String Algorithms and Data Structures

Z-values and the Z-algorithm

CS 199-225

Brad Solomon

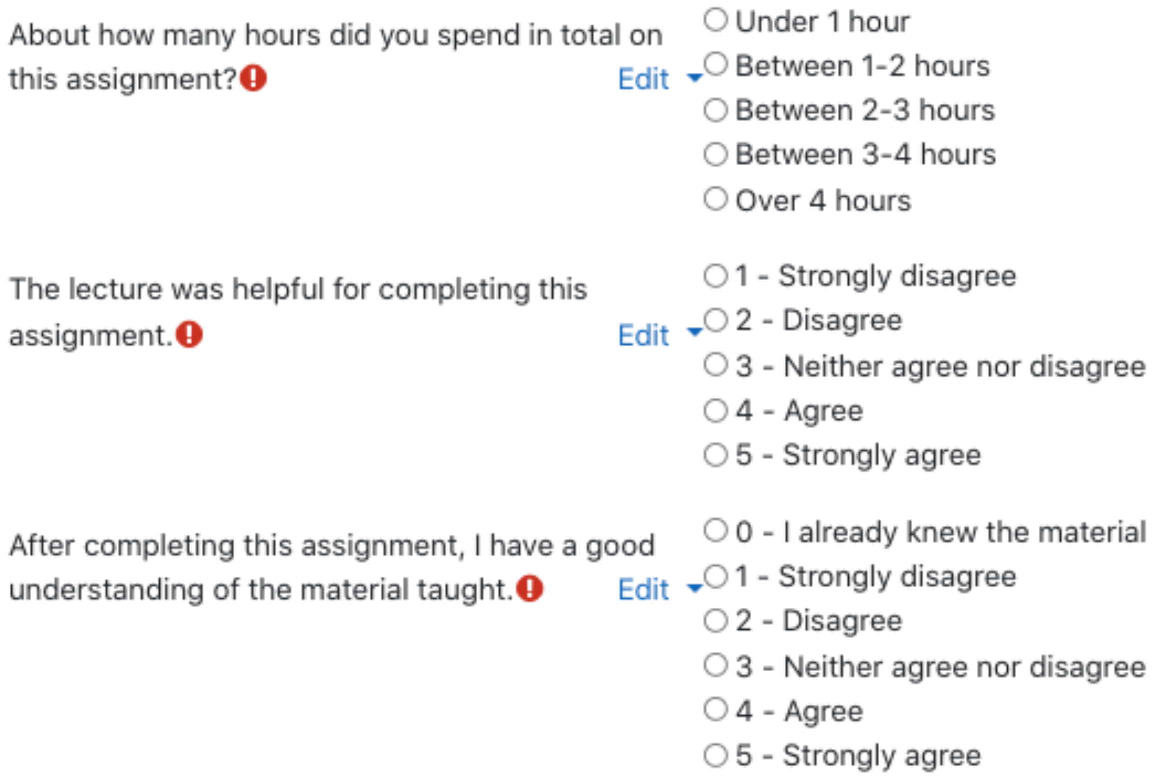
January 31, 2022

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Assignment 1: a_naive due today!

Don’t forget to submit feedback on Moodle



About how many hours did you spend in total on this assignment?

○ Under 1 hour
○ Between 1-2 hours
○ Between 2-3 hours
○ Between 3-4 hours
○ Over 4 hours

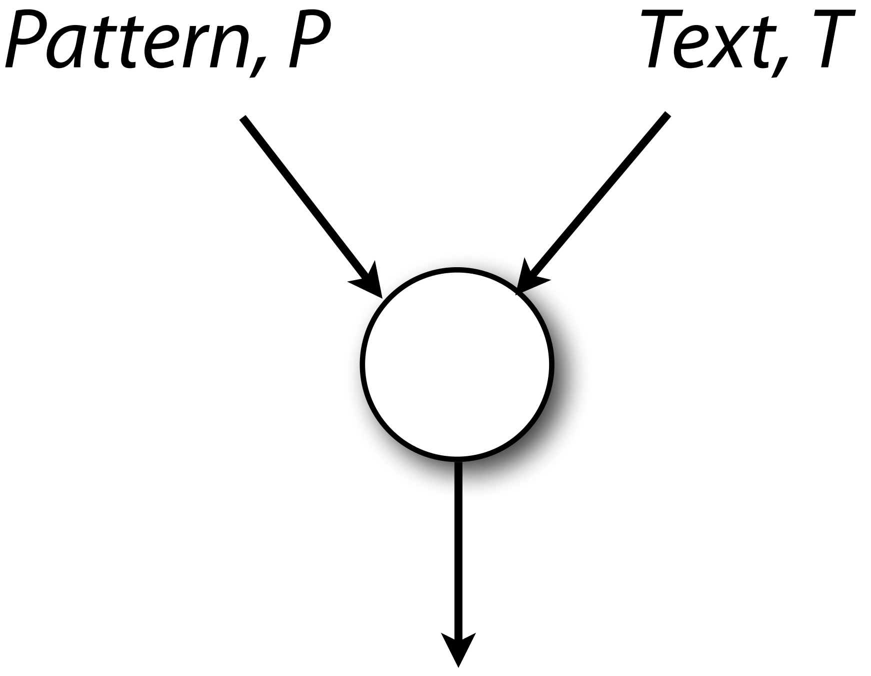
The lecture was helpful for completing this assignment.

○ 1 - Strongly disagree
○ 2 - Disagree
○ 3 - Neither agree nor disagree
○ 4 - Agree
○ 5 - Strongly agree

After completing this assignment, I have a good understanding of the material taught.

○ 0 - I already knew the material
○ 1 - Strongly disagree
○ 2 - Disagree
○ 3 - Neither agree nor disagree
○ 4 - Agree
○ 5 - Strongly agree

Exact Pattern Matching



Pattern, P

Text, T

Find instances of *P* in *T*

‘instances’: An exact, full length copy

Exact Pattern Matching

What's a simple algorithm for exact matching?

```
P: word
T: There would have been a time for such a word
   word word word word word word word word word
    word word word word word word word word
     word word word word word word word word
      word word word word word word word word
       word word word word word word word word
```

One occurrence

Try all possible alignments. For each, check if it matches. This is the *naïve algorithm*.

Exact Pattern Matching

What is good about the naive solution?

What is bad?

Exact Pattern Matching

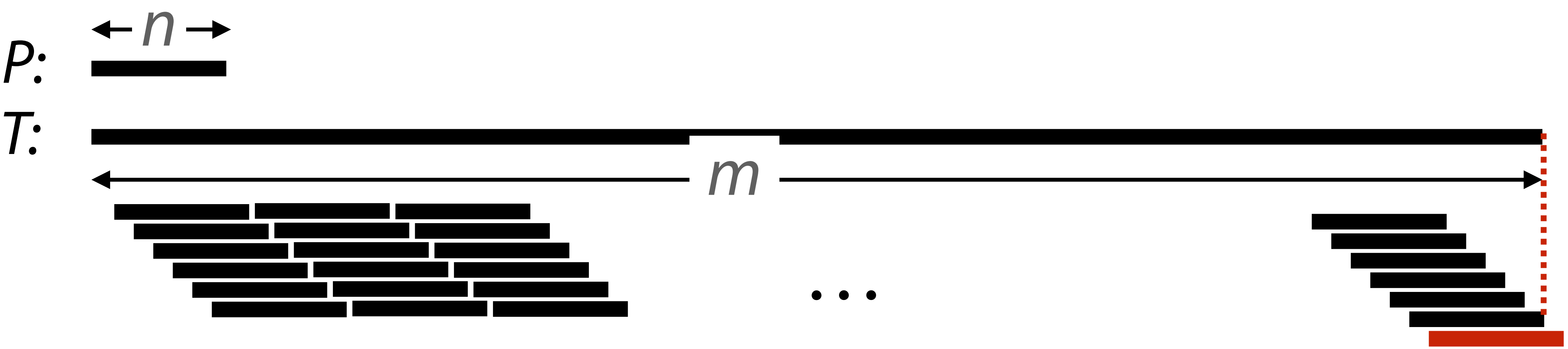
What is our time complexity? $(n = |P|, \quad m = |T|)$

(# of alignments) x (cost of an alignment)

Exact Pattern Matching

What is our time complexity? $(n = |P|, \quad m = |T|)$

(# of alignments) x (cost of an alignment)



P can fit at each `position' along T except the edge

Exact Pattern Matching

What is our time complexity? $(n = |P|, \quad m = |T|)$

(________________) x (cost of an alignment)

P: **aaaa**

T: **aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa**

aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa

aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa **aaaa**

aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa **aaaa**

aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa **aaaa**

aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa

There are ________ positions which extend past the edge of T

Exact Pattern Matching

What is our time complexity? $(n = |P|, \quad m = |T|)$

(m-n+1) x (cost of an alignment)

P: **aaaa**

T: **aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa**

aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa

Each alignment compares __________ characters.

Exact Pattern Matching

What is our time complexity? $(n = |P|, \quad m = |T|)$

$$\theta\Big((m - n + 1) \times n\Big)$$

String Algorithms in Genomics

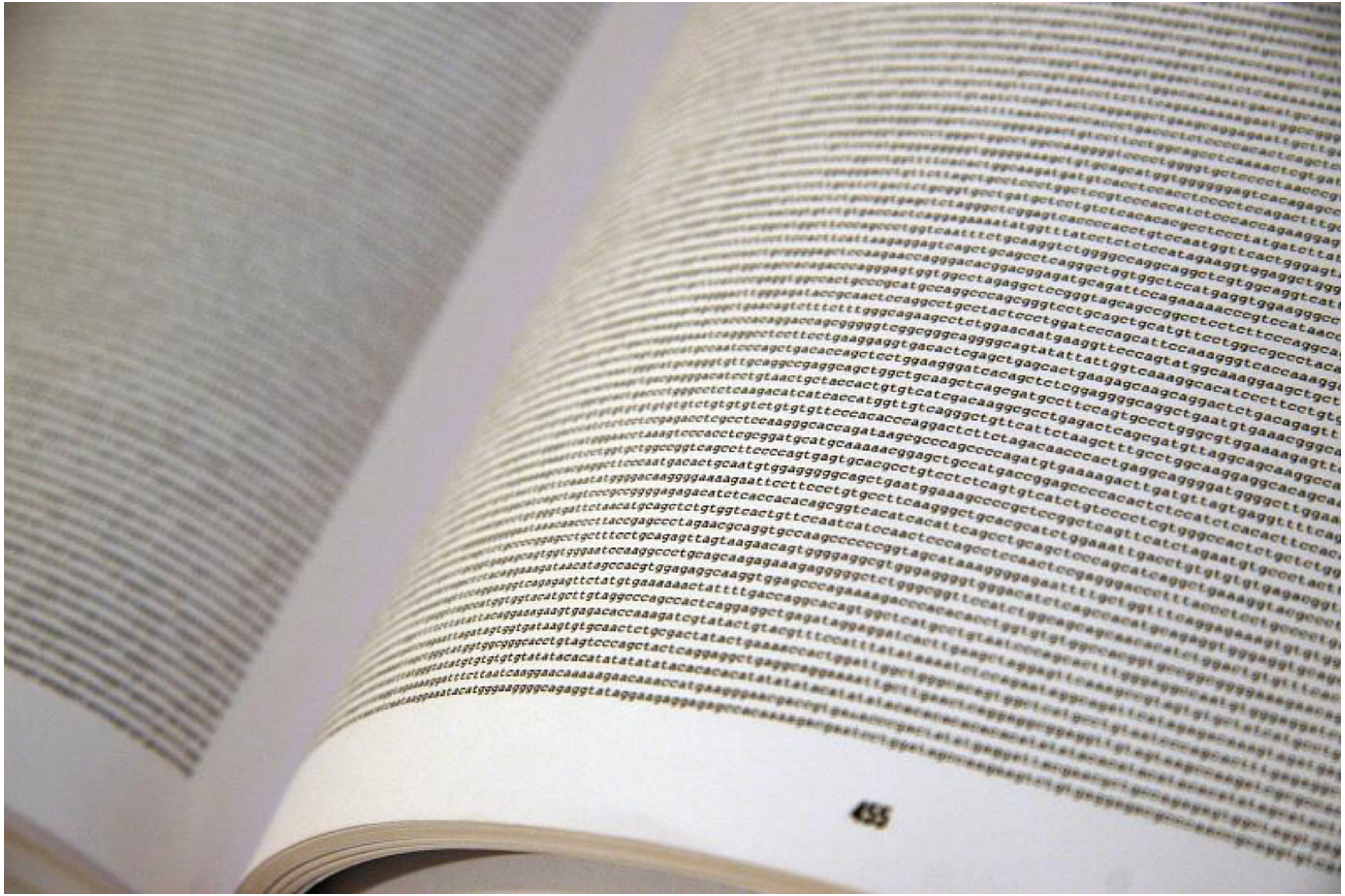
P: Read (n = ~50-150)

```
CTCAAACTCCTGACCTTTGGTGATCCACCCGCCTAGGCCTTC
```

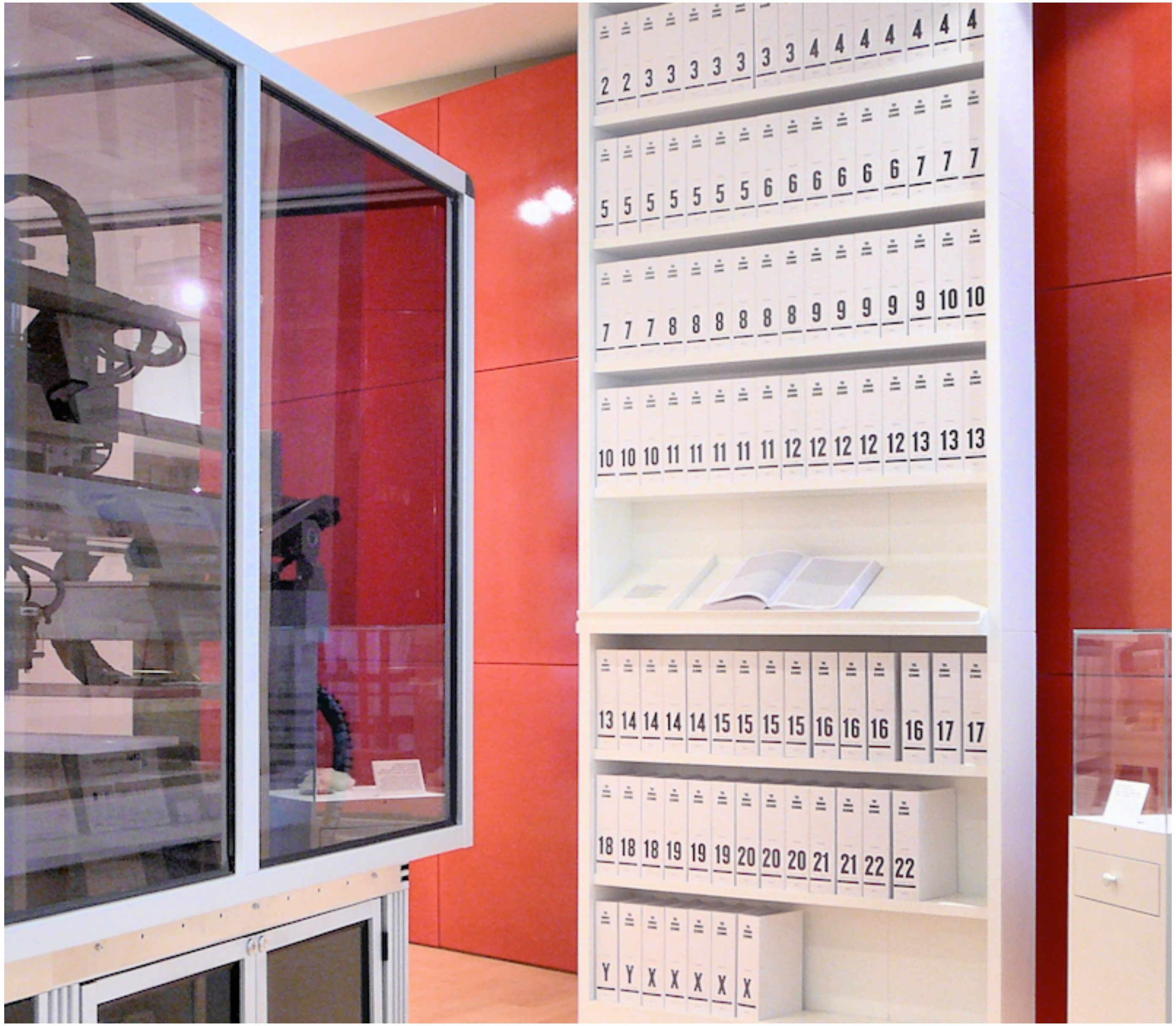
T: Reference (m = ~3 billion)

```
GATCACAGGTCTATCACCCTATTAACCACTCACGGGAGCTCTCCATGCATTTGGTATTTT
CGTCTGGGGGGTATGCACGCGATAGCATTGCGAGACGCTGGAGCCGGAGCACCCTATGTC
GCAGTATCTGTCTTTGATTCCTGCCTCATCCTATTATTTATCGCACCTACGTTCAATATT
ACAGGCGAACATACTTACTAAAGTGTGTTAATTAATTAATGCTTGTAGGACATAATAATA
ACAATTGAATGTCTGCACAGCCACTTTCCACACAGACATCATAACAAAAAATTTCCACCA
AACCCCCCCTCCCCCGCTTCTGGCCACAGCACTTAAACACATCTCTGCCAAACCCCAAAA
ACAAAGAACCCTAACACCAGCCTAACCAGATTTCAAATTTTATCTTTTGGCGGTATGCAC
TTTTAACAGTCACCCCCCAACTAACACATTATTTTCCCCTCCCACTCCCATACTACTAAT
CTCATCAATACAACCCCCGCCCATCCTACCCAGCACACACACACCGCTGCTAACCCCATA
CCCCGAACCAACCAAACCCCAAAGACACCCCCCACAGTTTATGTAGCTTACCTCCTCAAA
GCAATACACTGACCCGCTCAAACTCCTGGATTTTGGATCCACCCAGCGCCTTGGCCTAAA
CTAGCCTTTCTATTAGCTCTTAGTAAGATTACACATGCAAGCATCCCCGTTCCAGTGAGT
TCACCCTCTAAATCACCACGATCAAAAGGAACAAGCATCAAGCACGCAGCAATGCAGCTC
AAAACGCTTAGCCTAGCCACACCCCCACGGGAAACAGCAGTGATTAACCTTTAGCAATAA
ACGAAAGTTTAACTAAGCTATACTAACCCCAGGGTTGGTCAATTTCGTGCCAGCCACCGC
GGTCACACGATTAACCCAAGTCAATAGAAGCCGGCGTAAAGAGTGTTTTAGATCACCCCC
TCCCCAATAAAGCTAAAACTCACCTGAGTTGTAAAAAACTCCAGTTGACACAAAATAGAC
TACGAAAGTGGCTTTAACATATCTGAACACACAATAGCTAAGACCCAAACTGGGATTAGA
TACCCCACTATGCTTAGCCCTAAACCTCAACAGTTAAATCAACAAAACTGCTCGCCAGAA
CACTACGAGCCACAGCTTAAAACTCAAAGGACCTGGCGGTGCTTCATATCCCTCTAGAGG
AGCCTGTTCTGTAATCGATAAACCCCGATCAACCTCACCACCTCTTGCTCAGCCTATATA
CCGCCATCTTCAGCAAACCCTGATGAAGGCTACAAAGTAAGCGCAAGTACCCACGTAAAG
ACGTTAGGTCAAGGTGTAGCCCATGAGGTGGCAAGAAATGGGCTACATTTTCTACCCCAG
AAAACTACGATAGCCCTTATGAAACTTAAGGGTCGAAGGTGGATTTAGCAGTAAACTAAG
AGTAGAGTGCTTAGTTGAACAGGGCCCTGAAGCGCGTACACACCGCCCGTCACCCTCCTC
AAGTATACTTCAAAGGACATTTAACTAAAACCCCTACGCATTTATATAGAGGAGACAAGT
CGTAACCTCAAACTCCTGCCTTTGGTGATCCACCCGCCTTGGCCTACCTGCATAATGAAG
AAGCACCCAACTTACACTTAGGAGATTTCAACTTAACTTGACCGCTCTGAGCTAAACCTA
```

String Algorithms in Genomics



String Algorithms in Genomics



Improving exact pattern matching

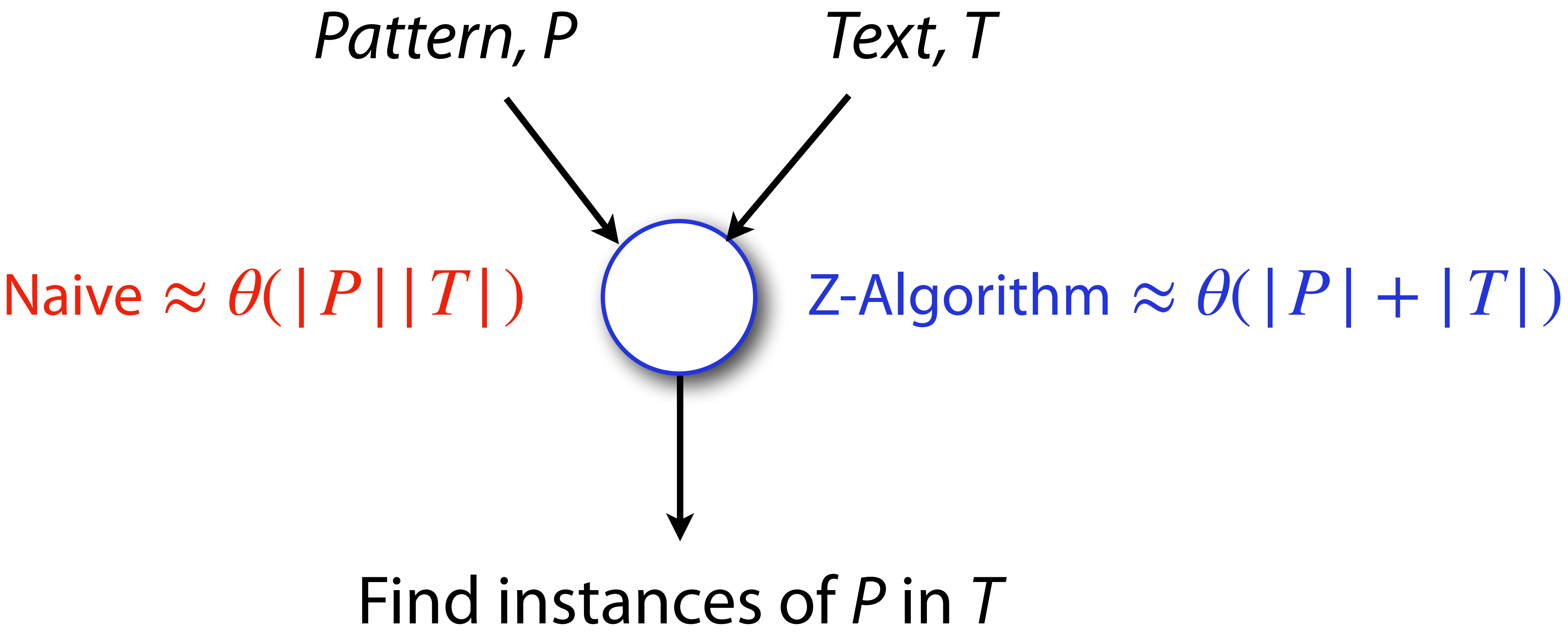
How can we do better than the naïve algorithm?

… If we have infinite space?

… If I tell you the pattern ahead of time?

… If I tell you the text ahead of time?

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

The Z-value [$Z_i(S)$]

Given a string *S*, $Z_i(S)$ is the length of the longest substring in *S*, starting at position *i* , that matches a prefix of *S*.

	0	1	2	3	4	5	6	7	8	9
S:	**T**	**T**	**C**	**G**	**T**	**T**	**A**	**G**	**C**	**G**

$Z_0(S) =$

$Z_1(S) =$

$Z_2(S) =$

$Z_3(S) =$

$Z_4(S) =$

$Z_5(S) =$

The Z-value [$Z_i(S)$]

Given a string *S*, $Z_i(S)$ is the length of the longest substring in *S*, starting at position *i* , that matches a prefix of *S*.

	0	1	2	3	4	5	6	7	8	9
S:	**T**	**T**	**C**	**G**	**T**	**T**	**A**	**G**	**C**	**G**

$Z_0(S) = 10$

$Z_1(S) = 1$

$Z_2(S) = 0$

$Z_3(S) =$

$Z_4(S) =$

$Z_5(S) =$

The Z-value [$Z_i(S)$]

Given a string S, $Z_i(S)$ is the length of the longest substring in S, starting at position $i > 0$, that matches a prefix of S.

	0	1	2	3	4	5	6	7	8	9
S:	**T**	**T**	**C**	**G**	**T**	**T**	**A**	**G**	**C**	**G**

$Z_0(S) = 10$

$Z_1(S) = 1$

$Z_2(S) = 0$

$Z_3(S) = 0$

$Z_4(S) = 2$

$Z_5(S) = 1$

Calculating the Z-values

Naive: Compute the Z-values by *explicitly* comparing characters (left-to-right scan):

$Z_1 =$

A A A A B A A C A A B A A ... **A A A A B A A C A A B A A ...**

$Z_5 =$

A A A A B A A C A A B A A ... **A A A A B A A C A A B A A ...**

What is our time complexity?

Calculating the Z-values

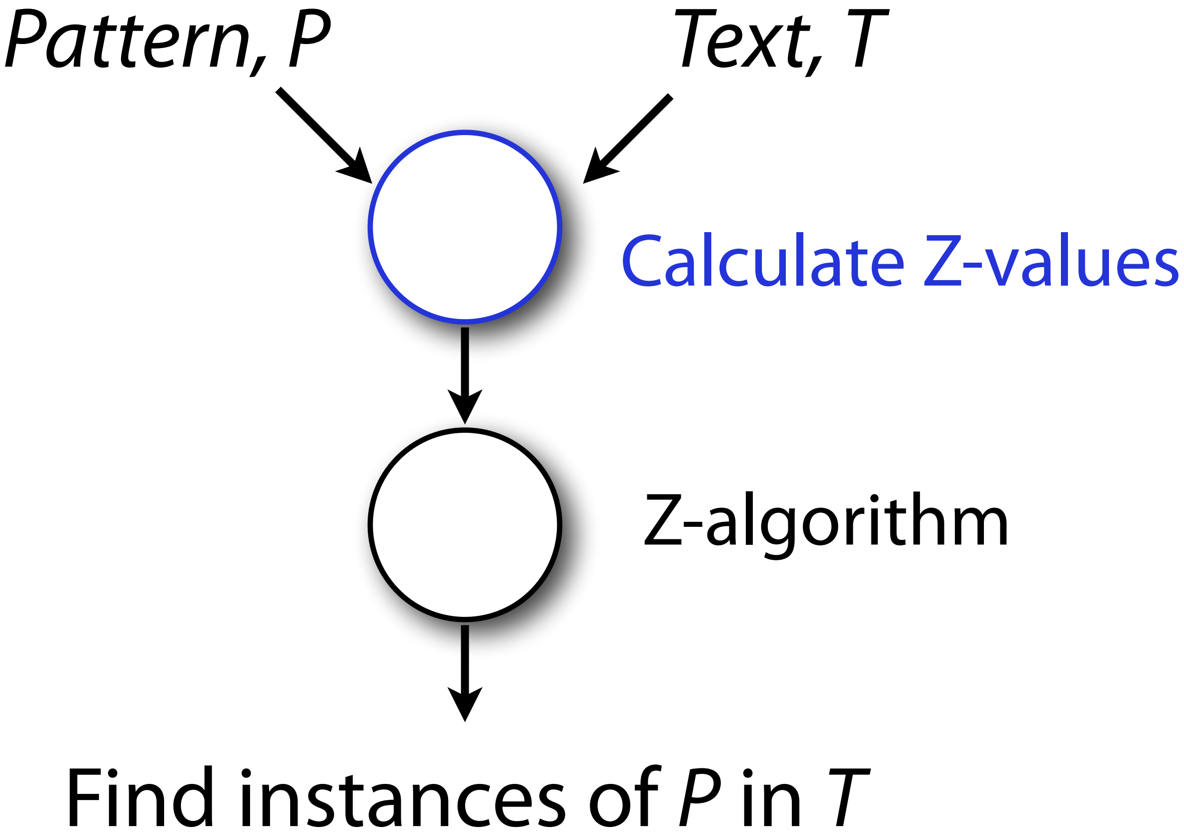
Naive: Compute the Z-values by *explicitly* comparing characters (left-to-right scan):

```
S: 1 1 0 1 1 0 0 1
   1 0 1 1 0 0 1
   0 1 1 0 0 1
   1 1 0 0 1
   1 0 0 1
   0 0 1
   0 1
   1
```

What is our time complexity?

Pattern matching with the Z-value

Given a Z_i value calculator, how do we solve pattern matching?



Pattern, P

Text, T

Calculate Z-values

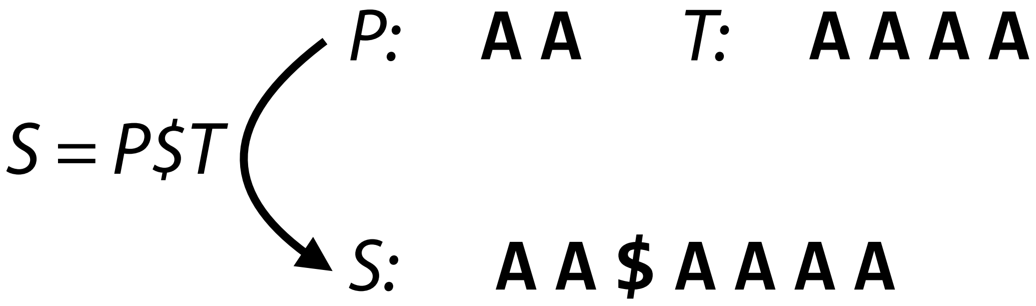
Z-algorithm

Find instances of *P* in *T*

Z-value Pattern Matching

To solve pattern matching (given *P* and *T*), let ***S* = *P$T***

$ = ‘terminal character’, outside alphabet



P: **A A** *T*: **A A A A**

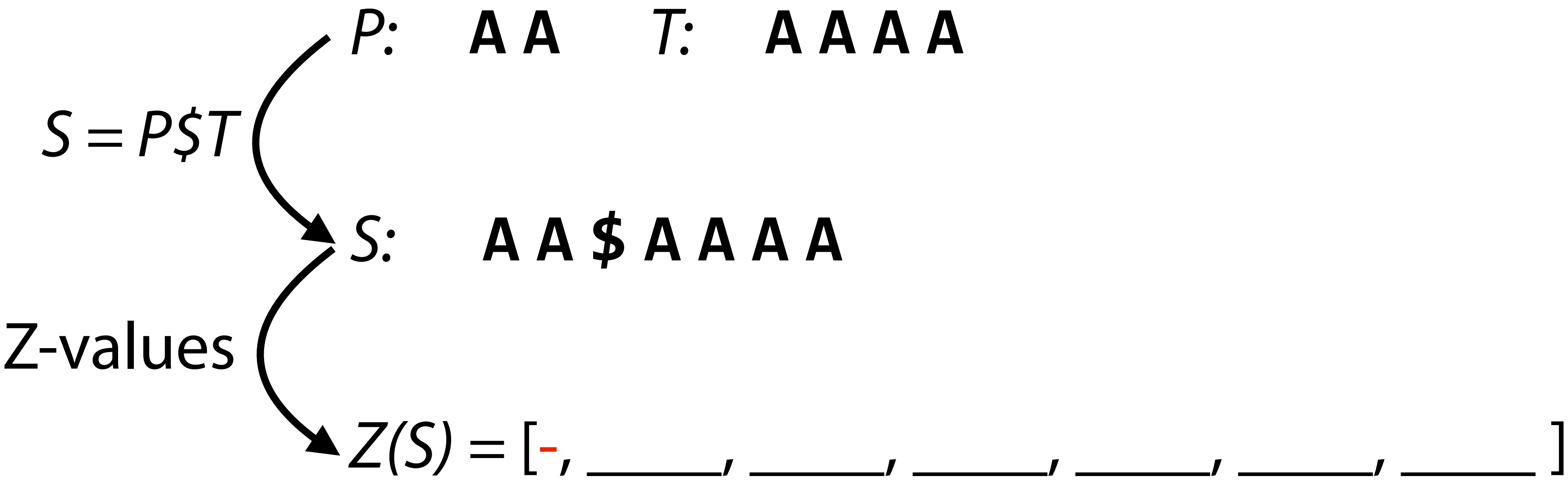
S = *P$T*

S: **A A $ A A A A**

Z-value Pattern Matching

To solve pattern matching (given *P* and *T*), let ***S* = *P$T***

$ = 'terminal character', outside alphabet



P: **A A** *T*: **A A A A**

S = *P$T*

S: **A A $ A A A A**

Z-values

Z(S) = [-, _____, _____, _____, _____, _____, _____]

Z-value Pattern Matching

To solve pattern matching (given *P* and *T*), let ***S* = *P$T***

$ = ‘terminal character’, outside alphabet

P: **A A** *T*: **A A A A**

	0	1	2	3	4	5	6
S:	**A**	**A**	**$**	**A**	**A**	**A**	**A**
				0	1	2	3

Z(*S*) = [-, 1, 0, 2, 2, 2, 1]

What Z_i values are matches?

What are the matching indices in *T*?

Z-value Pattern Matching

P: **T T** *T:* **C T T A**

S:

Z(S):

Z-value search pseudo-code

1. *Concatenate (S=P$T)*

2. *Calculate Z-values for S*

3. For i<0, match if Z_i = __________

   Match is ***not*** at i, but instead at

   ____

Assignment 2: a_zval

Learning Objective:

Construct a Z-value calculator and measure its efficiency

Demonstrate use of Z-values in pattern matching

Due: February 7th 11:59 PM

Consider: Our goal is $\theta(|P| + |T|)$. Does Z-value search match this?

End-of-class brainstorm

What information does a single Z-value tell us?

If I know $Z_{i-1}(S)$, can I use that information to help me compute $Z_i(S)$?

The Z-value (Take 2)

Given a string *S*, *$Z_i(S)$* is the length of the longest substring in *S*, starting at position *i* , that matches a prefix of *S*.

What information does this give us?

S: [redacted]

$Z_4(S) = 2$

0	1	2	3	4	5	6	7	8	9	…

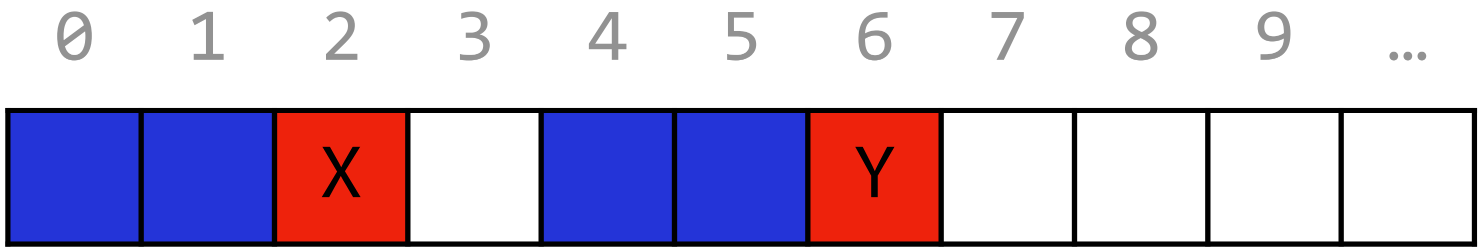
The Z-value (Take 2)

Given a string *S*, $Z_i(S)$ is the length of the longest substring in *S*, starting at position *i* , that matches a prefix of *S*.

What information does this give us?

S: TTCGTTAGCG

$$Z_4 = 2$$



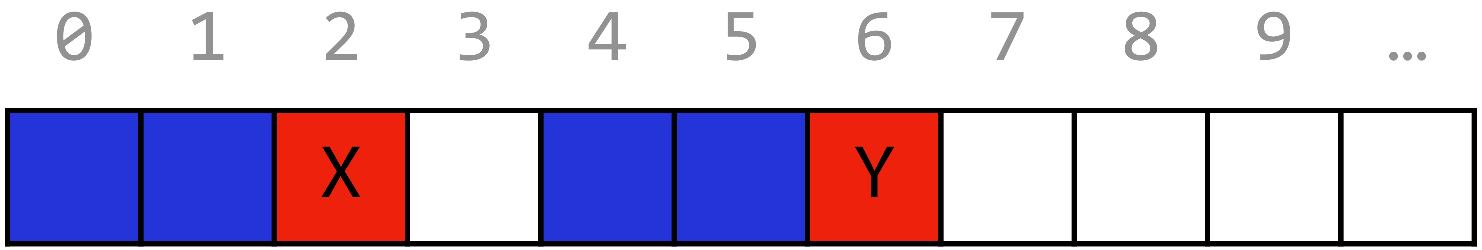
The Z-value (Take 2)

Given a string *S*, *Zᵢ(S)* is the length of the longest substring in *S*, starting at position *i* , that matches a prefix of *S*.

What information does this give us?

```
   0 1 2 3 4 5 6 7 8 9
S: T T C G T T A G C G
```

$Z_4 = 2$

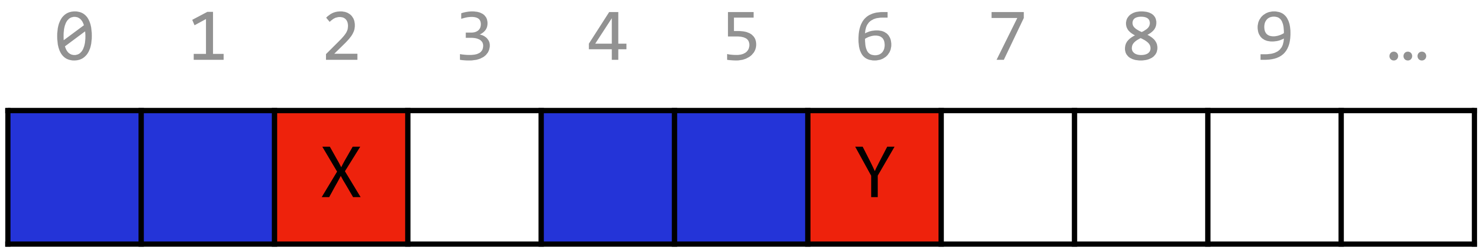


The Z-value (Take 2)

Given a string *S*, *Z*$Z_i(S)$ is the length of the longest substring in *S*, starting at position *i* , that matches a prefix of *S*.

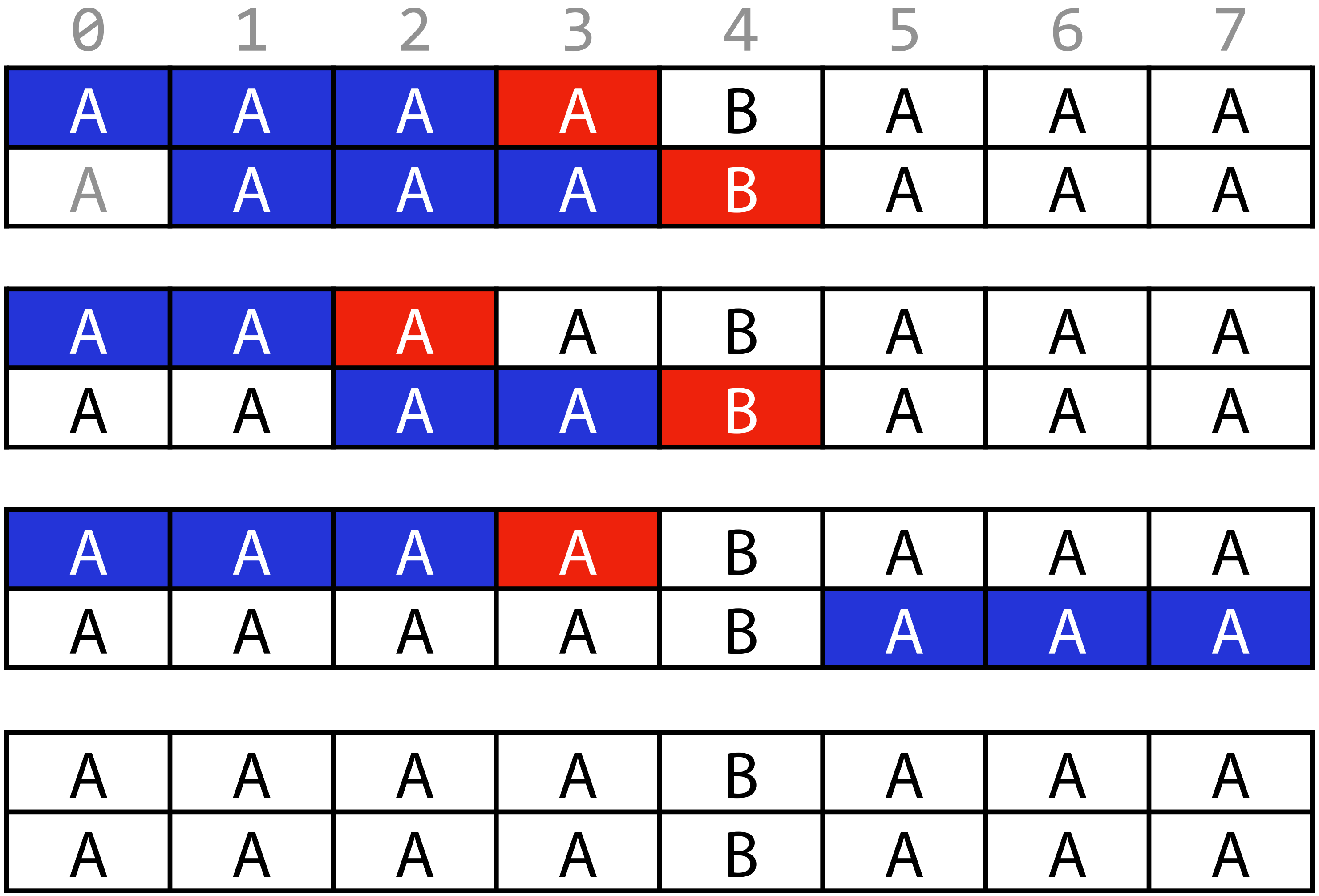
$Z_i \neq 0$ means that my *substring* (i, Z_i) matches my *prefix* $(0, Z_i)$

The characters after my substring and prefix ***must not match!***



Calculating the Z-values (Take 2)

Intuition: We can use the previous $Z_1, \ldots, Z_i$ to compute Z_{i+1}!



$Z_1 = 3$

0	1	2	3	4	5	6	7
A	A	A	A	B	A	A	A
A	A	A	A	B	A	A	A

$Z_2 = 2$

0	1	2	3	4	5	6	7
A	A	A	A	B	A	A	A
A	A	A	A	B	A	A	A

$Z_5 = 3$

0	1	2	3	4	5	6	7
A	A	A	A	B	A	A	A
A	A	A	A	B	A	A	A

$Z_6 = ?$

0	1	2	3	4	5	6	7
A	A	A	A	B	A	A	A
A	A	A	A	B	A	A	A

Calculating the Z-values (Take 2)

Intuition: We can use the previous $Z_1, \ldots, Z_i$ to compute Z_{i+1}!

The **Z-algorithm** (next week) will formalize this process.