String Algorithms and Data Structures

Boyer-Moore

CS 199-225
Brad Solomon

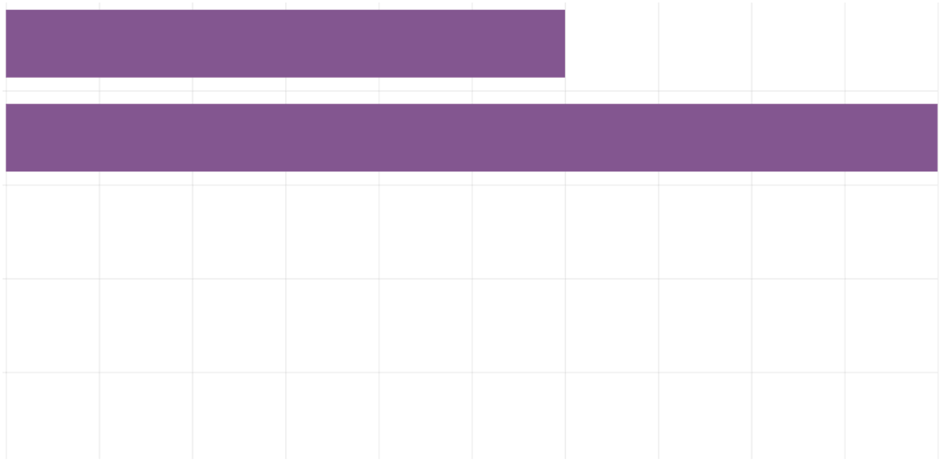
February 14, 2022

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

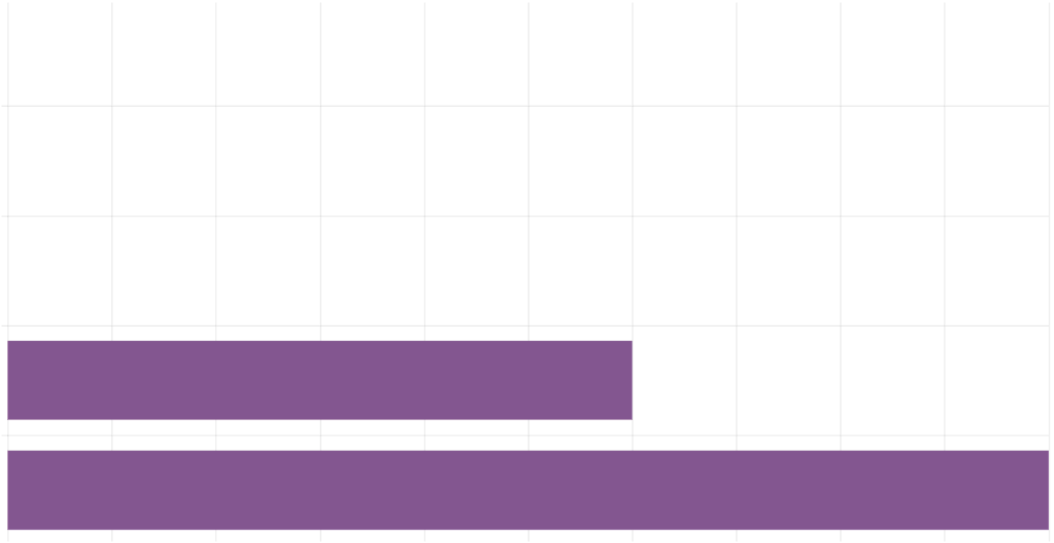
Department of Computer Science

A_zval reflection

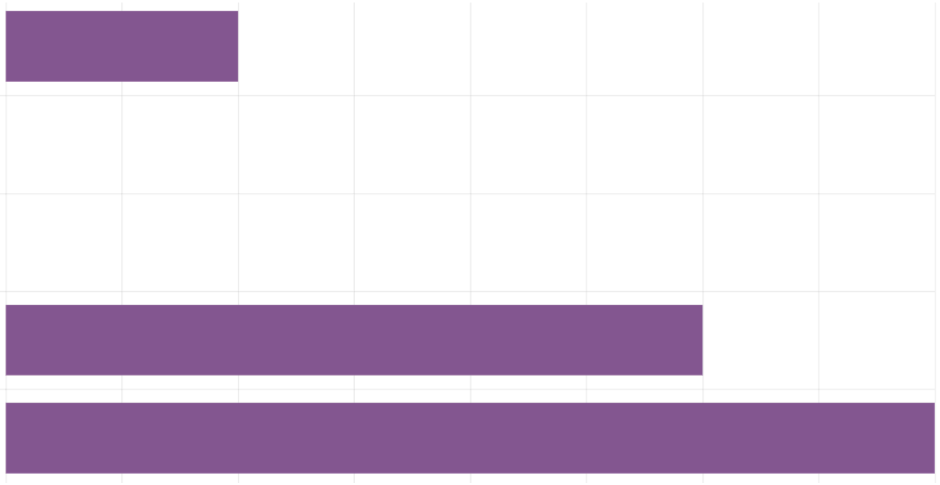
Time



Lecture Helpfulness



Learning Objectives met



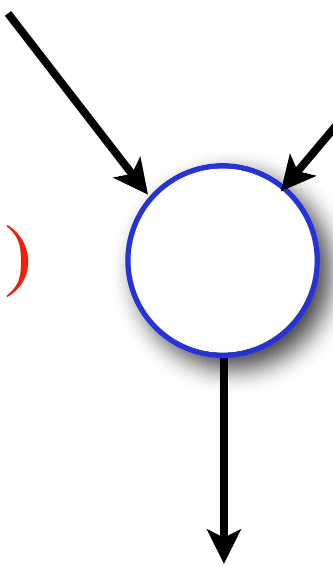
What was bad about the assignment?

A_zalg due today!

Remember you can re-use code from a_zval

You should not use code from the internet!

Exact Pattern Matching w/ Z-algorithm



Pattern, P

Text, T

Naive $\approx \theta(|P||T|)$

Z-Algorithm $\approx \theta(|P| + |T|)$

Find instances of *P* in *T*

'instances': An exact, full length copy

Why continue?

The Z-algorithm is:

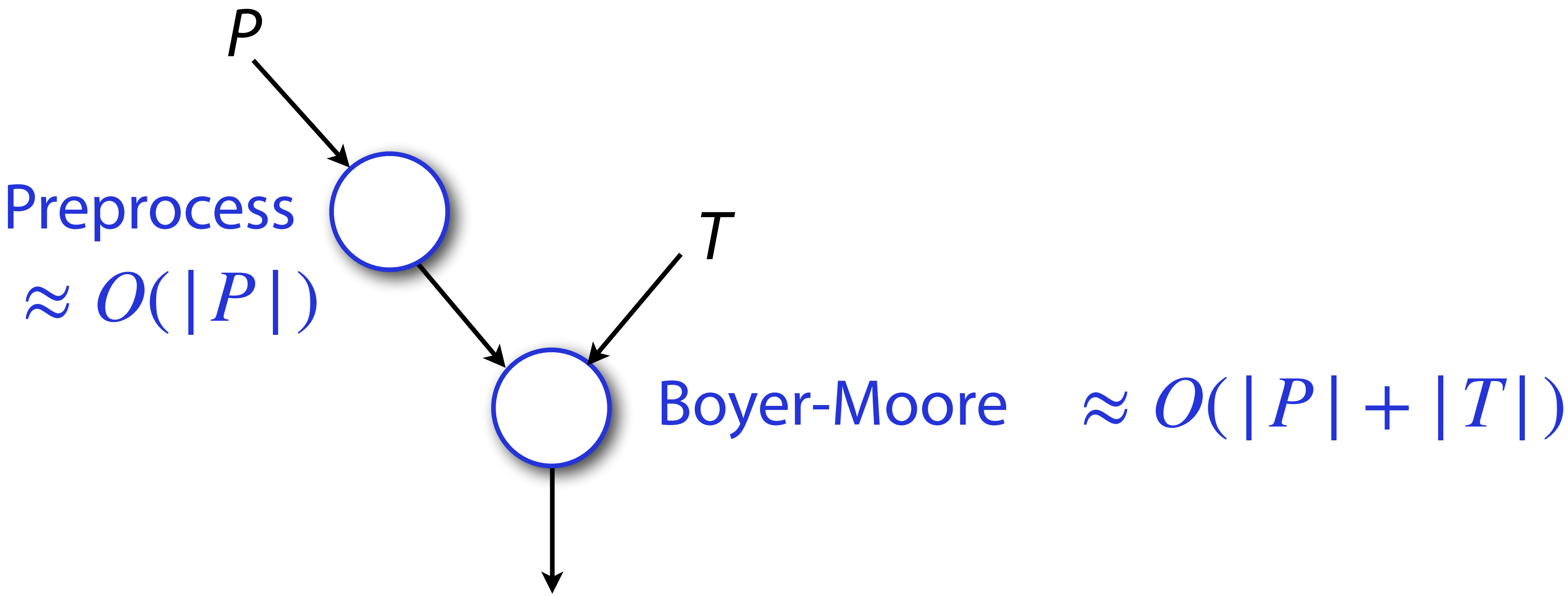
- The Z-algorithm is: $O(|P| + |T|)$ time
- An alphabet-independent solution

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 w/ Boyer-Moore

Boyer Moore **preprocesses** the pattern



P

Preprocess

$\approx O(|P|)$

T

Boyer-Moore

$\approx O(|P| + |T|)$

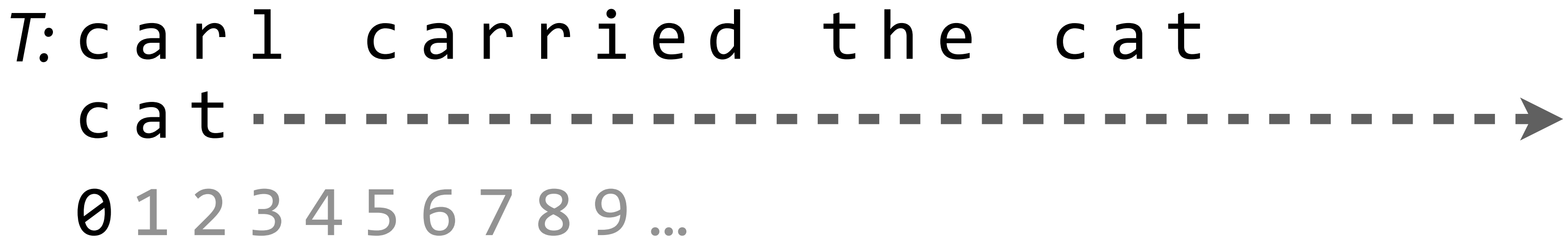
Find instances of *P* in *T*

'instances': An exact, full length copy

Boyer-Moore

Intuition: Learn from alignments to avoid others

P: `cat`

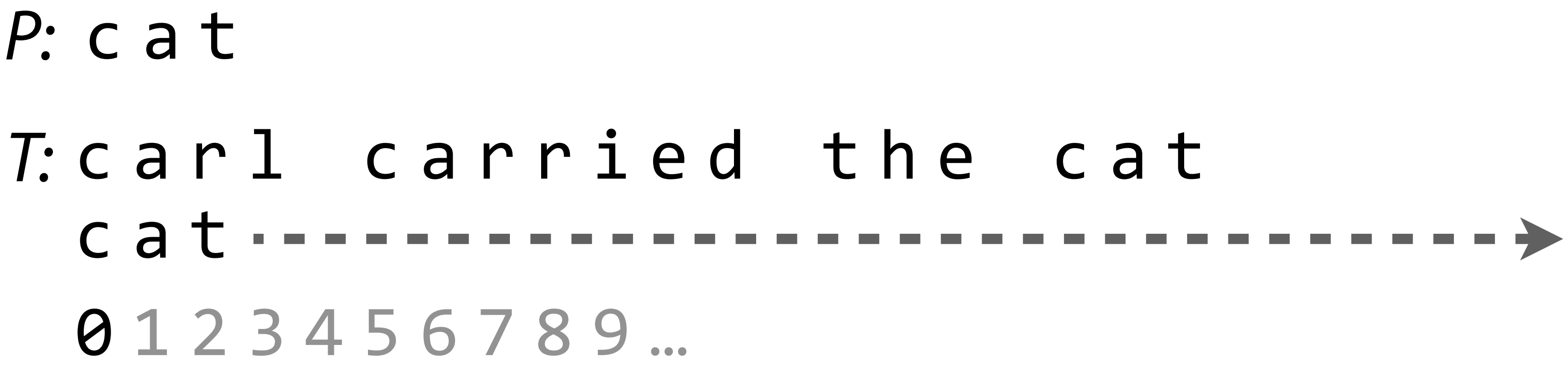


```
T: carl carried the cat
   cat - - - - - - - - - - - - - - - - - - ->
   0123456789...
```

What does this alignment tell us?

Boyer-Moore

Intuition: Learn from alignments to avoid others



P: cat

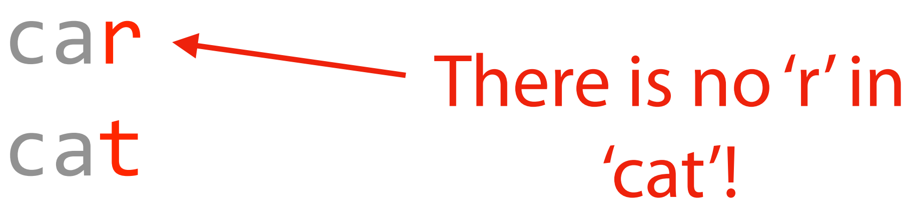
T: carl carried the cat

cat

0 1 2 3 4 5 6 7 8 9 …

What does this alignment tell us?

1) Our pattern doesn't match at this alignment



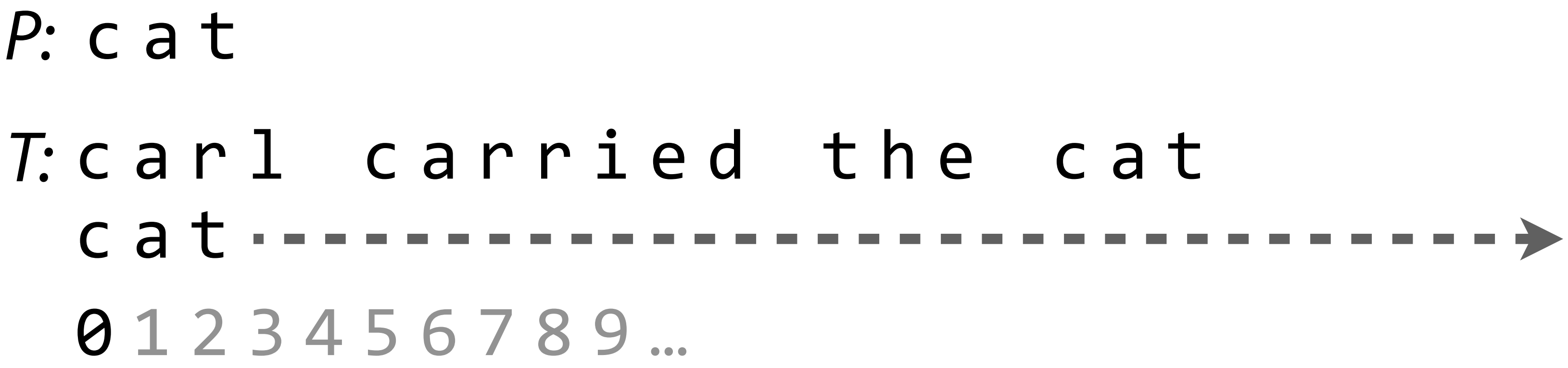
car

cat

There is no 'r' in 'cat'!

Boyer-Moore

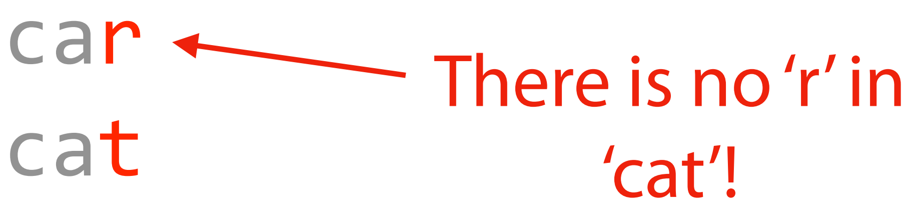
Intuition: Learn from alignments to avoid others



```
P: cat

T: carl carried the cat
   cat
   0 1 2 3 4 5 6 7 8 9 ...
```

What does this alignment tell us?

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

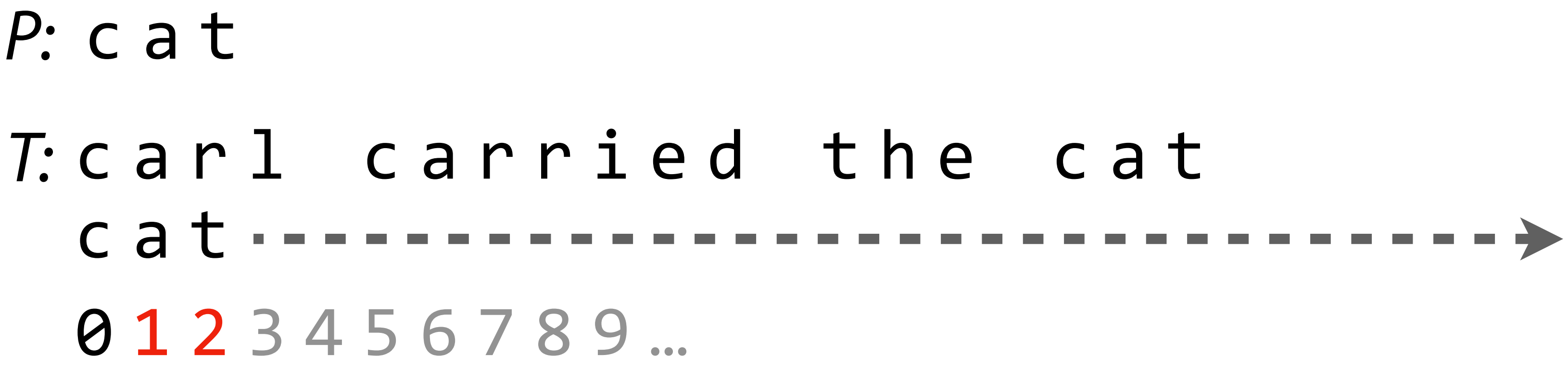


```
car
cat
```

There is no 'r' in 'cat'!

Boyer-Moore

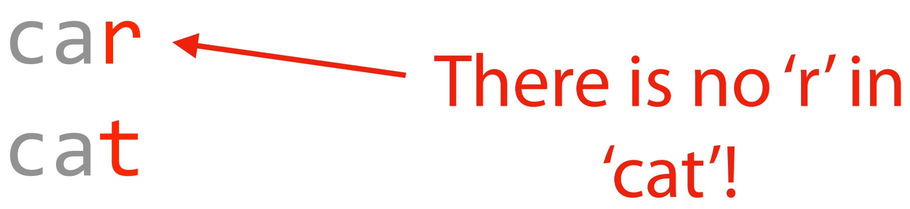
Intuition: Learn from alignments to avoid others



```
P: cat

T: carl carried the cat
   cat ------------------------->
   0123456789...
```

What does this alignment tell us?

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



```
car
cat
```

There is no 'r' in 'cat'!

Boyer-Moore

Intuition: Learn from alignments to avoid others

```
P: cat

T: carl carried the cat
   cat
    cat  skip!
     cat skip!
```

What does this alignment tell us?

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

```
car
cat
```

There is no 'r' in 'cat'!

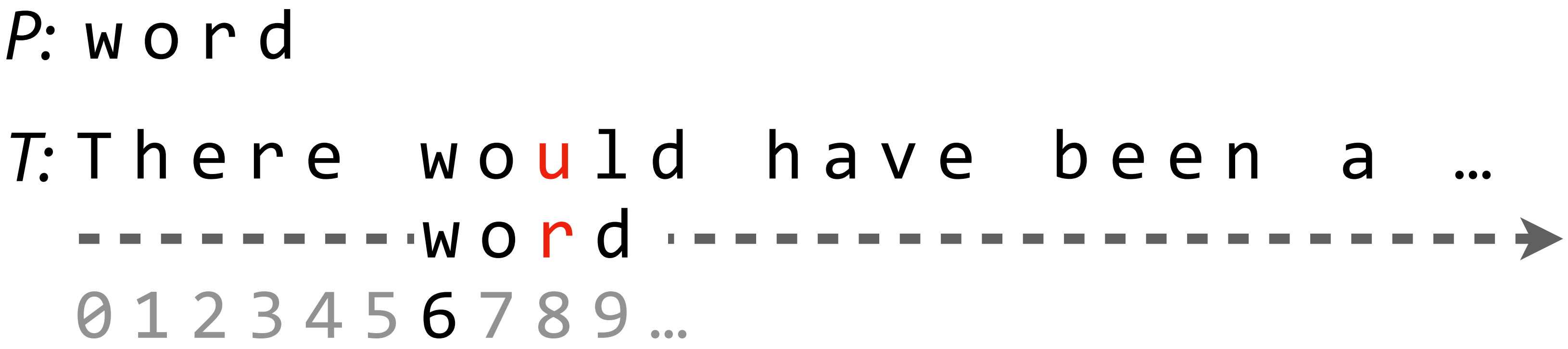
Boyer-Moore

Intuition: Learn from alignments to avoid others

```
P: word

T: There would have been a …
   ------word ---------------->
   0123456789…
```

Boyer-Moore

Intuition: Learn from alignments to avoid others



P: word

T: There would have been a …

word

0 1 2 3 4 5 6 7 8 9 …

1) Our pattern doesn’t match at this alignment

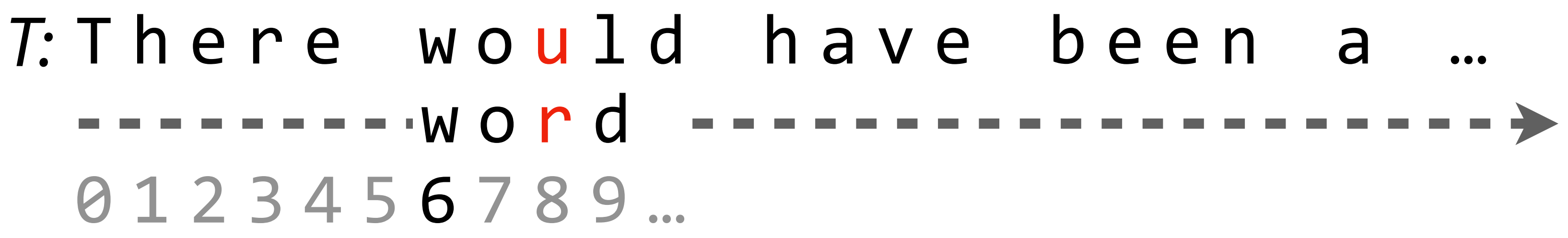
T: woul

P: word

Boyer-Moore

Intuition: Learn from alignments to avoid others

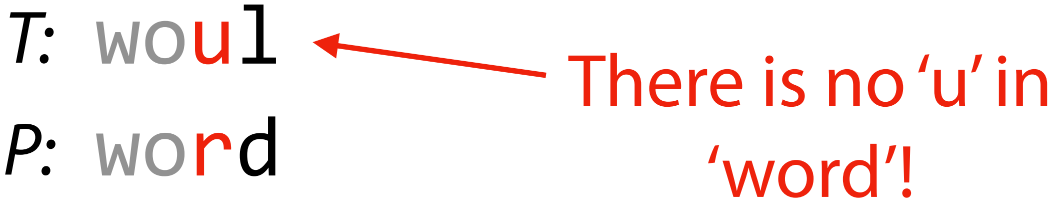
P: word



```
T: There would have been a …
   ------word ----------------->
   0123456789…
```

How many alignments can we skip?

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

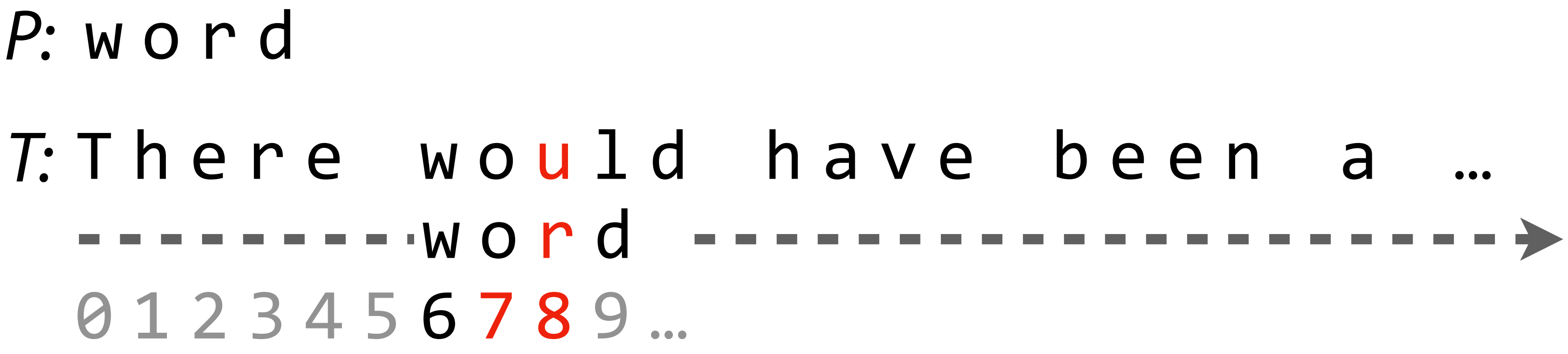


```
T: woul
P: word
```

There is no 'u' in 'word'!

Boyer-Moore

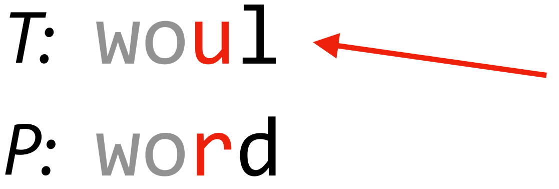
Intuition: Learn from alignments to avoid others



```
P: word

T: There would have been a …
         word
   0123456789…
```

How many alignments can we skip? 2

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



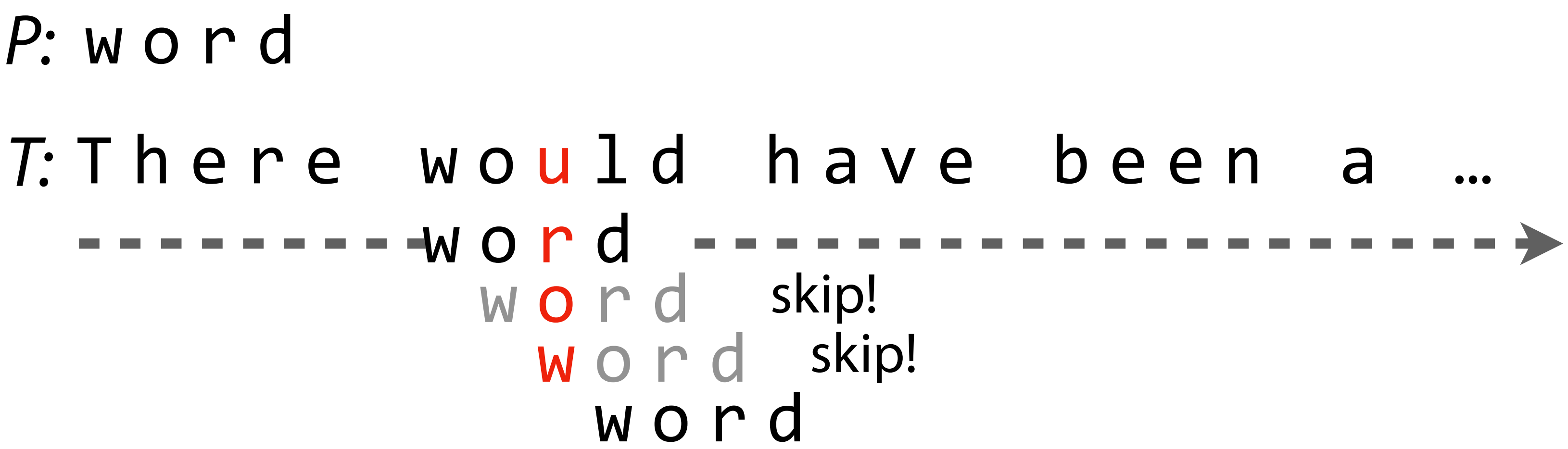
```
T: woul
P: word
```

There is no 'u' in 'word'!

Boyer-Moore

Intuition: Learn from alignments to avoid others

P: word



```
T: There would have been a ...
   ------word ------------------->
          word   skip!
           word   skip!
            word
```

How many alignments can we skip? 2

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

T: woul

P: word

There is no 'u' in 'word'!

Boyer-Moore

Intuition: Learn from alignments to avoid others

P: T A G A C

T: 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

T A G A C

How many alignments can we skip? 3

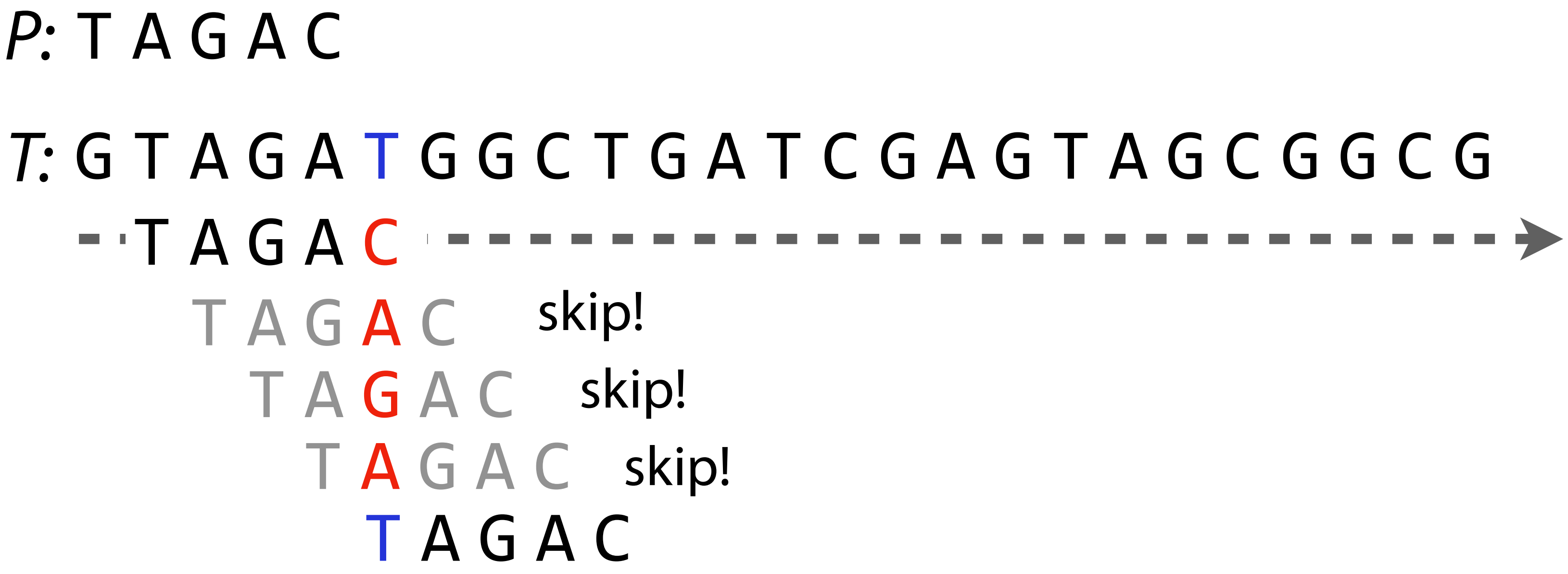
TAGAT

TAGAC

There IS a T in ‘TAGAC’!

Boyer-Moore

Intuition: Learn from alignments to avoid others



P: T A G A C

T: 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

T A G A C

T A G A C skip!

T A G A C skip!

T A G A C skip!

T A G A C

How many alignments can we skip? 3

TAGAT

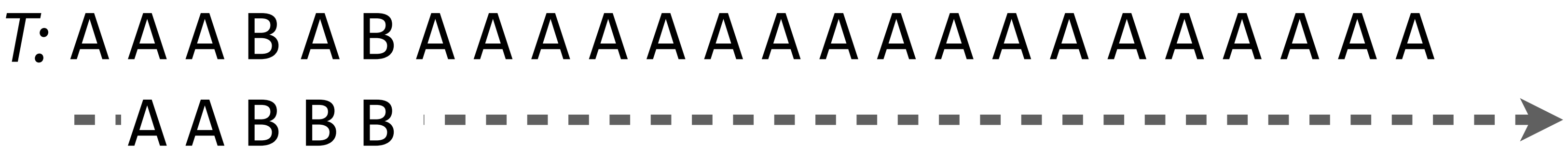
TAGAC

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 B B B

How many alignments can we skip?　　1

AABAB

AABBB

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 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? 1

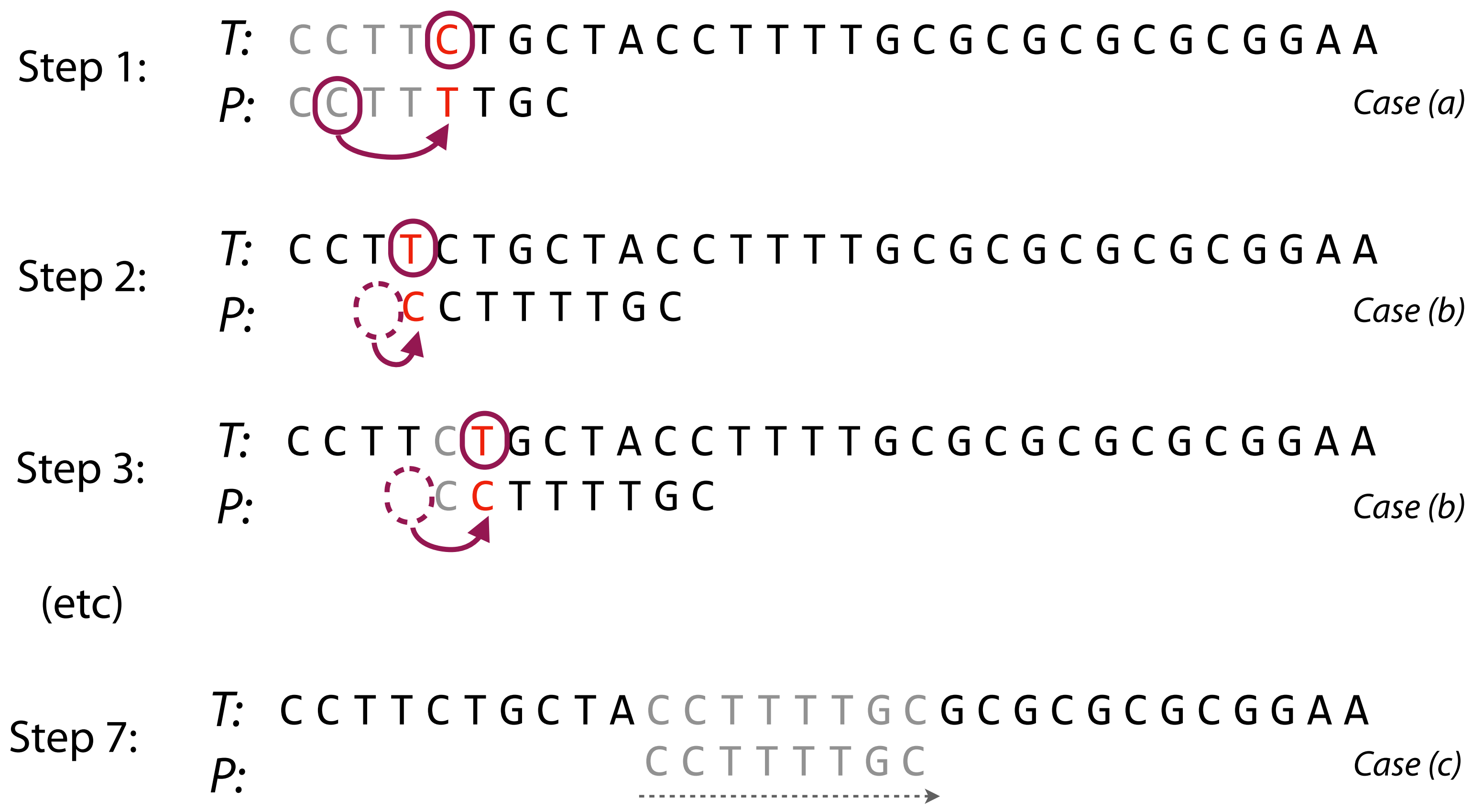
AABAB

AABBB

There IS an A in 'AAABB'!

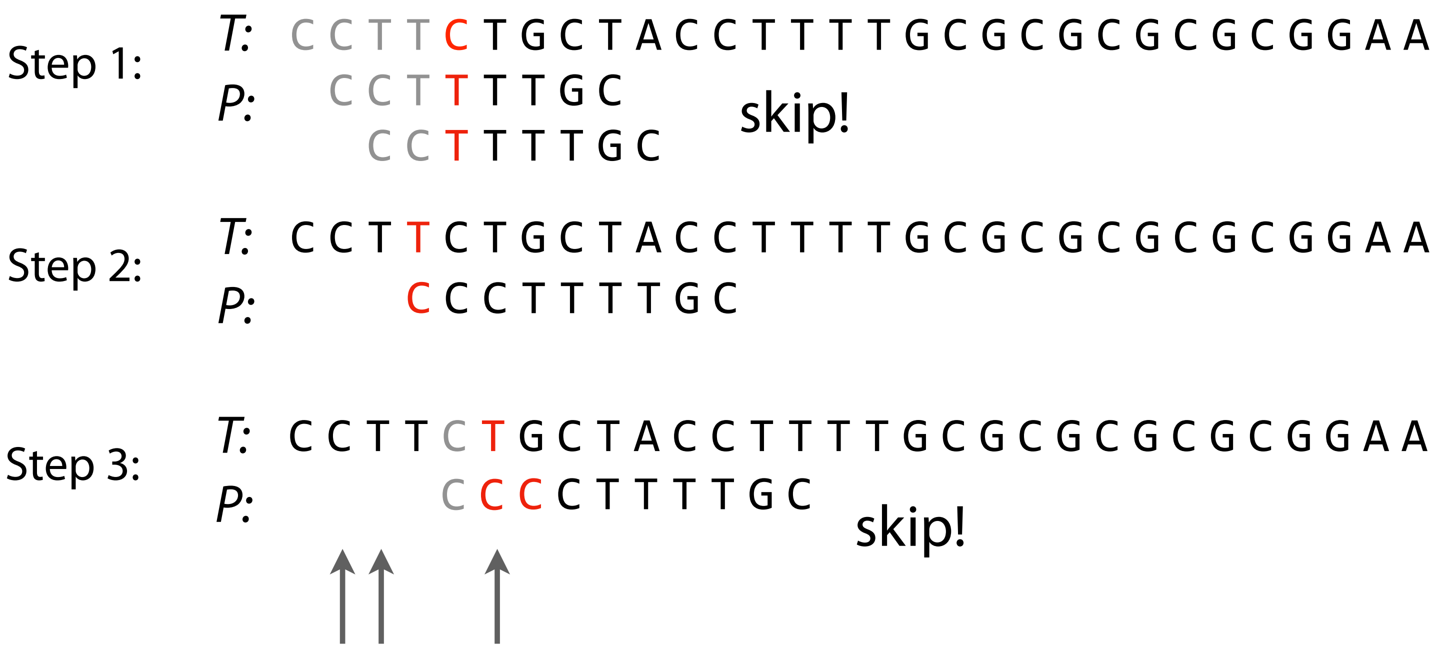
Boyer-Moore: Bad Character rule

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: C C T T C T G C T A C C T T T T G C G C G C G C G G A A
         P: C C T T T T G C                                          Case (a)

Step 2:  T: C C T T C T G C T A C C T T T T G C G C G C G C G G A A
         P:       C C T T T T G C                                    Case (b)

Step 3:  T: C C T T C T G C T A C C T T T T G C G C G C G C G G A A
         P:         C C T T T T G C                                  Case (b)

(etc)

Step 7:  T: C C T T C T G C T A C C T T T T G C G C G C G C G G A A
         P:                     C C T T T T G C                      Case (c)
```

Boyer-Moore: Bad Character rule



```
Step 1:  T: C C T T C T G C T A C C T T T T G C G C G C G C G G A A
         P:   C C T T T T G C
                C C T T T T G C          skip!

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

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

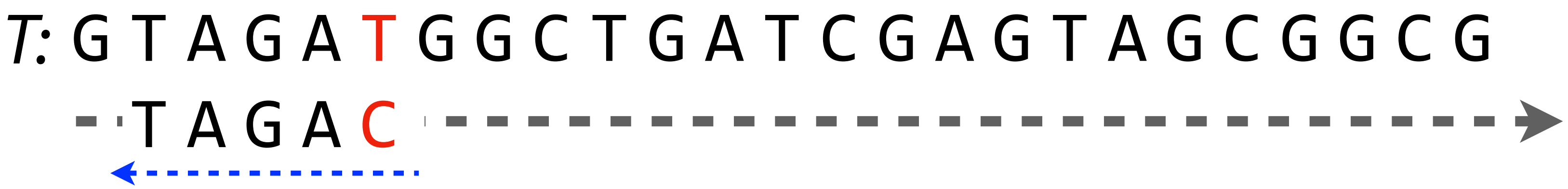
P: TAGAC

Boyer-Moore: Bad Character rule improvement

Continue to test alignment from left-to-right

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

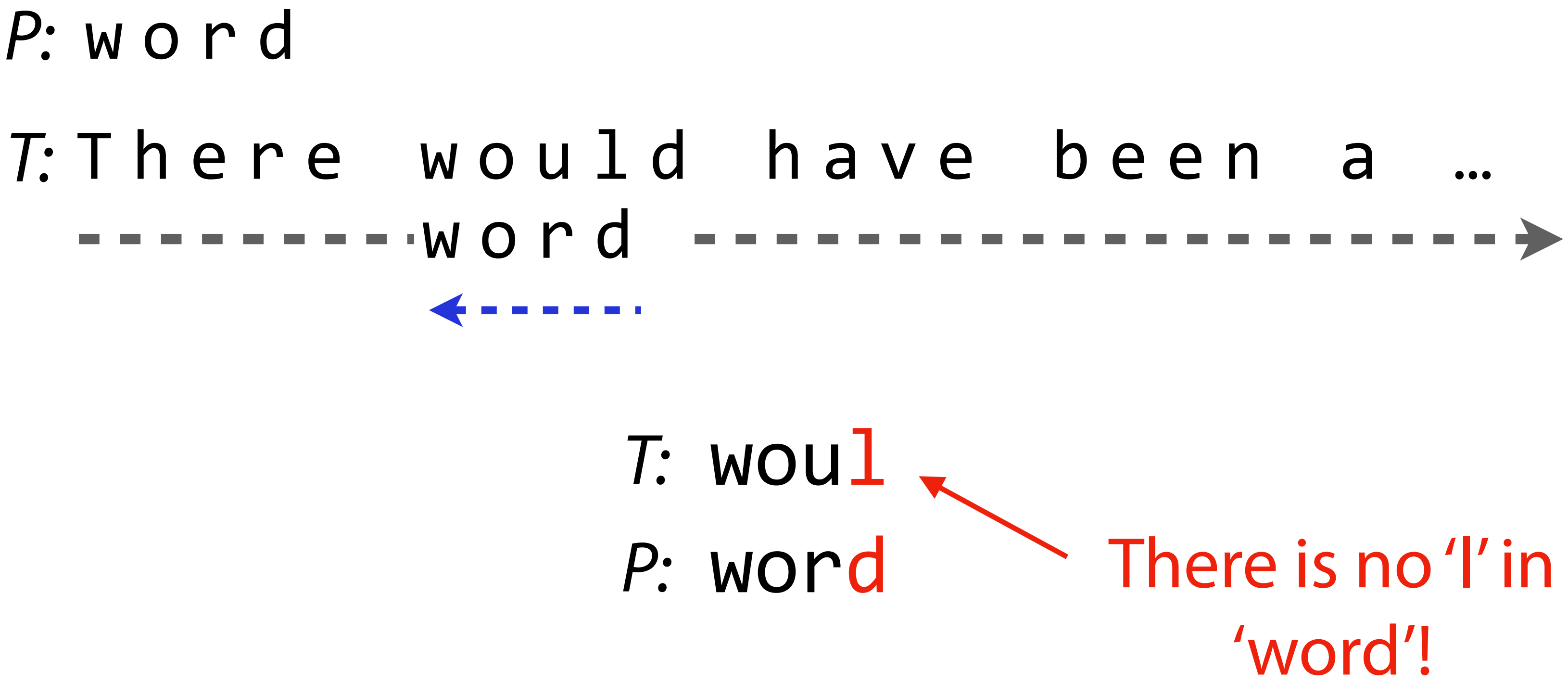
P: T A G A C



T: 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

T A G A C

Right-to-left-scanning w/ BC Rule



P: word

T: There would have been a …

word

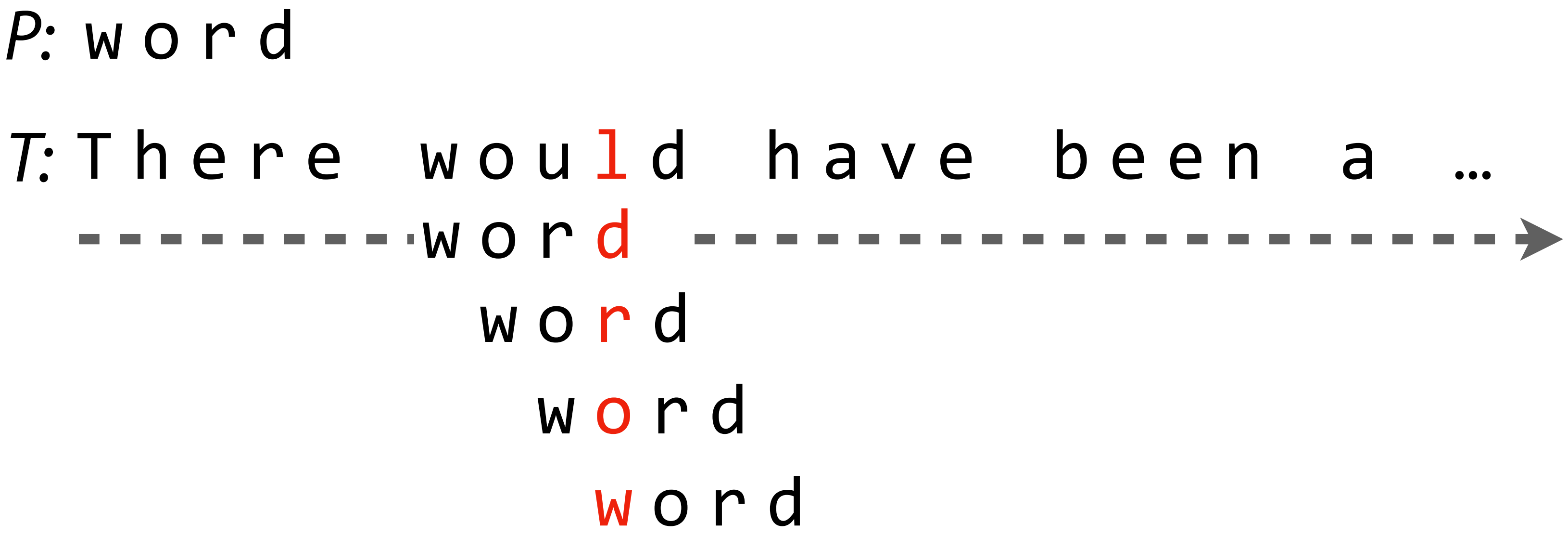
T: woul

P: word

There is no ‘l’ in ‘word’!

How many alignments do we skip?

Right-to-left-scanning w/ BC Rule

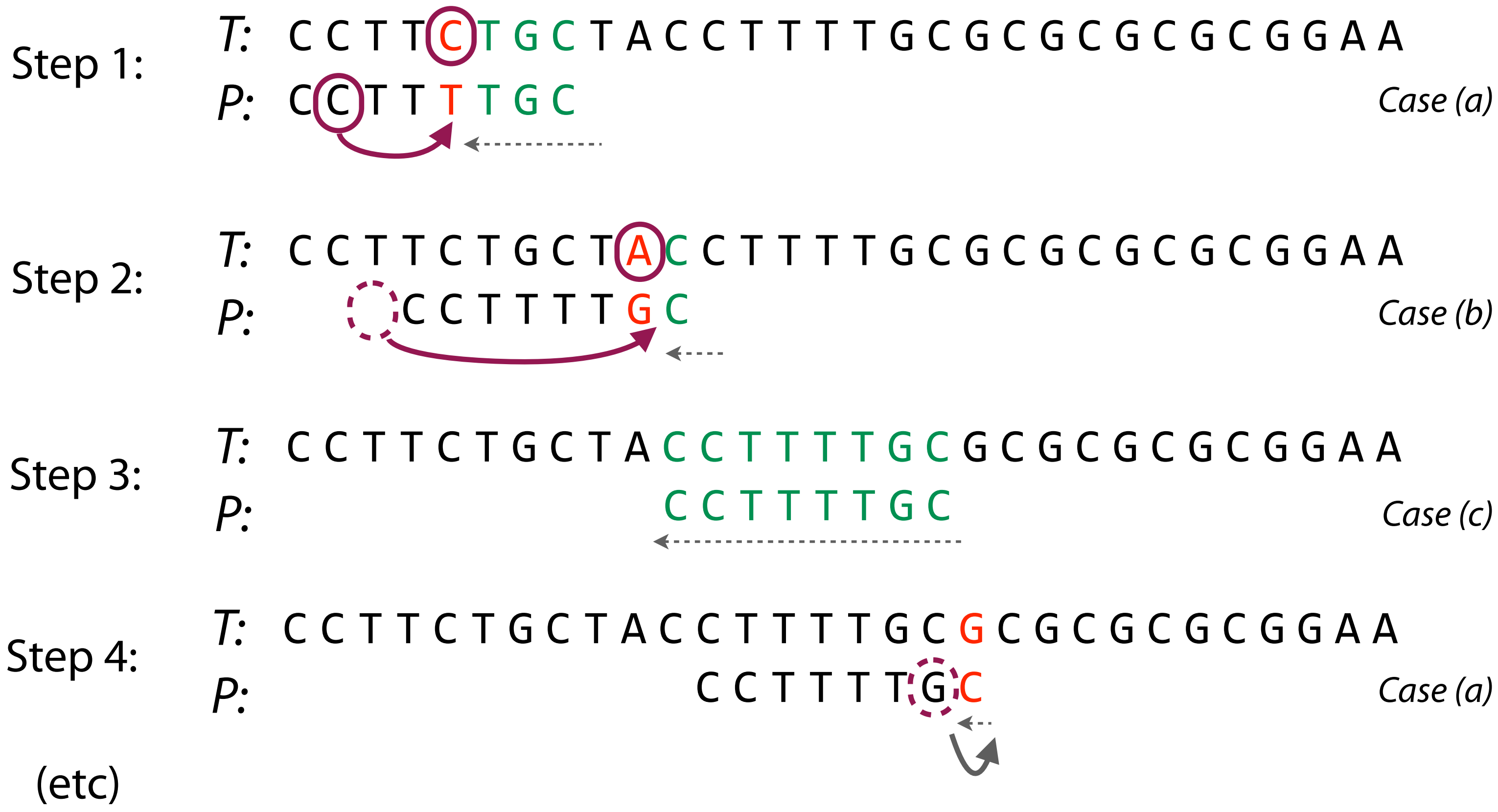


```
P: word

T: There would have been a ...
         word
          word
           word
            word
```

How many alignments do we skip? 3

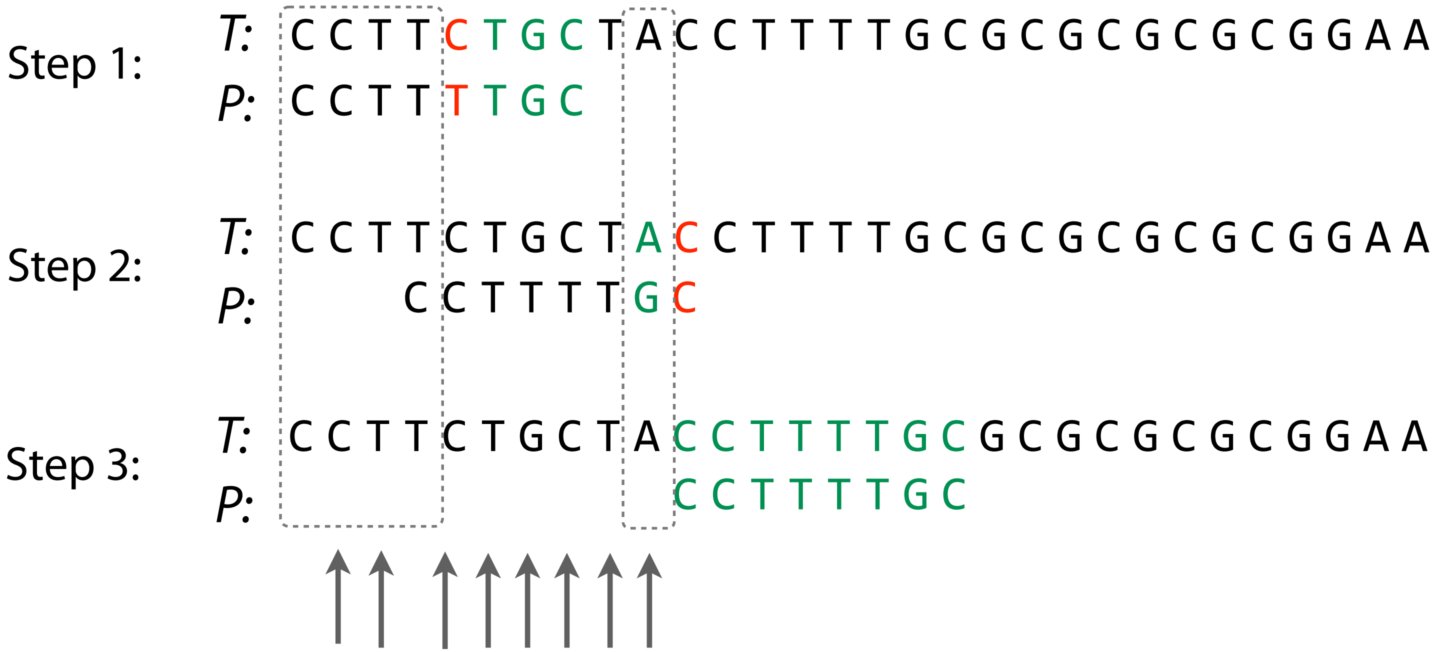
Right-to-left-scanning w/ BC Rule

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: CCTTCTGCTACCTTTTGCGCGCGCGGAA
         P: CCTTTTGC                        Case (a)

Step 2:  T: CCTTCTGCTACCTTTTGCGCGCGCGGAA
         P:    CCTTTTGC                     Case (b)

Step 3:  T: CCTTCTGCTACCTTTTGCGCGCGCGGAA
         P:           CCTTTTGC              Case (c)

Step 4:  T: CCTTCTGCTACCTTTTGCGCGCGCGGAA
         P:            CCTTTTGC             Case (a)

(etc)
```

Right-to-left-scanning w/ BC Rule



```
Step 1:  T: CCTTCTGCTACCTTTTGCGCGCGCGGAA
         P: CCTTTTGC

Step 2:  T: CCTTCTGCTACCTTTTGCGCGCGCGGAA
         P:    CCTTTTGC

Step 3:  T: CCTTCTGCTACCTTTTGCGCGCGCGGAA
         P:           CCTTTTGC
```

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*)

“Bad character rule”

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

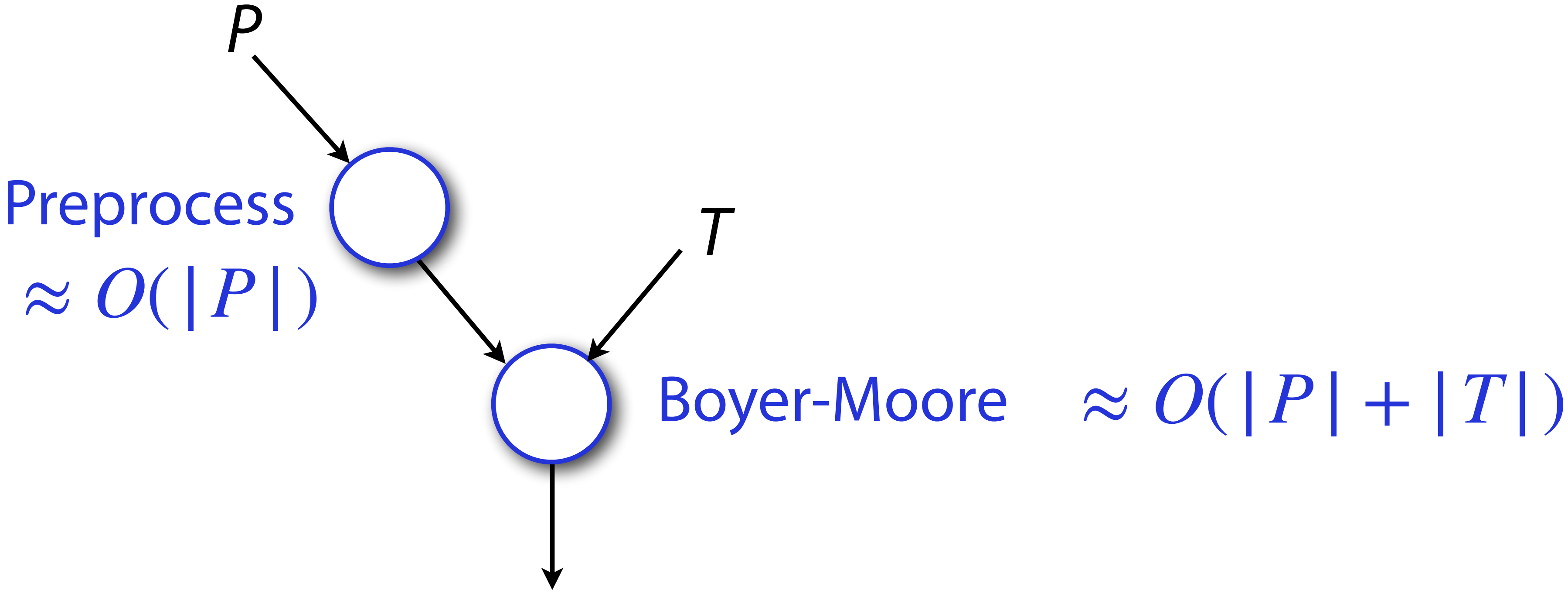
“Right-to-left scanning”

How do we put the first two rules in practice?

Boyer, RS and Moore, JS. "A fast string searching algorithm." *Communications of the ACM* 20.10 (1977): 762-772.

Exact pattern matching w/ Boyer-Moore

Boyer Moore **preprocesses** the pattern



P

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: BC rule preprocessing

Preprocessing requires two args: *P:* T C G C *Σ:* A C G T

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

		P			
		T	C	G	C
Σ	A				
	C				
	G				
	T				

Boyer-Moore: BC rule preprocessing

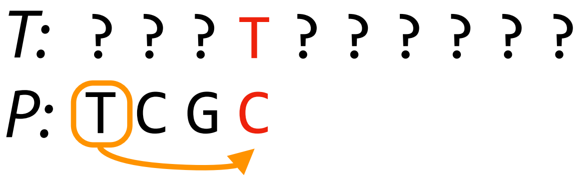
Preprocessing requires two args: *P:* T C G C *Σ:* A C G T

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

P

Σ

	T	C	G	C
A				
C				
G				
T				



T: ? ? ? T ? ? ? ? ? ?

P: T C G C

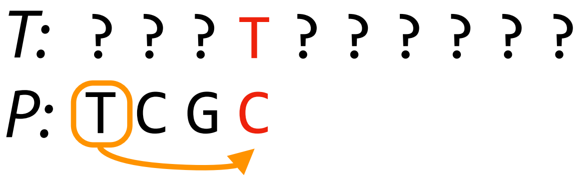
Boyer-Moore: BC rule preprocessing

Preprocessing requires two args: *P:* T C G C *Σ:* A C G T

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

P

		T	C	G	C
Σ	A				
	C				
	G				
	T				2



T: ? ? ? T ? ? ? ? ? ?
P: T C G C

Boyer-Moore: BC rule preprocessing

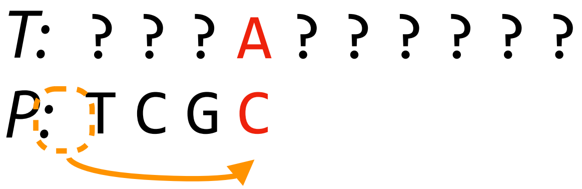
Preprocessing requires two args: *P:* T C G C *Σ:* A C G T

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

P

Σ

	T	C	G	C
A				
C				
G				
T				2



T: ? ? ? A ? ? ? ? ? ?

P: T C G C

Boyer-Moore: BC rule preprocessing

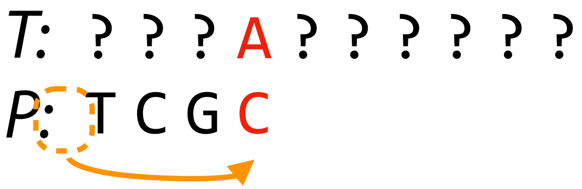
Preprocessing requires two args: *P*: T C G C *Σ*: A C G T

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

P

Σ

	T	C	G	C
A				3
C				
G				
T				2



T: ? ? ? A ? ? ? ? ? ?

P: T C G C

Boyer-Moore: BC rule preprocessing

Preprocessing requires two args: *P*: T C G C *Σ*: A C G T

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

P

Σ

	T	C	G	C
A	0	1	2	3
C	0	-	0	-
G	0	1	-	0
T	-	0	1	2

T: ? ? A ? ? ? ? ? ? ?
P: T C G C

T: ? ? C ? ? ? ? ? ?
P: T C G C

T: ? ? G ? ? ? ? ? ?
P: T C G C

T: ? ? T ? ? ? ? ? ?
P: T C G C

Boyer-Moore: BC rule preprocessing

Preprocessing requires two args: *P:* B A B A A A B *Σ:* A B

Pattern

		B	A	B	A	A	A	B
Σ	A							
	B							

Boyer-Moore: BC rule preprocessing

Preprocessing requires two args: *P:* B A B A A A B *Σ:* A B

For each character *p* in pattern *P*

For each character *c* in alphabet *Σ*

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

Pattern

		B	A	B	A	A	A	B
Σ	A	0	1					
	B	0	0					

Boyer-Moore: BC rule preprocessing

Preprocessing requires two args: *P:* B A B A A A B *Σ:* A B

For each character p in pattern *P*

For each character c in alphabet *Σ*

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

Pattern

		B	A	B	A	A	A	B
Σ	A	0	1	0	1			
	B	0	0	1	0			

Boyer-Moore: BC rule preprocessing

Preprocessing requires two args: *P*: B A B A A A B *Σ*: A B

For each character p in pattern *P*

For each character c in alphabet *Σ*

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

Pattern

		B	A	B	A	A	A	B
Σ	A	0	1	0	1	0	0	0
	B	0	0	1	0	1	2	3

Assignment 4: a_bmoore

Learning Objective:

Implement preprocessing of patterns with Boyer-Moore*

Observe Boyer-Moore* efficiency *as a heuristic*

Due: February 21th 11:59 PM

Consider: Optimal preprocessing is $\theta(|P||\Sigma|)$. Can you code it?

Boyer-Moore: Using the BC Table

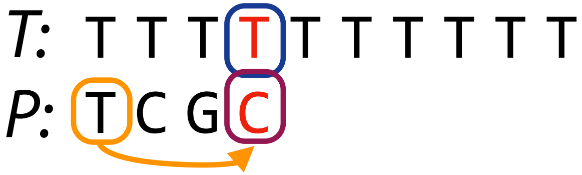
Try alignments from left-to-right and match characters from right-to-left

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

P

Σ

	T	C	G	C
A	0	1	2	3
C	0	-	0	-
G	0	1	-	0
T	-	0	1	2



T: T T T T T T T T T T

P: T C G C

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

		P			
		T	C	G	C
Σ	A	0	1	2	3
	C	0	-	0	-
	G	0	1	-	0
	T	-	0	1	2

T: G G G G G G G G G G
P: T C G C

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

P

Σ

	T	C	G	C
A	0	1	2	3
C	0	-	0	-
G	0	1	-	0
T	-	0	1	2

```
T: A A T C A A T A G C
P:     T C G C
```

Boyer-Moore: Tracking total skips

P

		A	A
Σ	A	0	0
	B	0	1

T: B B B B

T: B B B B B

T: B B B B B B

Boyer-Moore: Tracking total skips

P

		A	A	A
Σ	A	0	0	0
	B	0	1	2

T: B B B B

Assignment 4: a_bmoore

Learning Objective:

Implement preprocessing of patterns with Boyer-Moore*

Observe Boyer-Moore* efficiency *as a heuristic*

Due: February 21th 11:59 PM

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

But is it slower in practice? What is our total character comparisons?