String Algorithms and Data Structures
Z-values and the Z-algorithm

CS 199-225

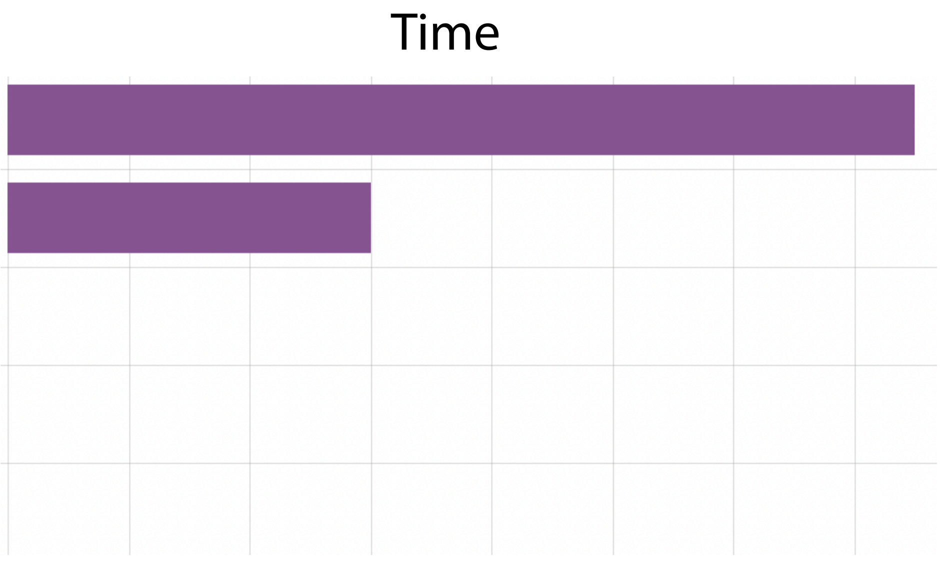
February 7, 2022

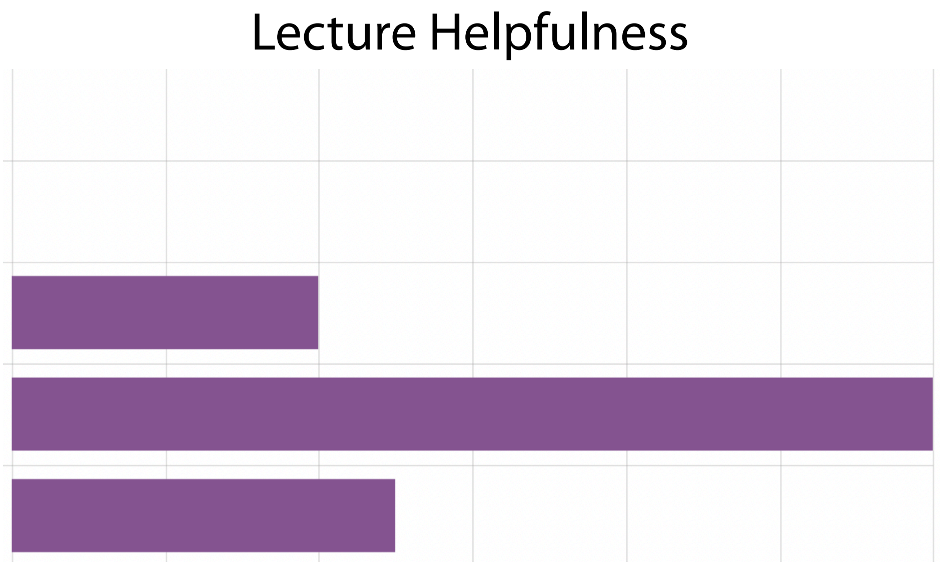
Brad Solomon

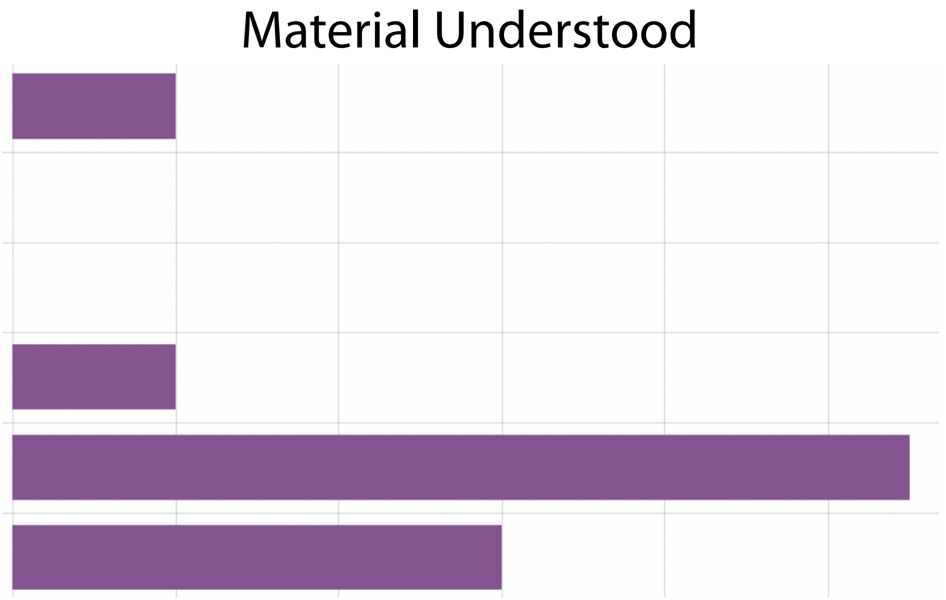
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

A_naive reflection







Optional task for faster algorithm

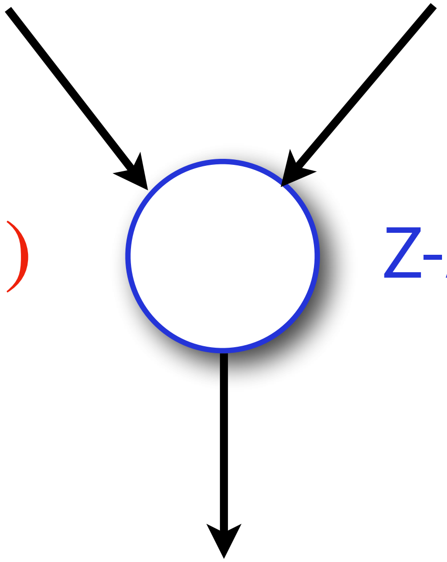
A_zval due today!

A_zalg will build off of a_zval (and include a second chance at search!)

Correct character counting is key (same as last week)

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 $i > 0$, that matches a prefix of *S*.

```
   0 1 2 3 4 5 6 7 8 9
S: A B C D A B C D A B
```

$Z_4(S) =$

```
S: C G C G A ? ? ? ? ?
```

$Z_5(S) = 3$

```
S: A ? ? ? ? ? ? ? ? ?
```

$Z_1(S) = 7$

Z-value Pattern Matching

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

S: **T T $ C T T A**

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

Z-value search pseudo-code

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

2. *Calculate Z-values for S*

3. For i<0, match if $\boldsymbol{Z_i} = \mathbf{|P|}$

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

   T [i - |P| - 1]

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 $i > 0$, 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!***

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

$$Z_4 = 2$$

0	1	2	3	4	5	6	7	8	9	…
		X				Y				

The Z-Algorithm

Assume we've computed $Z_0, \ldots, Z_{i-1}$ and need to calculate Z_i

Case 1: We know nothing about the characters at S[i]

$Z_1 = ?$

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

Case 2: We know something about the characters at S[i]

$Z_2 = ?$

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

The Z-Algorithm

$Z_1 = 3$

$Z_2 = ?$

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

We track our current knowledge of *S* using three values: i, r, l

i, the current index position being calculated

r, the index of the rightmost character which has ever been matched

l, the index of Z-value which r belongs too

The Z-Algorithm

Start

End

i, the current index =

r, the furthest match char =

l, the furthest reaching Z-value =

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

The Z-Algorithm

Start End

i, the current index =

r, the furthest match char =

l, the furthest reaching Z-value =

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

The Z-Algorithm

Start

End

i, the current index =

r, the furthest match char =

l, the furthest reaching Z-value =

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

The Z-Algorithm

Start

End

i, the current index =

r, the furthest match char =

l, the furthest reaching Z-value =

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

The Z-Algorithm

Start

End

i, the current index =

r, the furthest match char =

l, the furthest reaching Z-value =

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

The Z-Algorithm

Start | End

i, the current index =

r, the furthest match char =

l, the furthest reaching Z-value =

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

The Z-Algorithm

Start End

i, the current index =

r, the furthest match char =

l, the furthest reaching Z-value =

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

The Z-Algorithm

$Z_2 = 2$

$Z_3 = ?$

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

We track our current knowledge of *S* using three values: i, r, l

i gets updated every iteration (as we compute Z_i)

r gets updated when $Z_i > 0$ *AND* $r_{new} > r_{old}$

l gets updated whenever r is updated (it stores the index of r's Z-value)

The Z-Algorithm

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

The values of i, r, l tell us how much work we need to do to compute Z_i

Case 1: $i > r$

Ex: $i = 1, r = 0, l = 0$

We must compute Z_i explicitly!

The Z-Algorithm

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

The values of i, r, l tell us how much work we need to do to compute Z_i

Case 1: $i > r$

Ex: $i = 5, r = 2, l = 1$

We must compute Z_i explicitly!

The Z-Algorithm

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

The values of i, r, l tell us how much work we need to do to compute Z_i

Case 2: $i \leq r$

Ex: $i = 6, r = 7, l = 5$

To find Z_6, we can save time by looking up the value ______________

The Z-Algorithm

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

The values of i, r, l tell us how much work we need to do to compute Z_i

Case 2: $i \leq r$

Ex: $i = 6, r = 7, l = 5$

To find Z_6, we can save time by looking up the value ______________

The Z-Algorithm

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

The values of i, r, l tell us how much work we need to do to compute Z_i

Case 2: $i \leq r$

Ex: $i = 4, r = 4, l = 3$

To find Z_4, we can save time by looking up the value ______________

The Z-Algorithm

Let $l = 0, r = 0$, for $i = [1, \ldots, |S| - 1]$:

Compute Z_i using irl:

Case 1 ($i > r$): Compute explicitly; update irl

Case 2 ($i \leq r$):

Use previous Z-values to avoid work

Explicitly compute only *'new'* characters

How can we tell the difference between cases?

The Z-Algorithm

$i = 5, r = 7, l = 4$

0	1	2	3	4	**5**	6	7
A	B	C	D	A	B	C	D
A	B	C	D	A	B	C	D

Let β be the characters from i to r (inclusive).

What is $|\beta|$ in terms of i, r, l?

Let k be the Z-value index we want to look up.

What is k in terms of i, r, l?

The Z-Algorithm

$i = 5, r = 7, l = 4$

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

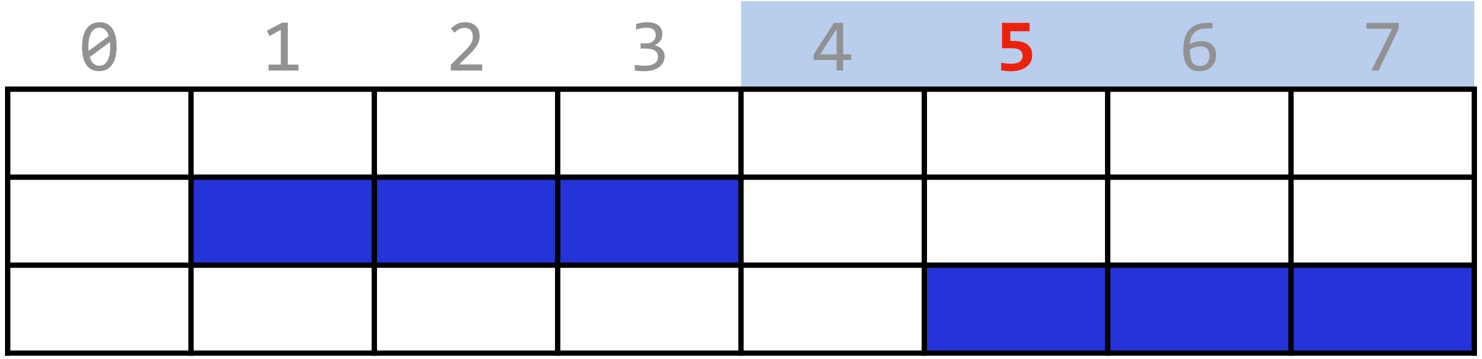
Case 2a: $i \leq r, Z_k < |\beta|$

$|\beta| =$ __________, $k =$ __________, $Z_k =$ __________

$Z_i =$ __________

The Z-Algorithm

$i = 5, r = 7, l = 4$

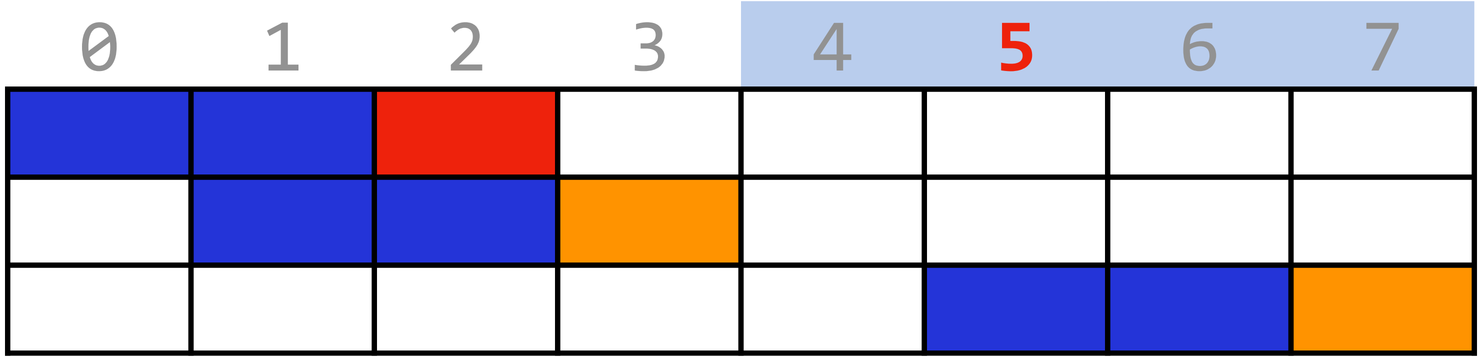


Case 2a: $i \leq r, Z_k < |\beta|$

Z_l tells us that β matches earlier.

The Z-Algorithm

$i = 5, r = 7, l = 4$

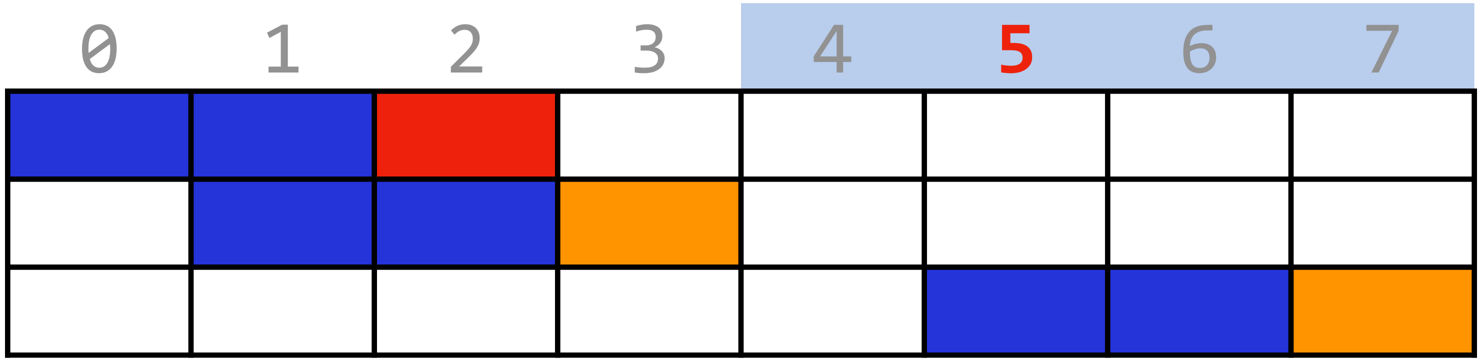


Case 2a: $i \leq r, Z_k < |\beta|$

Z_l tells us that β matches earlier. Z_k tells us how much matches the prefix.

The Z-Algorithm

$i = 5, r = 7, l = 4$



Case 2a: $i \leq r, Z_k < |\beta|$

Z_l tells us that β matches earlier. Z_k tells us how much matches the prefix.

Because $Z_k < |\beta|, Z_i =$ ___________

The Z-Algorithm

$i = 4, r = 4, l = 3$

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

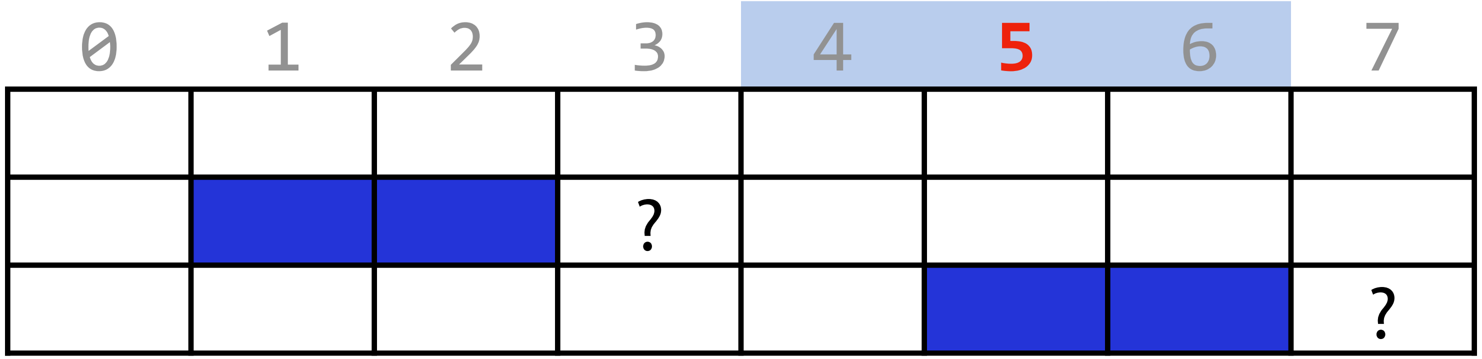
Case 2b: $i \leq r, Z_k = |\beta|$

$|\beta| =$ __________, $k =$ __________, $Z_k =$ __________

$Z_i =$ __________

The Z-Algorithm

$i = 5, r = 6, l = 4$

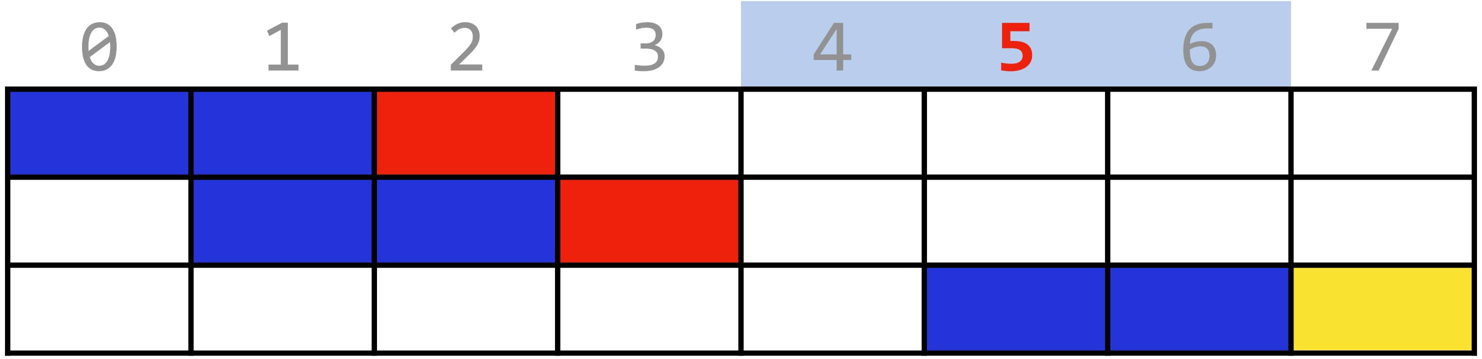


Case 2b: $i \leq r$, $Z_k = |\beta|$

Z_l tells us that β matches earlier.

The Z-Algorithm

$i = 5, r = 6, l = 4$



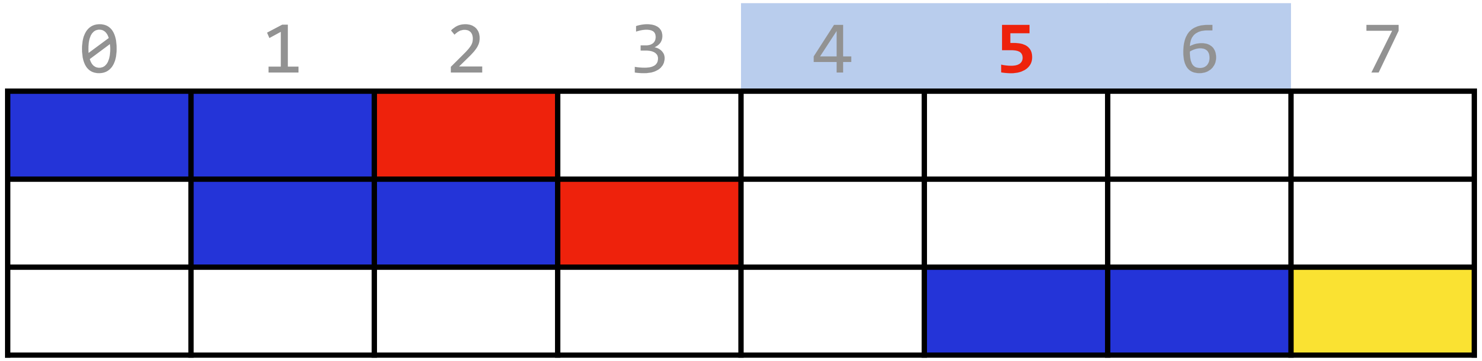
Case 2b: $i \leq r$, $Z_k = |\beta|$

Z_l tells us that β matches earlier.

Z_k tells us how much matches the prefix… but not everything!

The Z-Algorithm

$i = 5, r = 6, l = 4$



Case 2b: $i \leq r, Z_k = |\beta|$

We have all the same info as before but we have unseen characters!

Because $Z_k = |\beta|, Z_i =$ ___________

The Z-Algorithm

$i = 3, r = 5, l = 1$

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

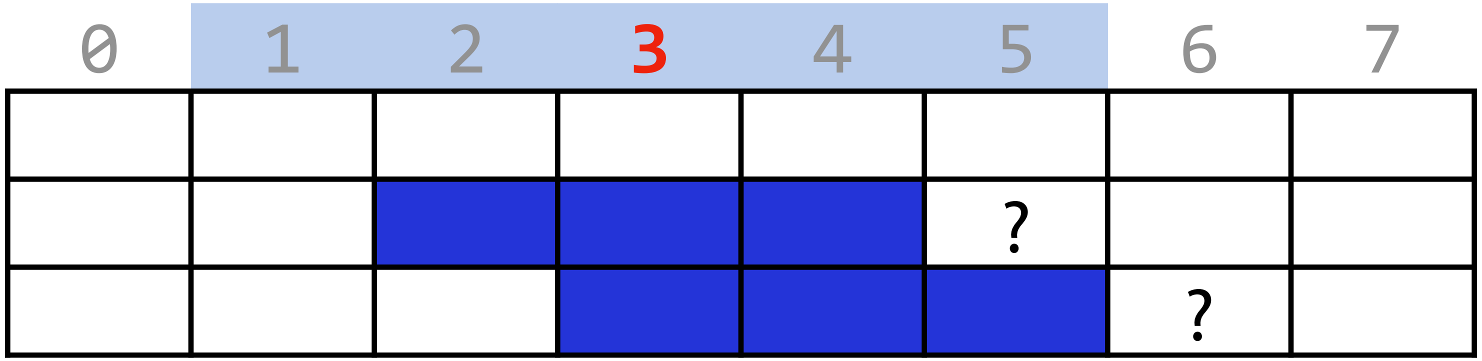
Case 2c: $i \leq r, Z_k > |\beta|$

$|\beta| =$ __________, $k =$ __________, $Z_k =$ __________

$Z_i =$ __________

The Z-Algorithm

$i = 3, r = 5, l = 1$

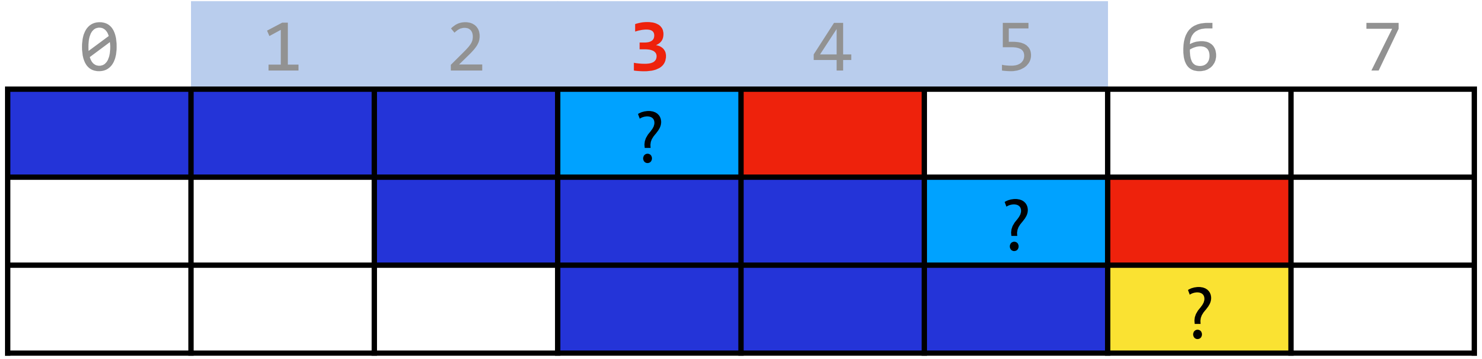


Case 2c: $i \leq r, Z_k > |\beta|$

Z_l tells us that β matches earlier.

The Z-Algorithm

$i = 3, r = 5, l = 1$



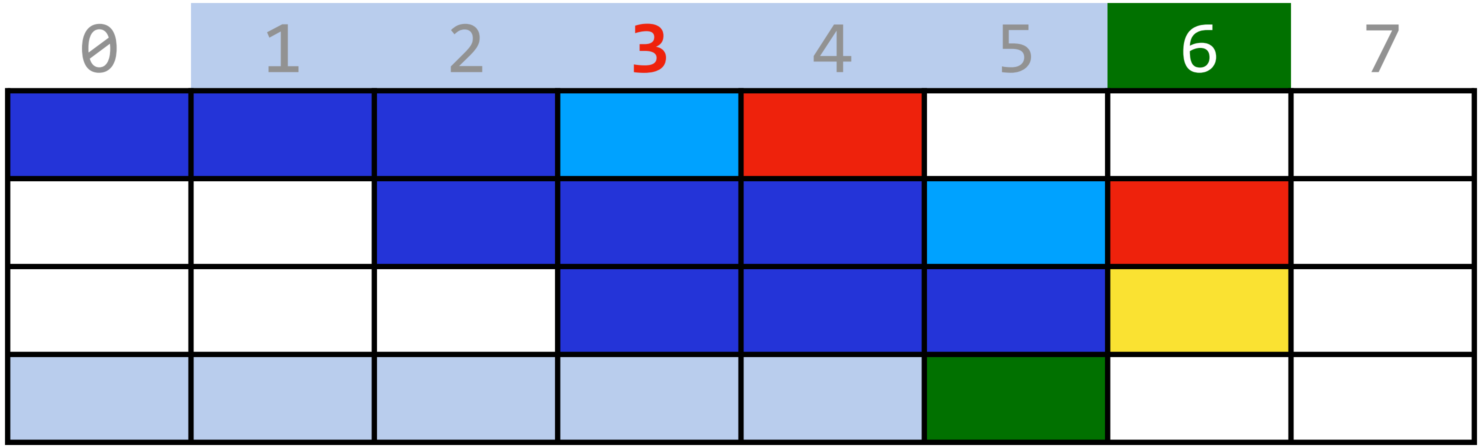
Case 2c: $i \leq r$, $Z_k > |\beta|$

Z_l tells us that β matches earlier. Z_k tells us how much matches the prefix.

What do we know about yellow?

The Z-Algorithm

$i = 3, r = 5, l = 1$



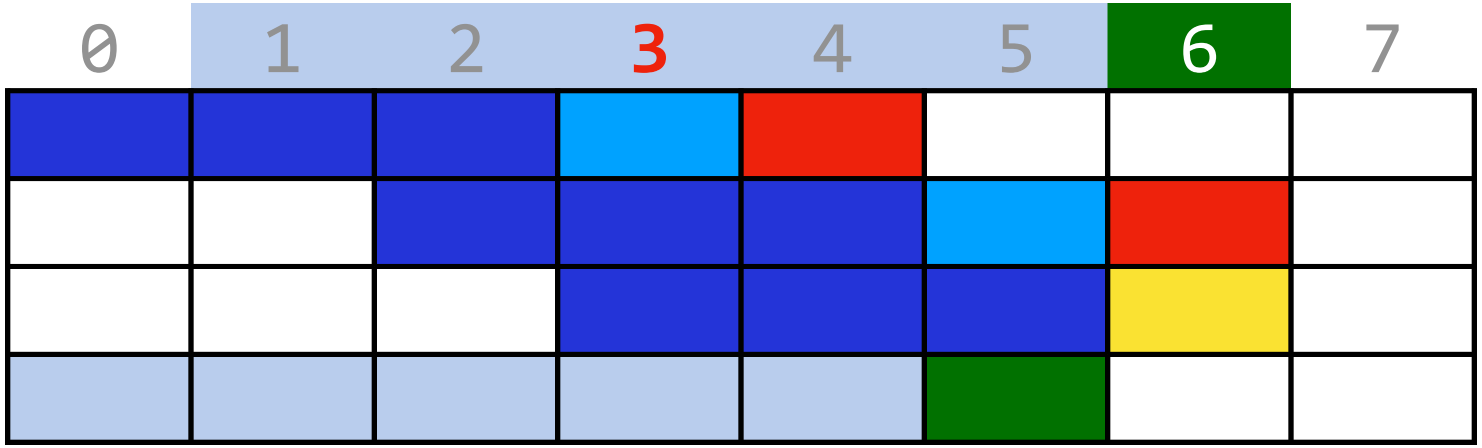
Case 2c: $i \leq r$, $Z_k > |\beta|$

Z_l tells us that β matches earlier. Z_k tells us how much matches the prefix.

Z_l also tells us that yellow and green can't be equal!

The Z-Algorithm

$i = 3, r = 5, l = 1$



Case 2c: $i \leq r$, $Z_k > |\beta|$

Z_l tells us that β is our prefix. Z_k is also a previously computed prefix.

Because $Z_k > |\beta|$, $Z_i =$ __________

The Z-Algorithm

Let $l = 0, r = 0$, for $i = [1, \ldots, |S| - 1]$:

Compute Z_i using irl:

Case 1 ($i > r$): Compute explicitly; update irl

Case 2 ($i \leq r$):

2a: ($Z_k < |\beta|$): $Z_i = Z_k$

2b: ($Z_k = |\beta|$): $Z_i = Z_k$+ *explicit*(r+1); update irl

2c: ($Z_k > |\beta|$): $Z_i = |\beta|$

Assignment 3: a_zalg

Learning Objective:

Construct the full Z-algorithm and measure its efficiency

Demonstrate use of Z-algorithm in pattern matching

Due: February 14th 11:59 PM

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

Next week:

If I gave you the pattern I was interested in ahead of time, what could you pre-compute to speed up search?

Ex: I’m going to try to look up the word **‘arrays’** — but you don’t know what text I’m going to search through.