

String Algorithms and Data Structures

The Z-algorithm

CS 199-225

Brad Solomon

February 13, 2023

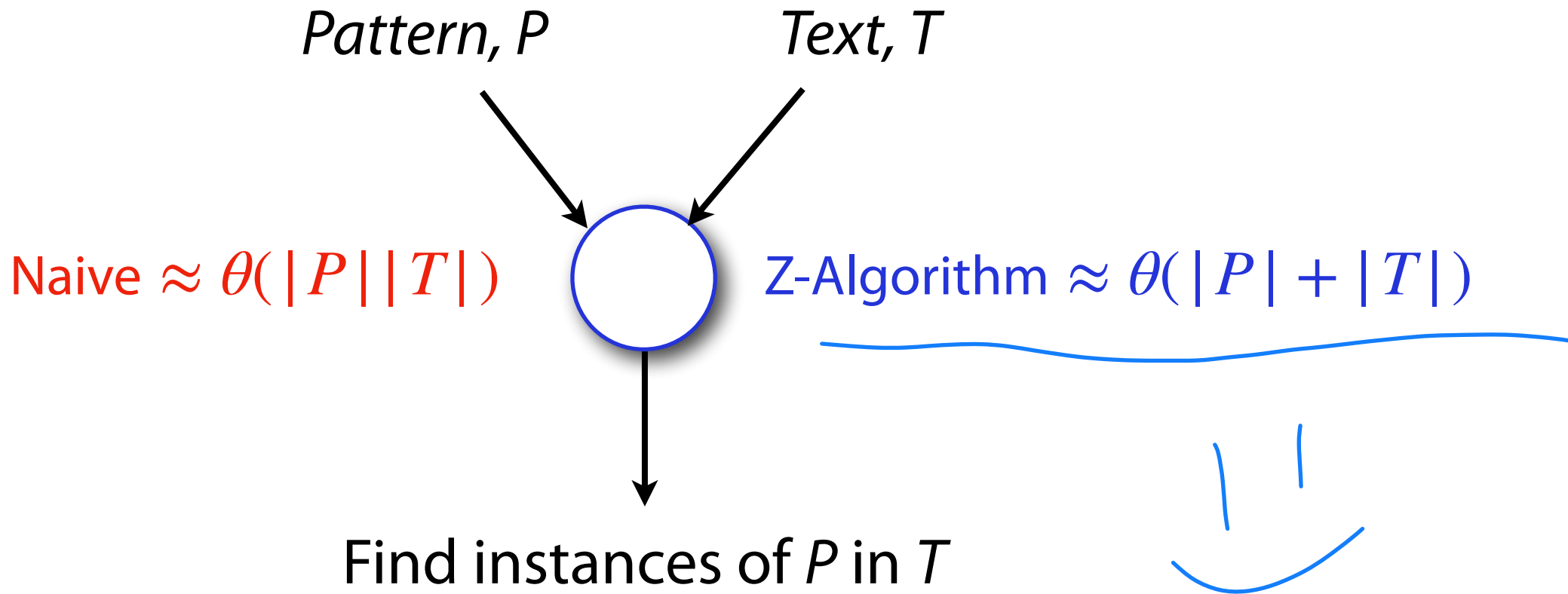


UNIVERSITY OF
ILLINOIS
URBANA-CHAMPAIGN

Department of Computer Science

0 1 2 3 4 5 6 7
A A A B B A A A
 $z_1 = 2$ $z_5 = "$
 $z_2 = 1$ $z_6 = \uparrow \uparrow$
 $z_3 = 0$ $z_7 = \text{☺}$
...

Exact Pattern Matching w/ Z-algorithm



'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
 ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
 ↑ ↑

$$Z_4(S) = 6$$

S: C G C G A ? ? ? ? ?
 C G C X
 Not G

$$Z_5(S) = 3$$

S: A ? ? ? ? ? ? ? ? ?
 A A A A A A X
 Not A

$$Z_1(S) = 7$$

The Z-Algorithm

P \$ T

S: 1 0 1 \$ 1 0 1 0 1 1

0 1 \$ 1 0 1 0 1 1

1 \$ 1 0 1 0 1 1

\$ 1 0 1 0 1 1

1 0 1 0 1 1

0 1 0 1 1

1 0 1 1

0 1 1

1 1

1

16 total char comps

↳ More work than naive!

The Z-Algorithm

$$Z_1 = 3$$

$$Z_2 =$$

\emptyset	1	2	3	4	5	6	7
A	A	A	A	B	B	B	B
A	A	A	A	B	B	B	B

l
 r (my frontier)

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

$z_4 = 3$

0	1	2	3	4	5	6	7	8	9
1	0	1	1	1	0	1	0	1	1
1	0	1	\$	1	0	1	0	1	1

i , the current index =

r , the furthest match char =

l , the furthest reaching Z-value =

Start

End

4

5

2

6

2

4

The Z-Algorithm

0	1	2	3	4	5	6	7	8	9
1	0	1	\$	1	0	1	0	1	1
1	0	1	\$	1	0	1	0	1	1

$z_5 = 0$, w/o doing any work

i , the current index =

r , the furthest match char =

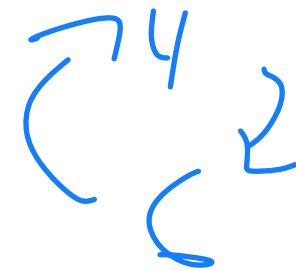
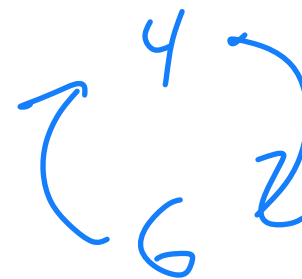
l , the furthest reaching Z-value =

Start

End

5

6



The Z-Algorithm



0	1	2	3	4	5	6	7	8	9
1	0	1	\$	1	0	1	0	1	1
1	0	1	\$	1	0	1	0	1	1

i , the current index =

r , the furthest match char =

l , the furthest reaching Z-value =

Start

End

6

7

6

8

4

6

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

Handwritten annotations: Blue arrows point down from indices 0, 1, and 2. Blue arrows point up from indices 1, 2, and 3. A blue scribble is under index 0.

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

Case 1: $i > r$ default to 0

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

$$Z_1 = 2$$

We must compute Z_i explicitly!

The Z-Algorithm

$Z_5 = 3$

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

Handwritten annotations: Red arrows point from the indices 2 and 5 to the corresponding cells in the table. Red 'X' marks are placed above indices 3 and 4. Red slashes are drawn through the 'A' characters in the first row at indices 0, 1, and 2, and in the second row at indices 5, 6, and 7.

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

2nd character

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 2

The Z-Algorithm

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

$Z_1 = 0$ $Z_5 = 0$

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

To find Z_5 , we can save time by looking up the value Z_1

The Z-Algorithm

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

Handwritten annotations: z_1 (red arrow from index 1 to 4), $z_1=1$ (red arrow to index 1), and red arrows pointing to indices 5 and 6.

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 $\frac{z_l + \text{more work}}{\text{after } r}$



The Z-Algorithm

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

Compute Z_i using irl :

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

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

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

0	1	2	3	4	5	6	7	8
A	A	A	A	C	A	A	A	B
A	A	A	A	C	A	A	A	B
A	A	A	A	C	A	A	A	B

The amount of work required depends on two pieces of information

1. # of characters at or after i that we have seen before

↳ these are our Z-value calc characters

2. The Z-value that matches part or all of the string starting at i

↳ what Z-value do I look up?

The Z-Algorithm

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

0	1	2	3	4	5	6	7	8
A	A	A	A	C	A	A	A	B
A	A	A	A	C	A	A	A	B
A	A	A	A	C	A	A	A	B

The amount of work required depends on two pieces of information

1. # of characters at or after i that we have seen before

Call this value $|\beta|$. What is $|\beta|$ in terms of i, r, l ?

$$\beta = r - i + 1$$

The Z-Algorithm

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

0	1	2	3	4	5	6	7	8
A	A	A	A	C	A	A	A	B
A	A	A	A	C	A	A	A	B
A	A	A	A	C	A	A	A	B

Handwritten annotations: l above index 5, i above index 6, a red arrow pointing from l to i , and a red arrow pointing down from i to index 6.

The amount of work required depends on two pieces of information

2. The Z-value that matches part or all of the string starting at i

Call this value Z_k . What is k in terms of i, r, l ?

$$k = i - l$$

The Z-Algorithm

$$Z_5 = 3$$

$$l = 5$$

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



0	1	2	3	4	5	6	7	8
A	A	A	A	C	A	A	A	B
A	A	A	A	C	A	A	A	B
A	A	A	A	C	A	A	A	B

$$Z_k = Z_1 = 3$$

The amount of work required depends on two pieces of information

1. # of characters at or after i that we have seen before

$$|\beta| = 7 - 6 + 1 = 2$$

2. The Z-value that matches part or all of the string starting at i

$$k = 6 - 5 = 1$$

The Z-Algorithm

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

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

↗
↘

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

$$r - i + 1$$

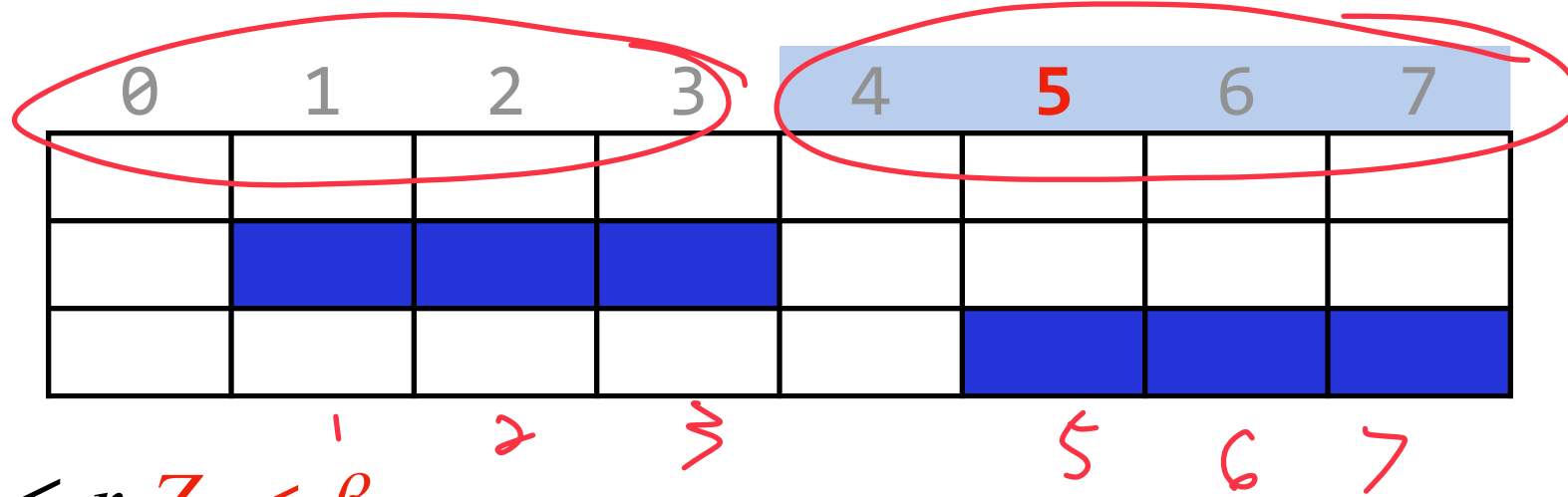
$$|\beta| = \underline{7 - 5 + 1 = 3}, k = \underline{1}, Z_k = \underline{2}$$

$$Z_i = \underline{2}$$

$$Z_i = Z_k$$

The Z-Algorithm

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

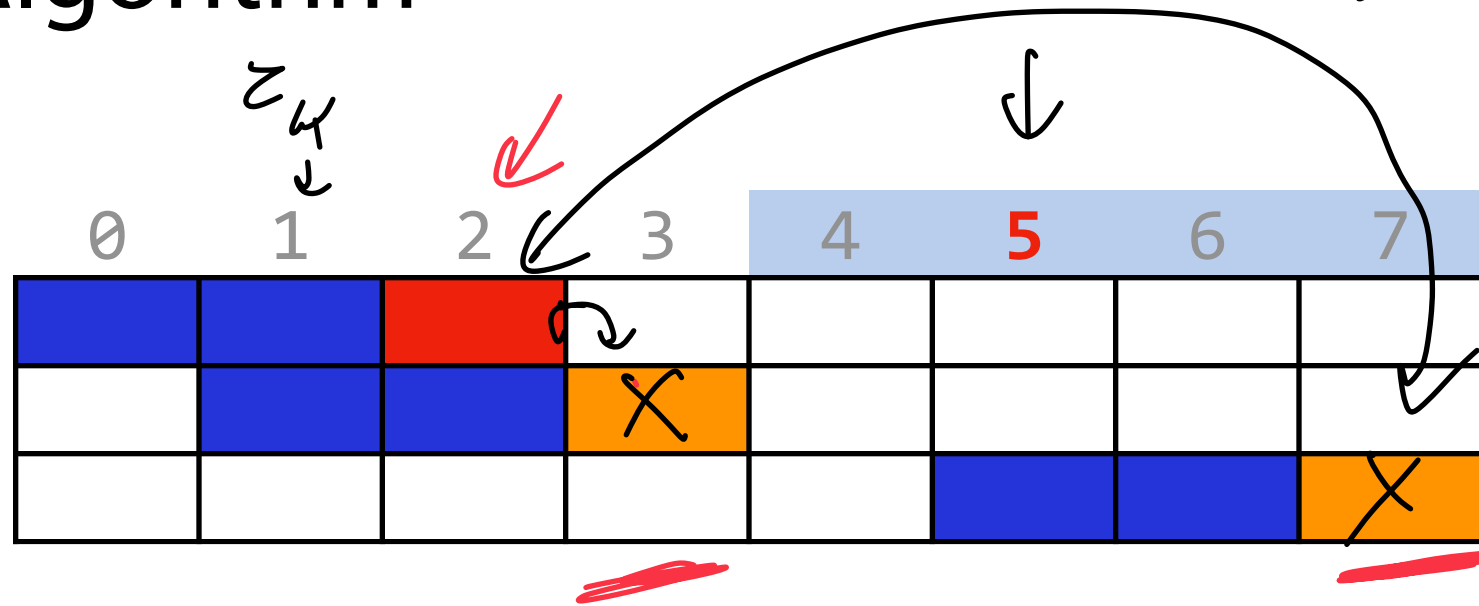


Case 2a: $i \leq r, Z_k < \beta$

Z_l (defined by r, l) tells us that β matches earlier.

The Z-Algorithm

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

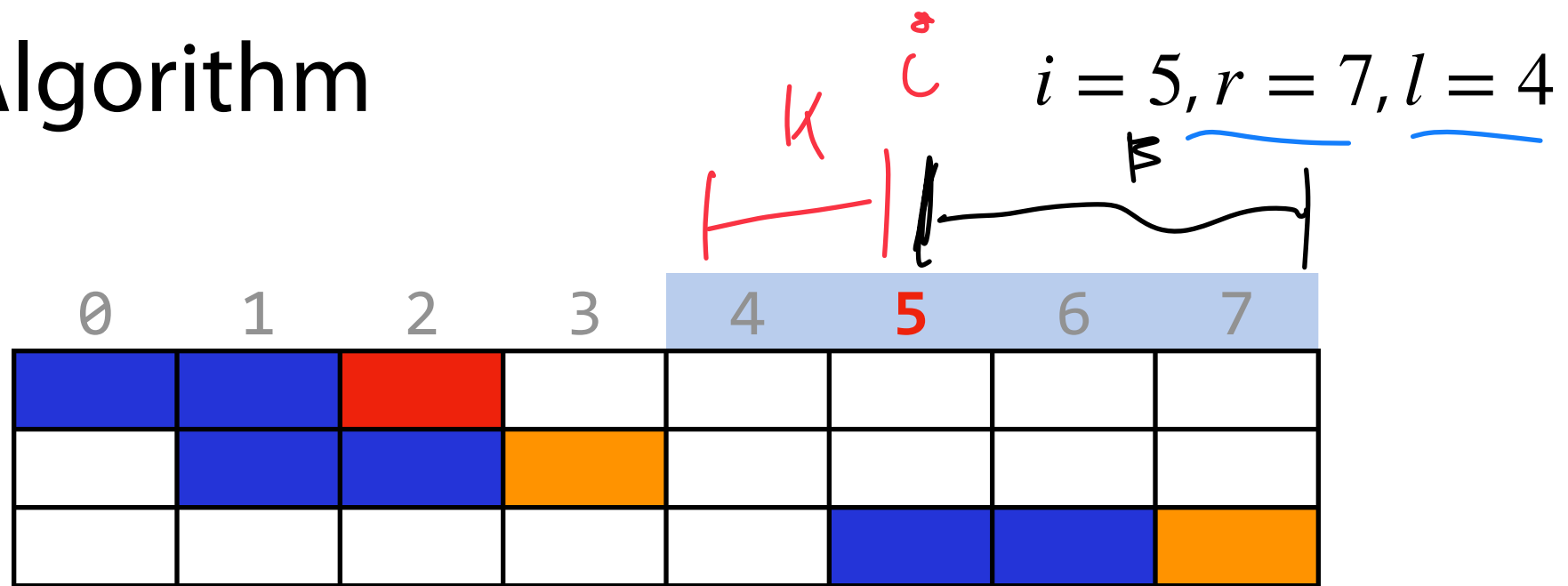


Case 2a: $i \leq r, Z_k < |\beta|$

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

$\beta / \alpha \quad \alpha \neq \beta \quad \neq \quad \beta = \alpha \quad , \quad \alpha \neq \beta \quad ! \quad \downarrow$

The Z-Algorithm



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 $Z_k < |\beta|, Z_i = \underline{Z_k}$

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

$l \rightarrow$
 \rightarrow
 $i = 4$
 $r = 4$

no need to compare
Do this explicitly

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

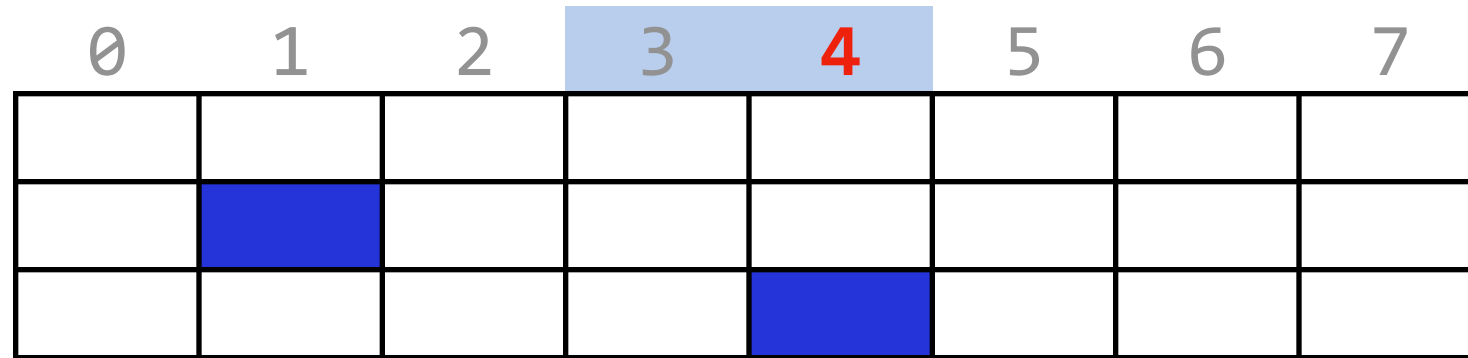
$|\beta| = \underline{4 - 4 + 1 = 1}, k = \underline{1}, Z_k = \underline{1}$

$Z_i = \underline{3}$

The Z-Algorithm

$l = 3$
 $r = 4$

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



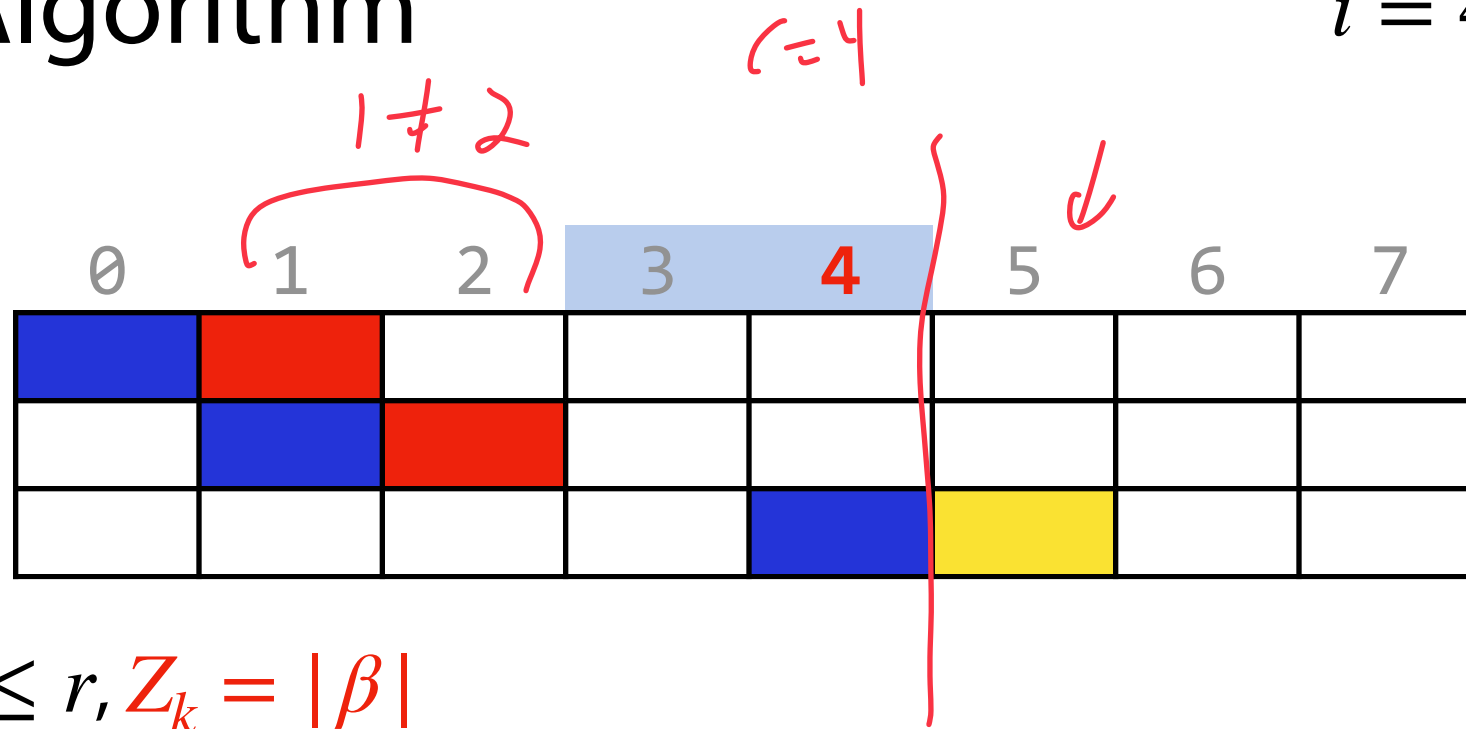
Case 2b: $i \leq r, Z_k = |\beta|$

Z_l (defined by r, l) tells us that β matches earlier.

$01 = = 39$

The Z-Algorithm

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



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

Z_l (defined by r, l) tells us that β matches earlier.

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

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

Handwritten annotations:
 - Blue circles around 'A' at (1,1), (2,2), (3,3), (4,4), (5,5), (6,6), (7,7) in the second row.
 - Blue circles around 'A' at (1,1), (2,2), (3,3), (4,4), (5,5), (6,6), (7,7) in the first row.
 - Blue arrows pointing to (1,1), (2,2), (3,3), (4,4) from above.
 - Blue arrows pointing to (5,5), (6,6), (7,7) from below.
 - A blue 'X' above (4,4) and another blue 'X' to the right of (7,7).
 - A red 'r+1' with a blue underline above (5,5).
 - A blue 'Z_k' next to (6,6).
 - A blue 'B' next to (7,7).
 - A red bracket on the right side of the page.

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

$|\beta| = 1, k = 1, Z_k = 1$

$Z_i = Z_k + \underline{\text{explicit calc}} \left(\underline{r+1} \text{ vs } \begin{matrix} Z_k \\ \beta \end{matrix} \right)$

The Z-Algorithm

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

\emptyset	1	2	3	4	5	6	7
A	A	A	A	A	A	B	C
A	A	A	A	A	A	B	C

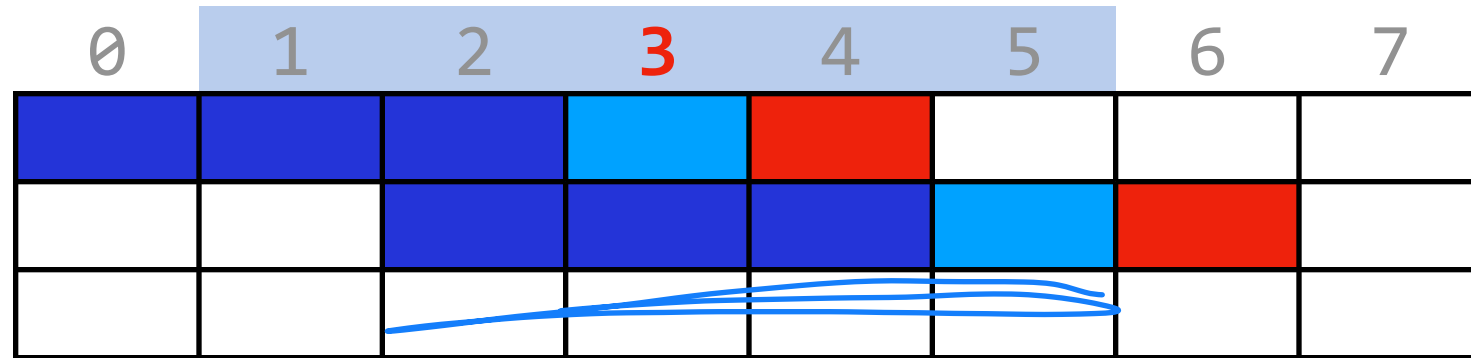
Case 2c: $i \leq r, Z_k > |\beta|$

$$|\beta| = \underline{3}, k = \underline{2}, Z_k = \underline{4}$$

$$Z_i = \underline{|\beta| = 3}$$

The Z-Algorithm

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

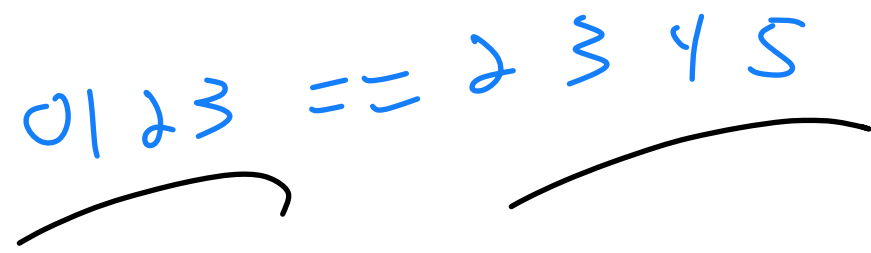


Case 2c: $i \leq r, Z_k > |\beta|$

Z_k tells us how much matches the prefix.

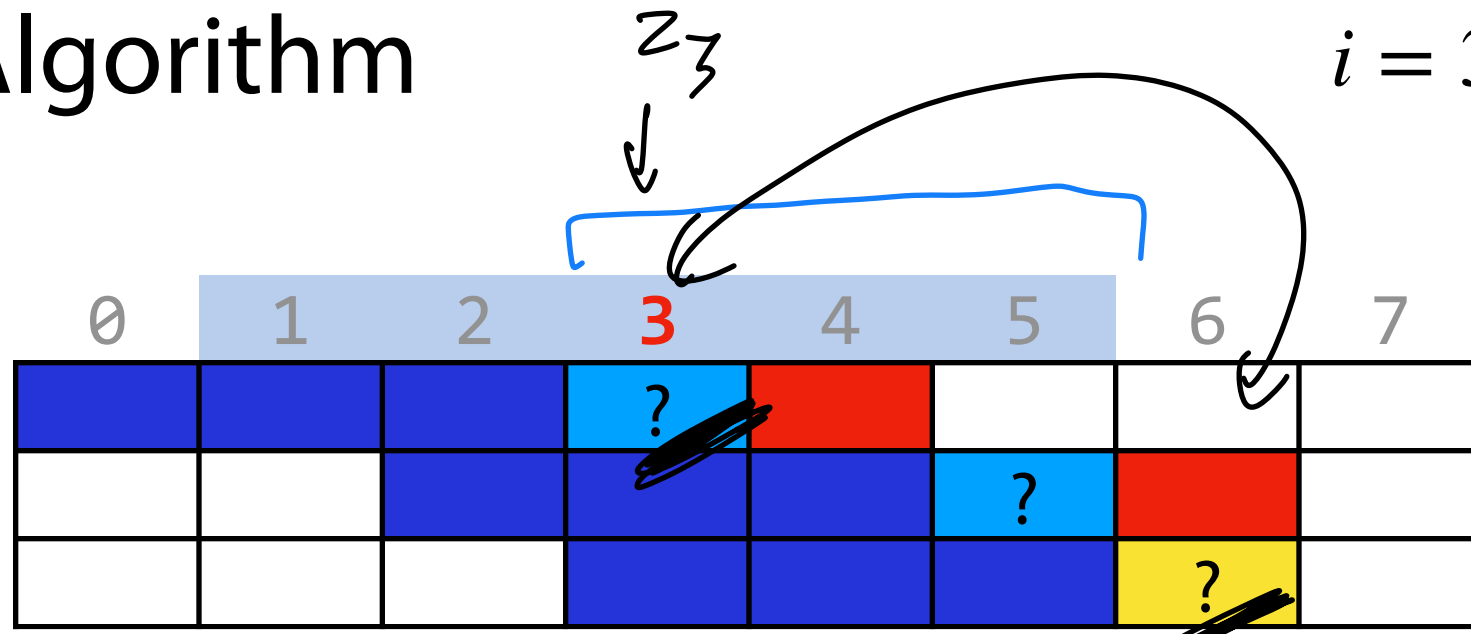
$$\rightarrow z = 5$$

$$z_3 = 4$$



The Z-Algorithm

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



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

$$0 | 2 = 2 \geq 1 = \geq 4 5$$

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

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

$z_l = z_1 = 5$

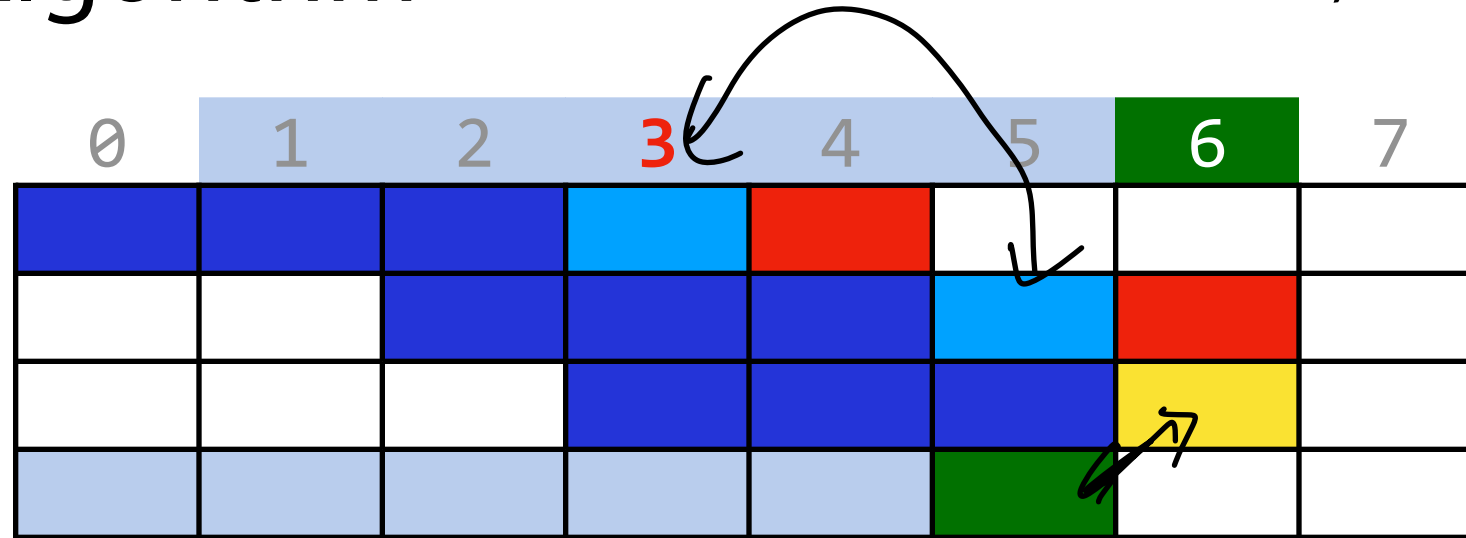
0	1	2	3	4	5	6	7
0	1	2	3	4	X		
	1	2	3	4	5	X	

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

Z_l tells us that our entire range (β included) matches earlier
... and that it failed to match the next character.

The Z-Algorithm

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



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

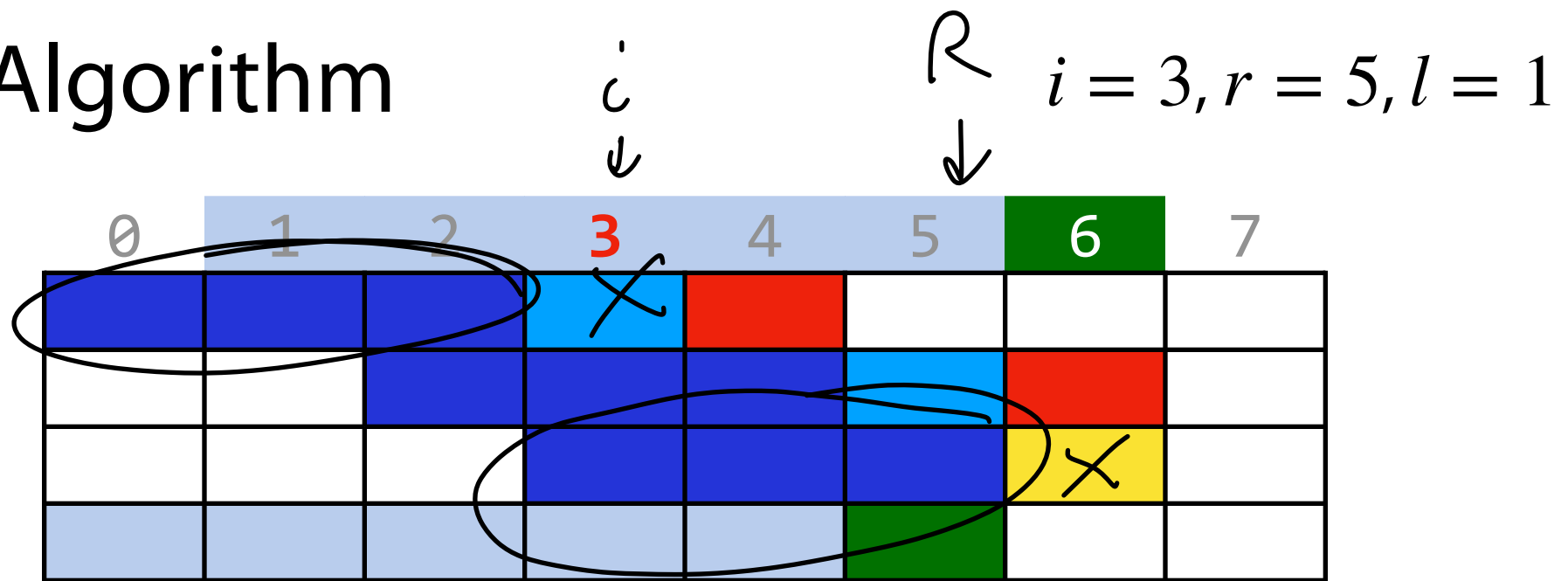
$$3 = 5 \neq 6$$

$$3 \neq 6$$

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



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 $Z_k > |\beta|$, $Z_i = \underline{\quad |\beta| \quad}$

The Z-Algorithm

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

Compute Z_i using irl :

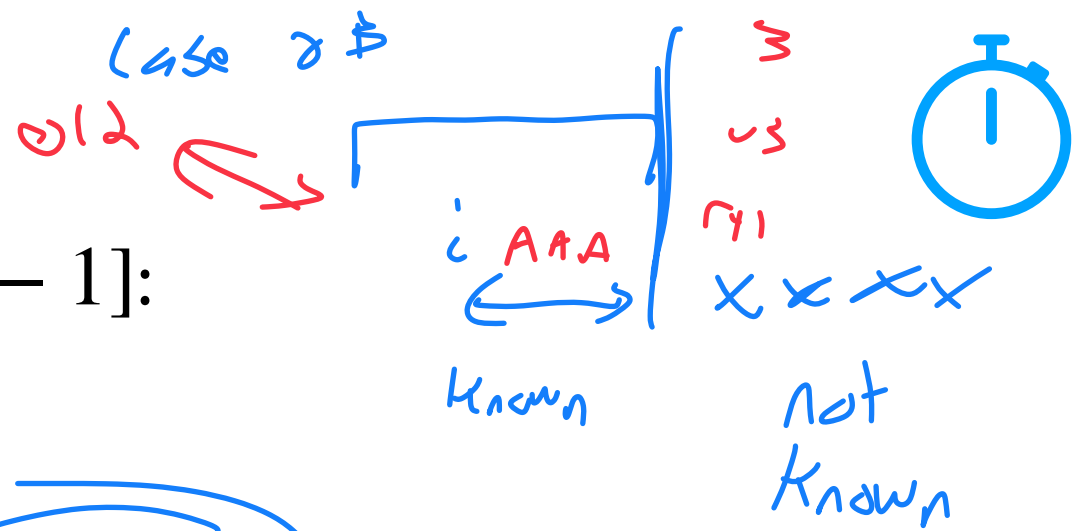
Case 1 ($i > r$): Compute explicitly; update irl

Case 2 ($i \leq r$):

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

2b: ($Z_k = |\beta|$): $Z_i = Z_k + \text{explicit}(\text{indices } r + 1 \text{ vs } Z_k)$; update irl

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



Assignment 3: a_zalg

Learning Objective:

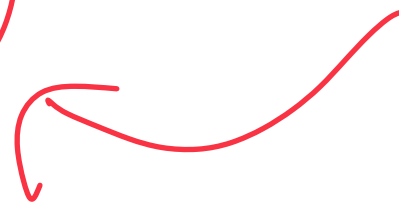
Construct the full Z-algorithm and measure its efficiency

↙ 100 pts

Demonstrate use of Z-algorithm in pattern matching

↘ 0 pts!

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.