This is a full length match!

How many alignments do we skip?
“Weak” Good Suffix rule

Intuition: Learn from alignments to avoid others

\[ P: \text{G C A G C} \]

\[ T: \text{A G T} \text{AGC} \text{A G C A C A G T A G C A G C T A G A} \]

Any alignment that overlaps this region of the text must match the suffix! So we can look for another _______ somewhere in the pattern!

How many alignments do we skip?
“Weak” Good Suffix rule

**Intuition:** Learn from alignments to avoid others

\[ P: \text{G C A G C} \]

\[ T: \text{A G T} \square \text{G C A C A G T A G C A G C T A G A} \]

\[ \text{G C A G C} \]

Any alignment that overlaps this region of the text must match the suffix … *or have a prefix-suffix partial match!*

How many alignments do we skip?
“Weak” Good Suffix rule

Let $t =$ longest suffix match at alignment; skip until (a) we find another instance of $t$ or (b) $P$ moves past $t$

Step 1: $T$: CGTG C TAC TTTACTTACCTTACTTAC GCGAA  
$P$: C TTA C TTTAC  
$t$ occurs *in its entirety* to the left within $P$

Step 2: $T$: CGTG C TAC TTTACTTACCTTACTTAC GCGAA  
$P$: C TTA C TTTAC  
prefix of $P$ matches a *suffix* of $t$

Step 3: $T$: CGTG C C TACTTACCTTACTTACTTAC GCGAA  
$P$: C TTA C TTTAC  

An *instance* of $t$ is either a full match to the left within $P$ or a prefix of $P$ matches a suffix of $t$
Boyer-Moore: Putting it together

How to combine bad character and good suffix rules?

How many characters does bad character skip? 2 characters

How many characters does good suffix skip? 7 characters

Take the maximum (7)!
Boyer-Moore: Putting it together

Use bad character or good suffix rule, *whichever skips more*

Step 1:

**T:** GTTATAGC

**P:** GTAGCGGC

*T: GTTATAGCTGATCGCGGGTAGCGGCGAA

**P:** GTAGCGGCG

*bc: 6, gs: 0* bad character

Step 2:

**T:** GTTATAGCTGATCGCGGGTAGCGGCGAA

**P:** GTAGCGGC

*T: GTTATAGCTGATCGCGGGTAGCGGCGAA

**P:** GTAGCGGC

*bc: 0, gs: 2* good suffix

Step 3:

**T:** GTTATAGCTGATCGCGGGTAGCGGCGAA

**P:** GTAGCGGC

*T: GTTATAGCTGATCGCGGGTAGCGGCGAA

**P:** GTAGCGGC

*bc: 2, gs: 7* good suffix

Step 4:

**T:** GTTATAGCTGATCGCGGGTAGCGGCGAA

**P:** GTAGCGGC

*T: GTTATAGCTGATCGCGGGTAGCGGCGAA

**P:** GTAGCGGC

*Good suffix*
Boyer-Moore: Putting it together

11 characters of $T$ ignored completely!

**Step 1:**

$T$: GTTATAGCTGATCGCGGCGTAGCGGCGAA  
$P$: GTAGCGGCG

**Step 2:**

$T$: GTTATAGCTGATCGCGGCGTAGCGGCGAA  
$P$: GTAGCGGCG

**Step 3:**

$T$: GTTATAGCTGATCGCGGCGTAGCGGCGAA  
$P$: GTAGCGGCG

**Step 4:**

$T$: GTTATAGCTGATCGCGGCGTAGCGGCGAA  
$P$: GTAGCGGCG

Skipped 15 alignments
Boyer-Moore

Learn from character comparisons to skip pointless alignments

1. When we hit a mismatch $c$, move $P$ along until $c$ becomes a match (or $P$ moves past $c$) 
   “Bad character rule”

2. Try alignments in one direction, but do character comparisons in opposite direction 
   “Right-to-left scanning”

3. When we move $P$ along, make sure characters that matched in the last alignment also match in the next alignment 
   “Good suffix rule”