String Algorithms and Data Structures

The Z-algorithm

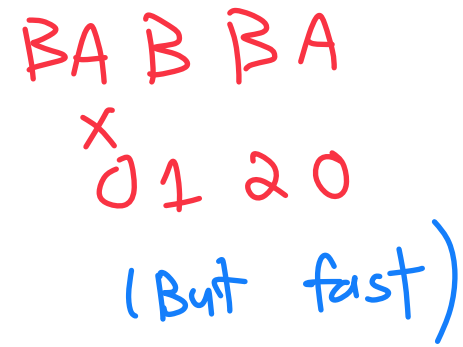
CS 199-225

Brad Solomon

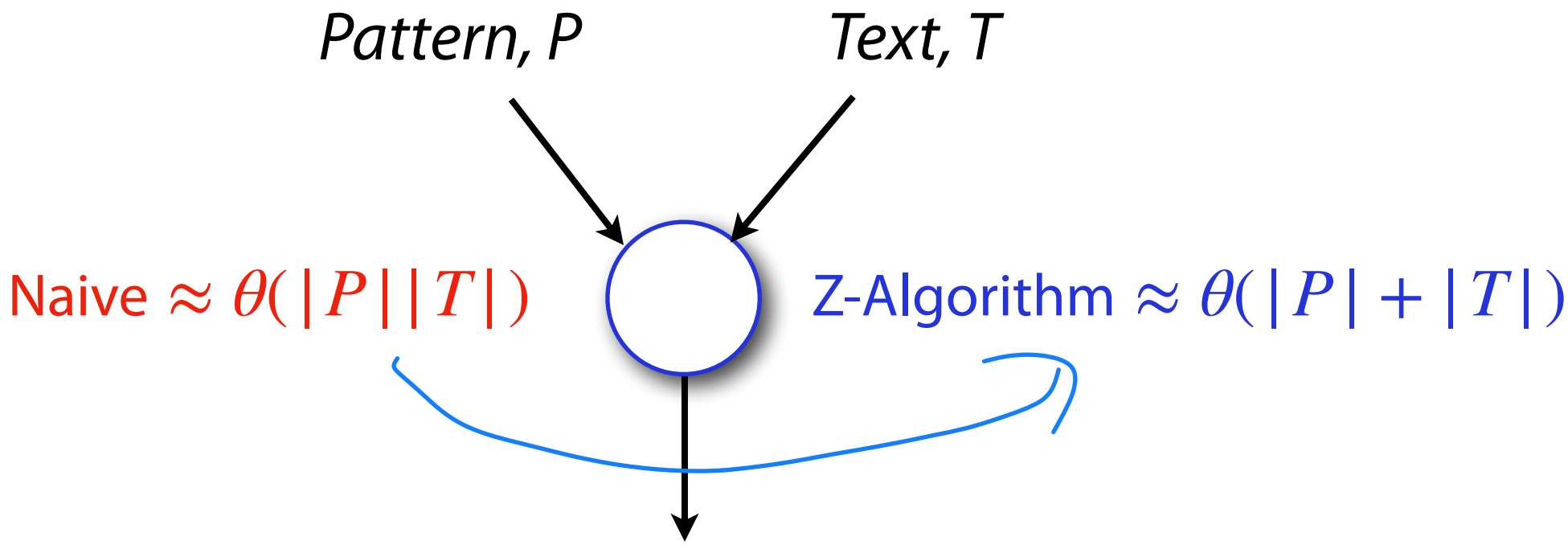
February 9, 2026

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science



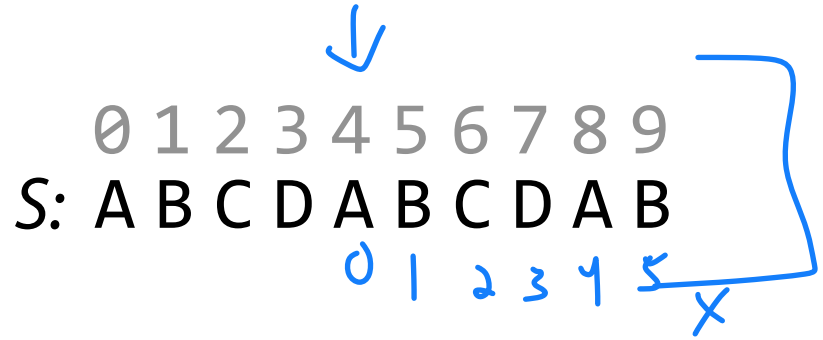
Exact Pattern Matching w/ Z-algorithm



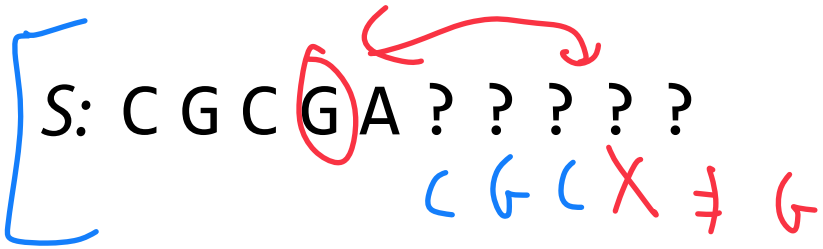
'instances': An exact, full length copy

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

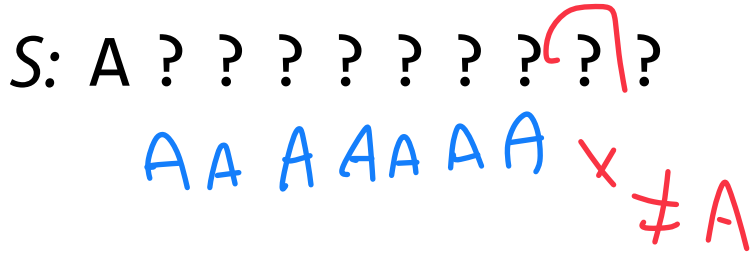
Given a string S, $Z_i(S)$ is the length of the longest substring in S, starting at position $i > 0$, that matches a prefix of S.



$Z_4(S) =$ 6



$Z_5(S) = 3$



$Z_1(S) = 7$

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
```

$Z_4(S) = 6$

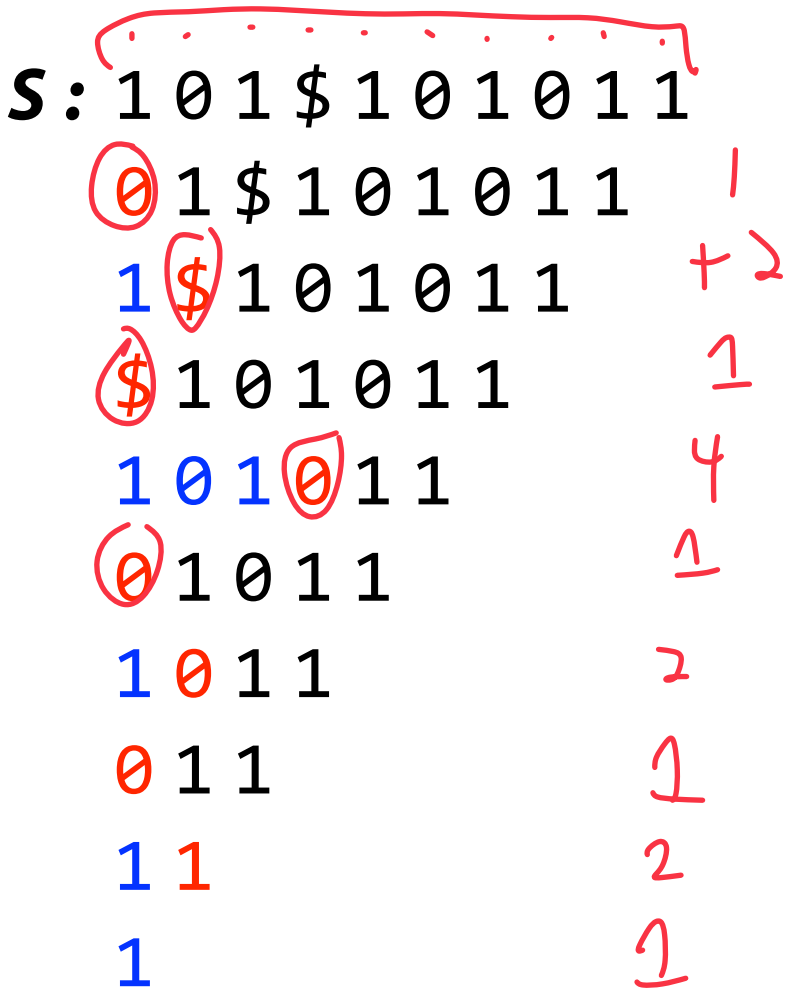
```
S: C G C G A C G C X ?
                X ! = G
```

$Z_5(S) = 3$

```
S: A A A A A A A A X ?
                X ! = A
```

$Z_1(S) = 7$

The Z-Algorithm



16 characters total

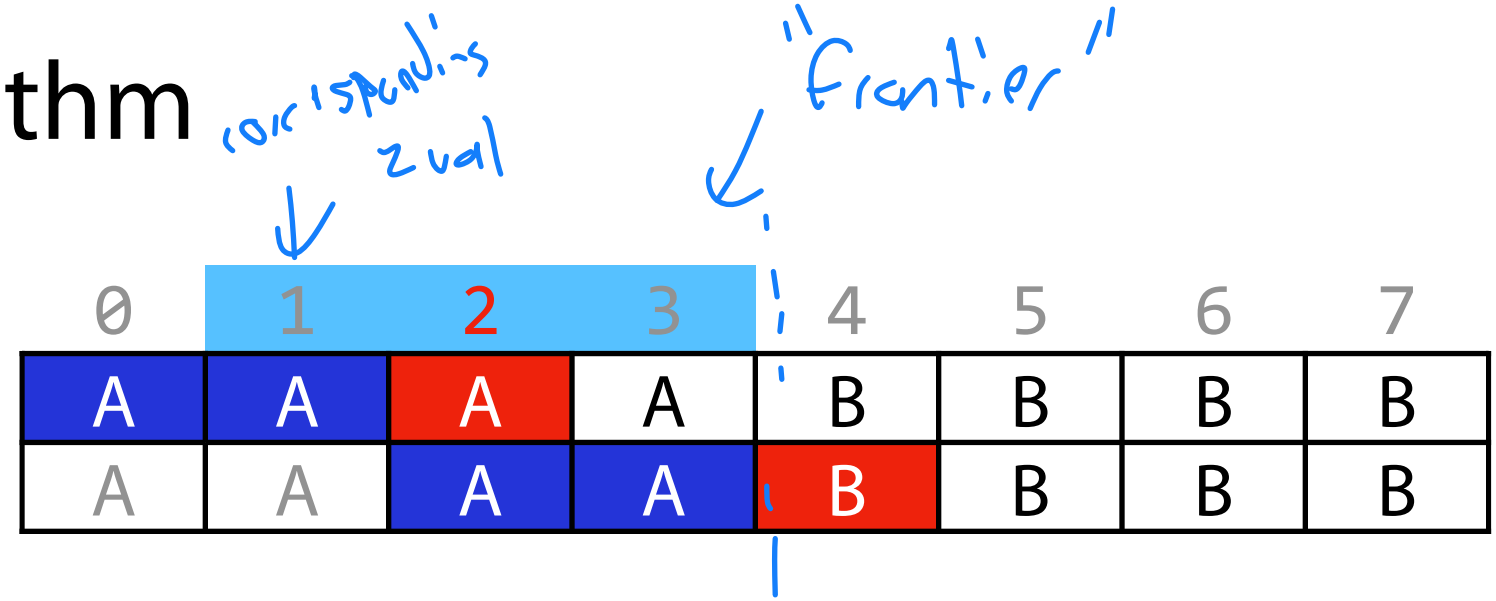
S length 10

Not efficient at all ☺

The Z-Algorithm

$Z_1 = 3$

$Z_2 =$



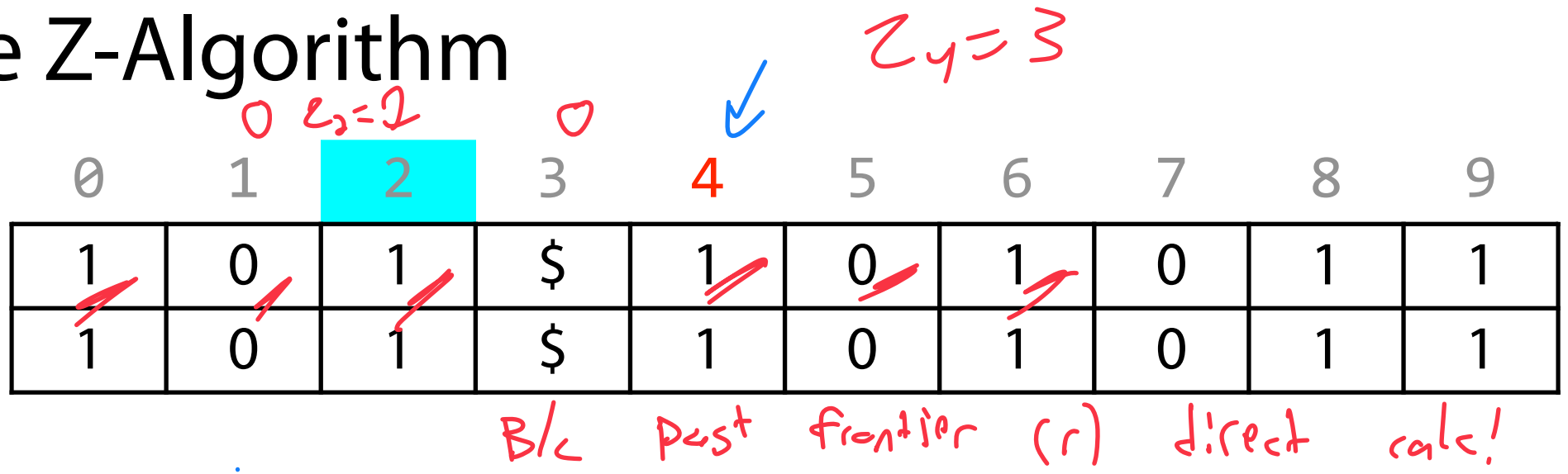
We track our current knowledge of *S* using three values: i, r, l

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



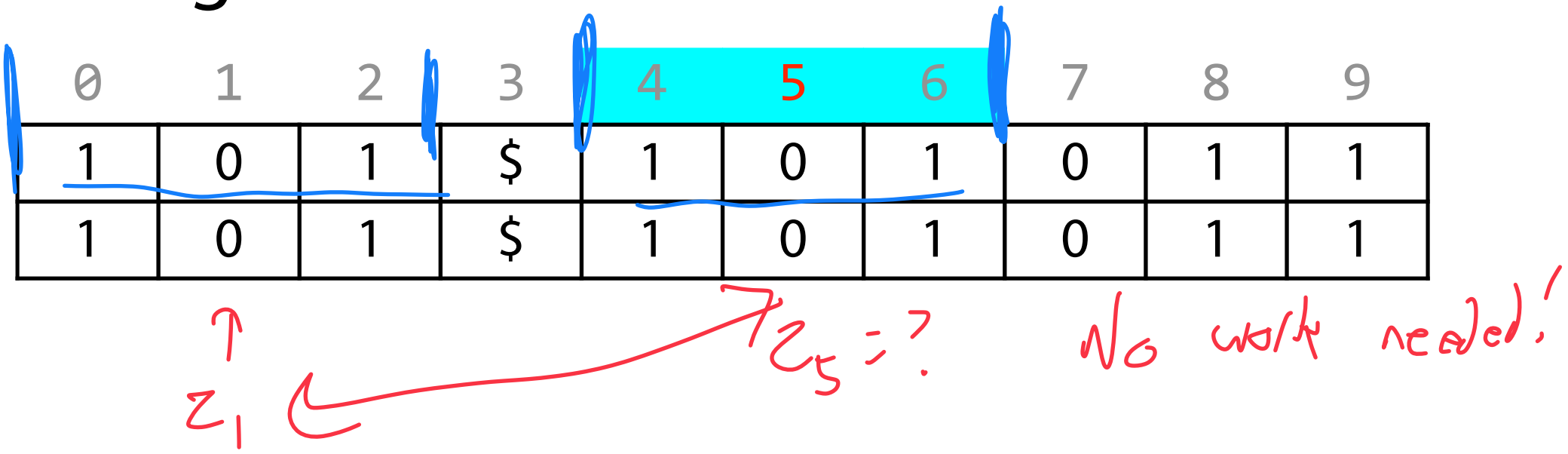
i, the current index =

r, the furthest match char =

l, the furthest reaching Z-value =

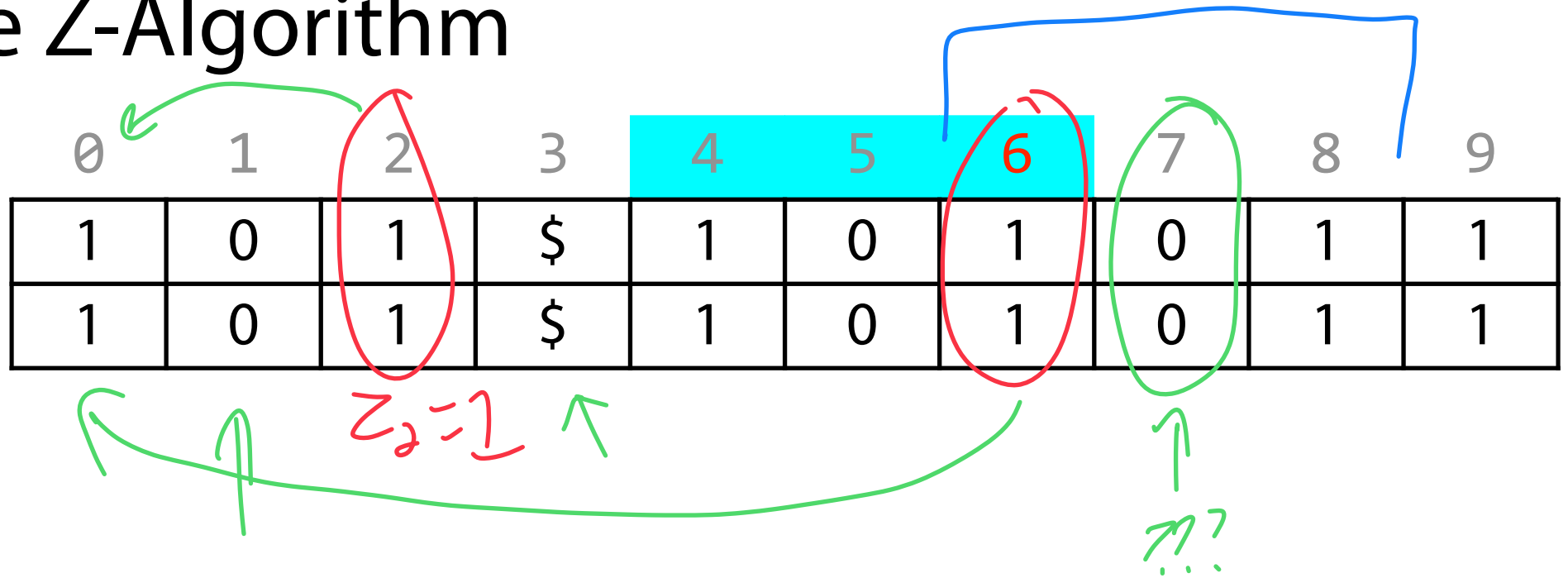
	Start	End
i, the current index =	4	5
r, the furthest match char =	2	6
l, the furthest reaching Z-value =	2	4

The Z-Algorithm



	Start	End
i, the current index =	5	6
r, the furthest match char =	6	6
l, the furthest reaching Z-value =	4	4

The Z-Algorithm



	Start	End
i, the current index =	6	7
r, the furthest match char =	6	8
l, the furthest reaching Z-value =	4	6

The Z-Algorithm

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

At a glance, there seems to be three distinct cases of calculation…

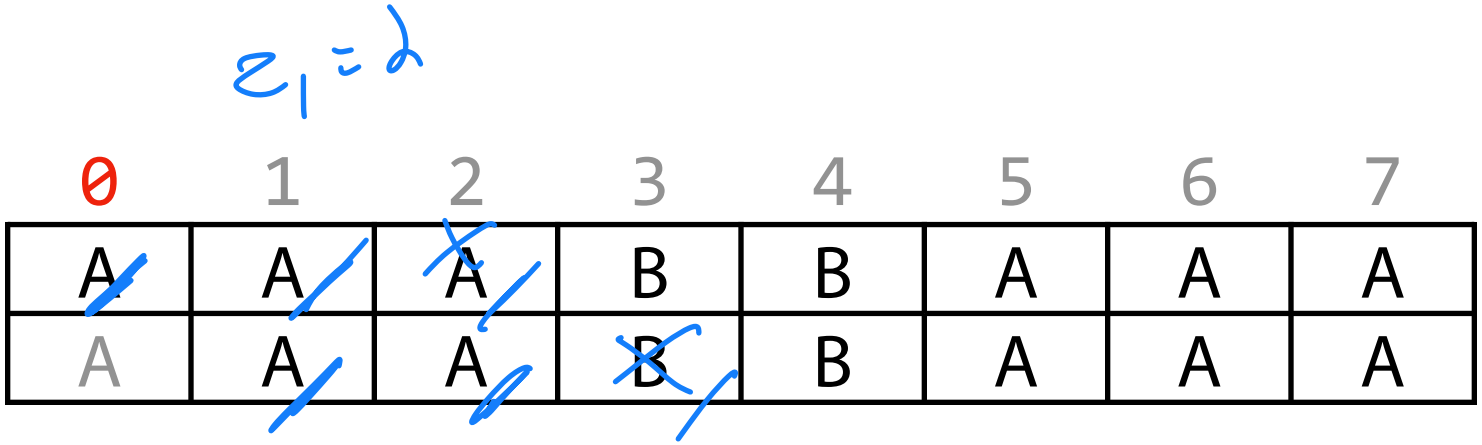
1) I know nothing (naive calculation needed)

2) I know everything already (no work needed)

3)I know some things (calculate only what is not known)

Lets break down how we can tell which case we are in!

The Z-Algorithm



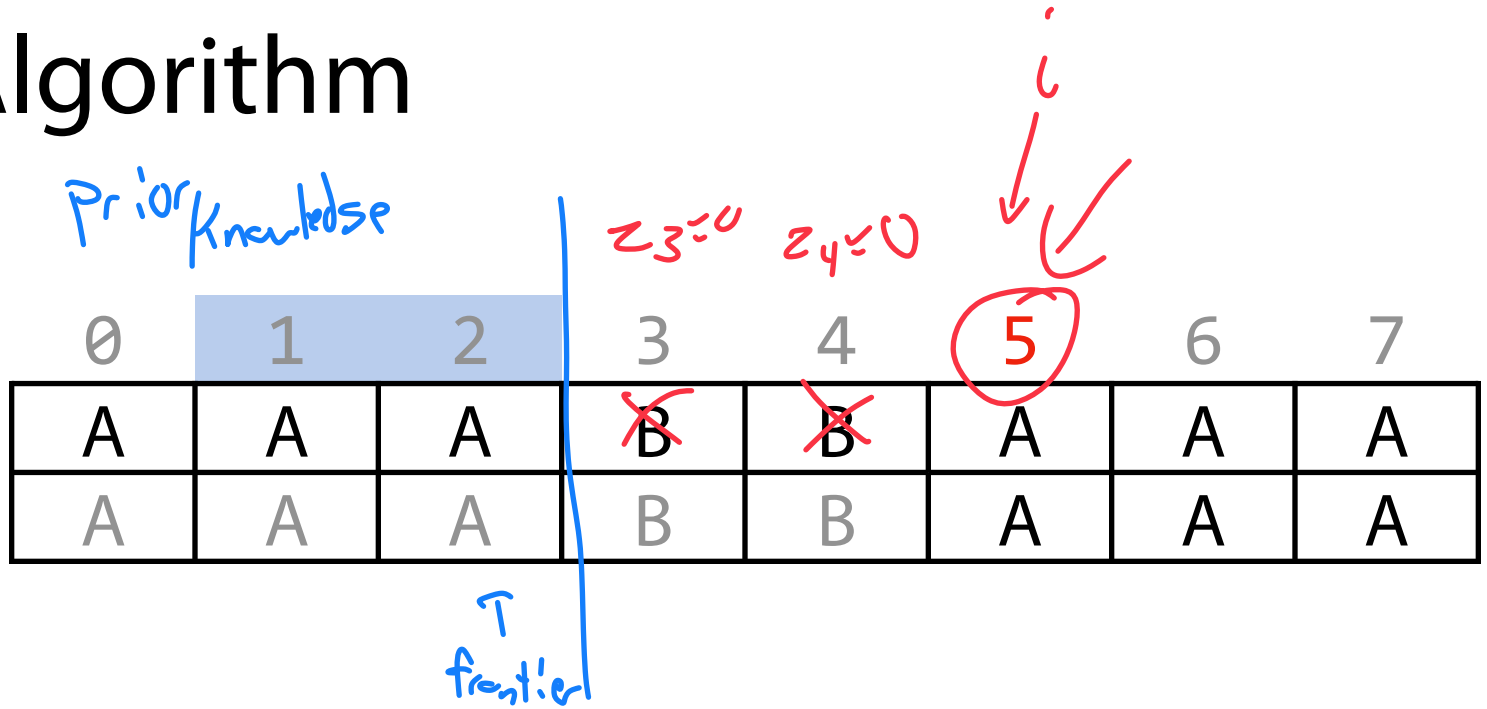
The values of i, r, l tell us how much work we need to do to compute Z_i

Case 1: $i > r$

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

We must compute Z_i explicitly!

The Z-Algorithm



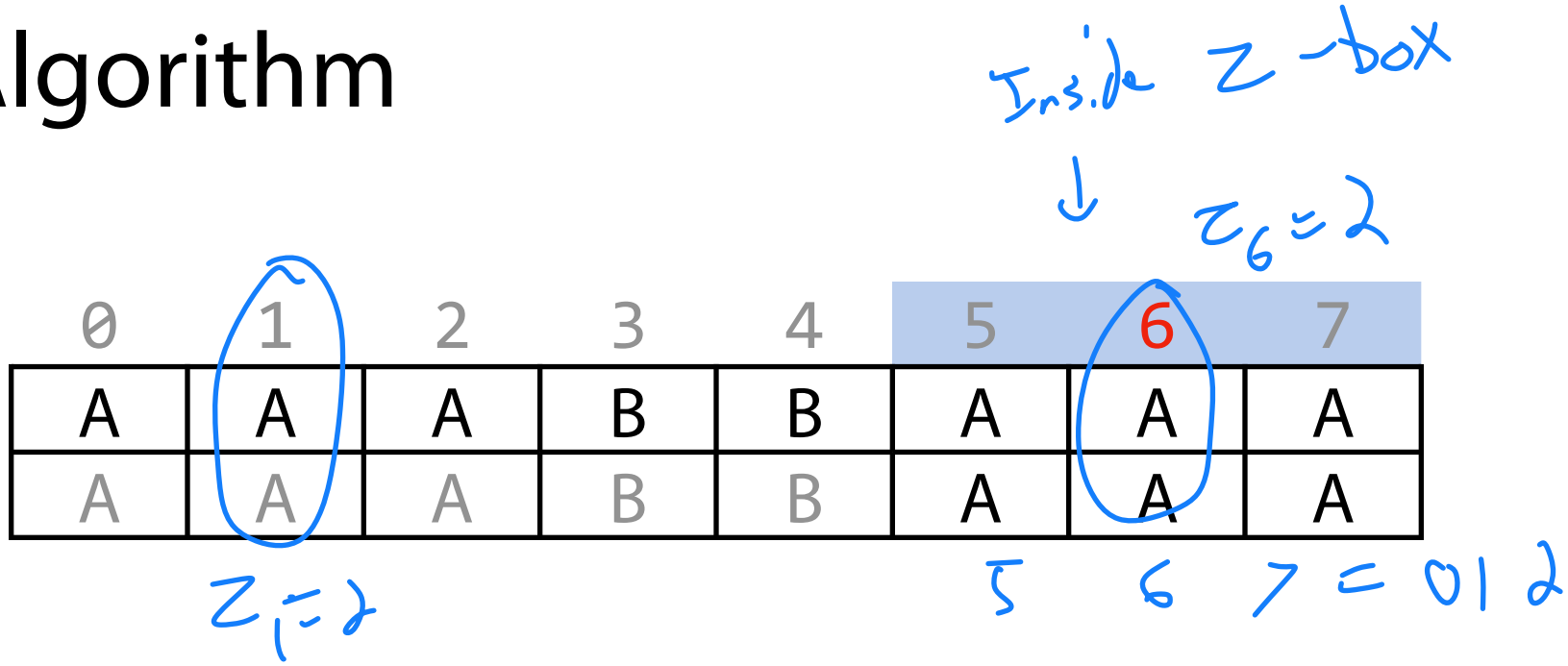
The values of i, r, l tell us how much work we need to do to compute Z_i

Case 1: $i > r$

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

We must compute Z_i explicitly!

The Z-Algorithm



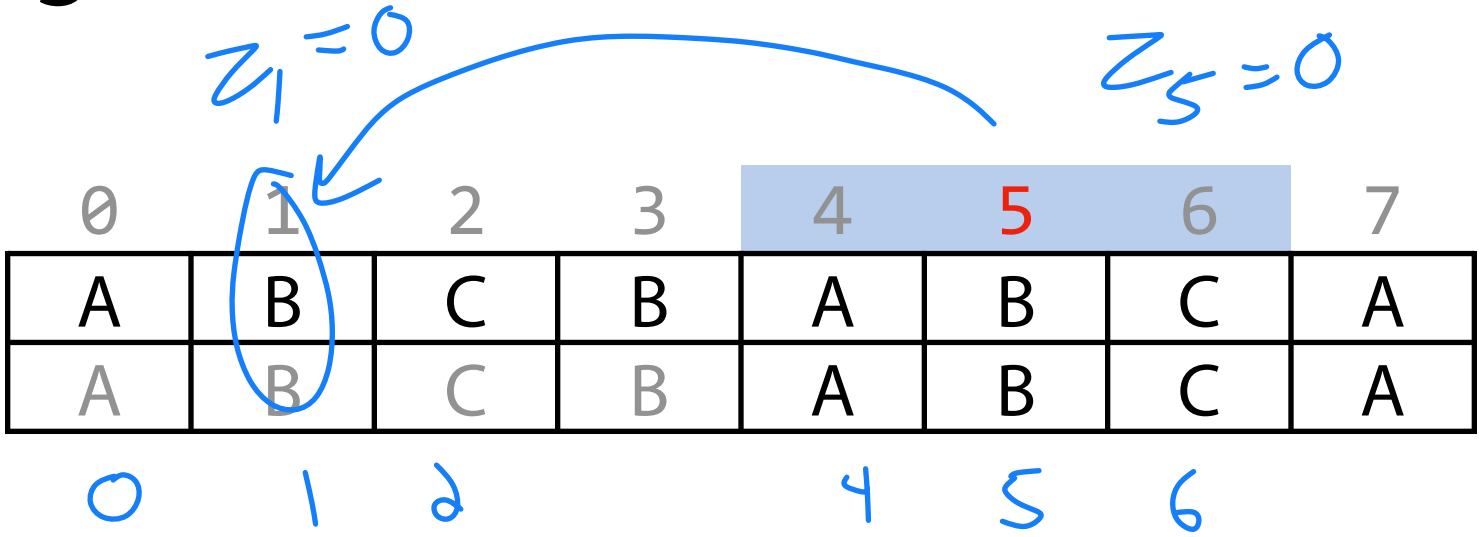
The values of i, r, l tell us how much work we need to do to compute Z_i

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 Z_1

The Z-Algorithm



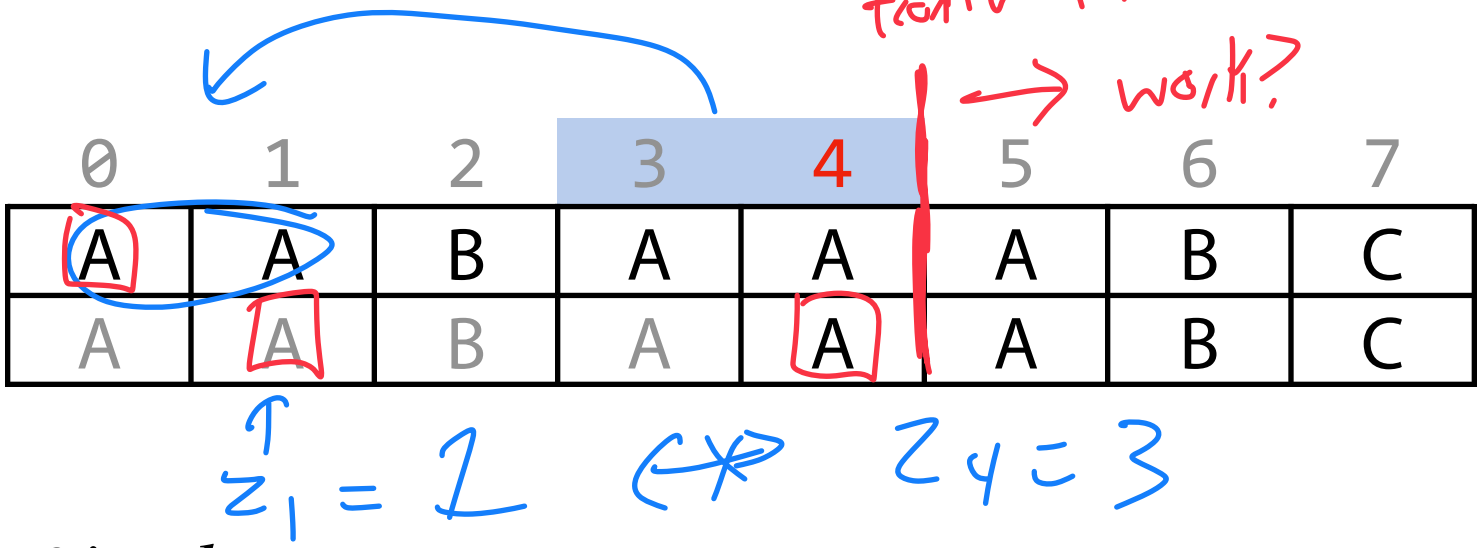
The values of i, r, l tell us how much work we need to do to compute Z_i

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



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 [if a 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?

3 sub 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

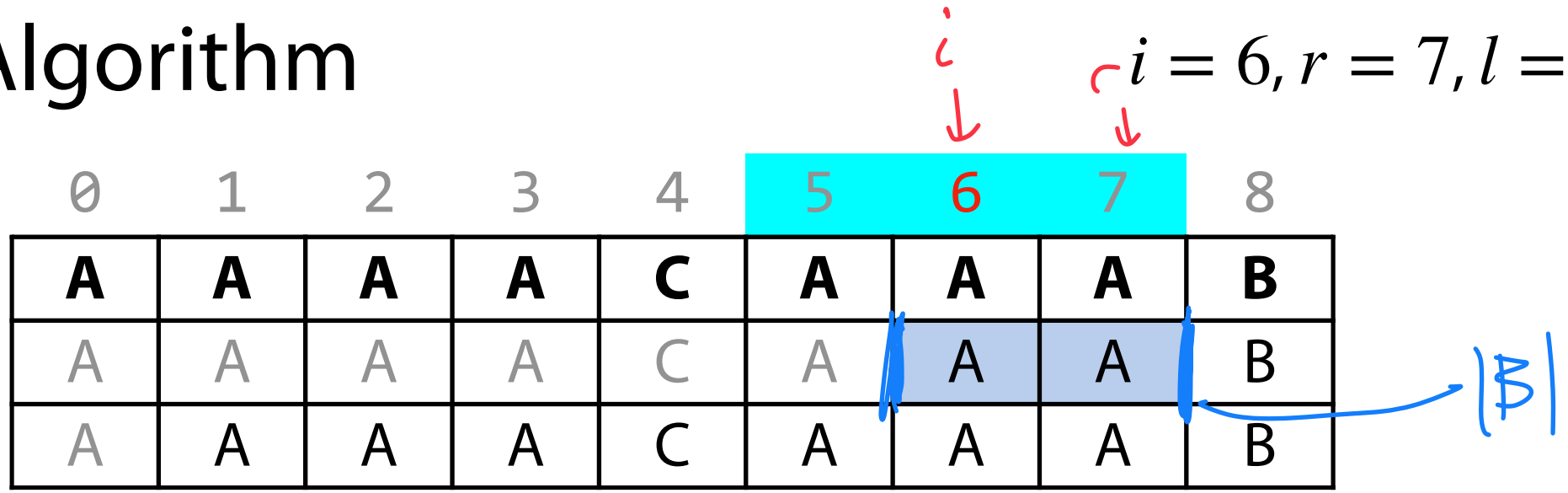
1. # of characters at or after i that we have matched before

If z_6 I might need info on 6 & 7

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

↳ Lets call this z_k (what I look up for info)

The Z-Algorithm



The amount of work required depends on two pieces of information

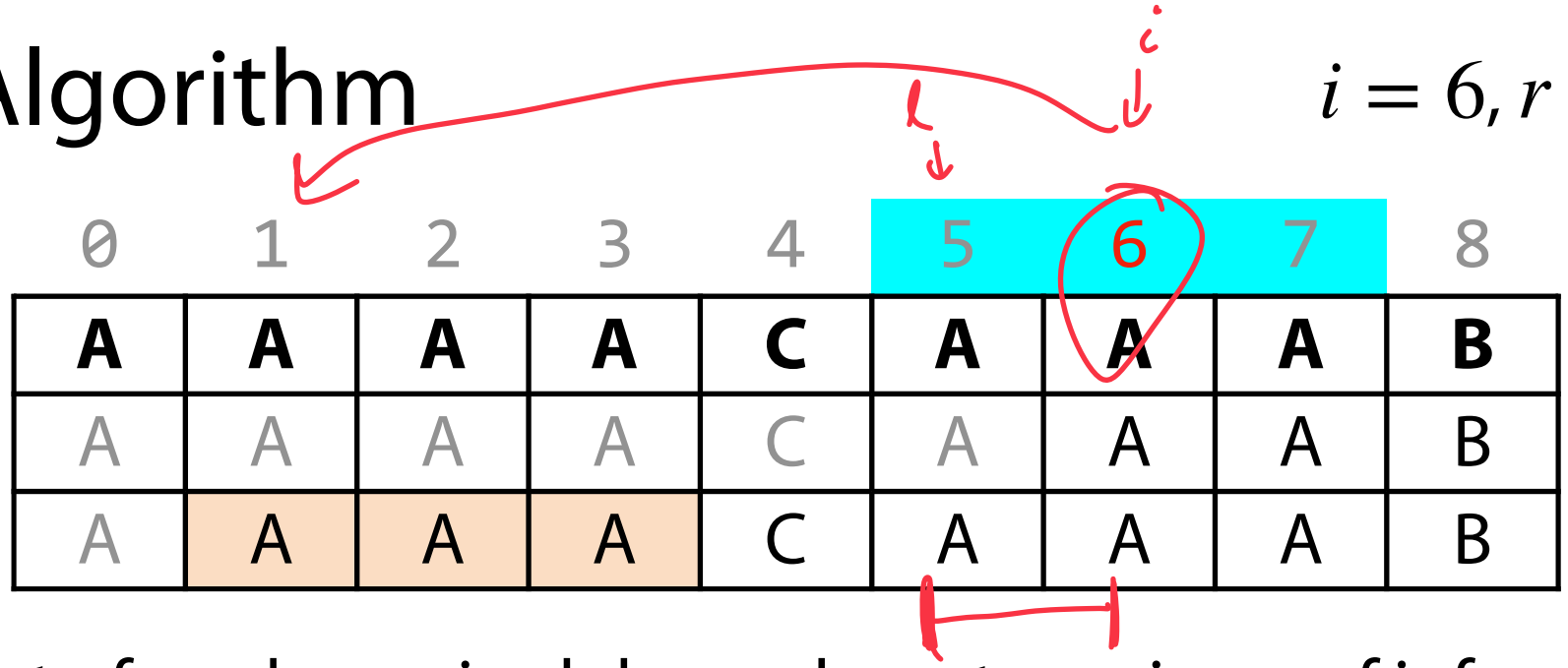
1. # of characters at or after i that we have matched before

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

$|B| = r - i + 1$

The Z-Algorithm

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



The amount of work required depends on two pieces of information

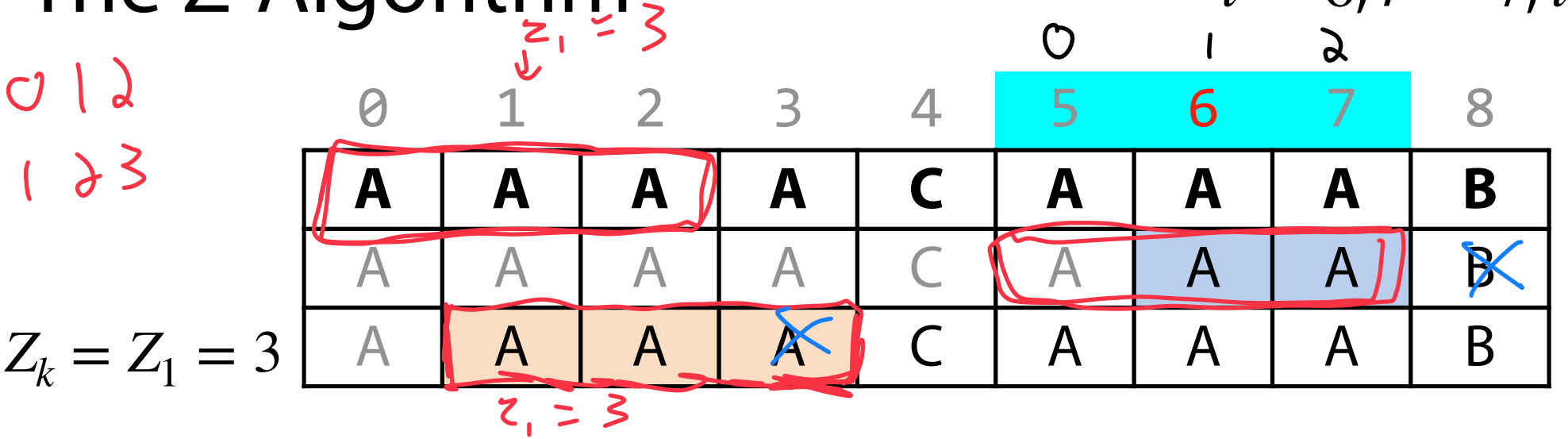
2. The Z-value that matches part or all of the string starting at i

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

$k = i - l$ → Define Z_k (took up)

The Z-Algorithm

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



The amount of work required depends on two pieces of information

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

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

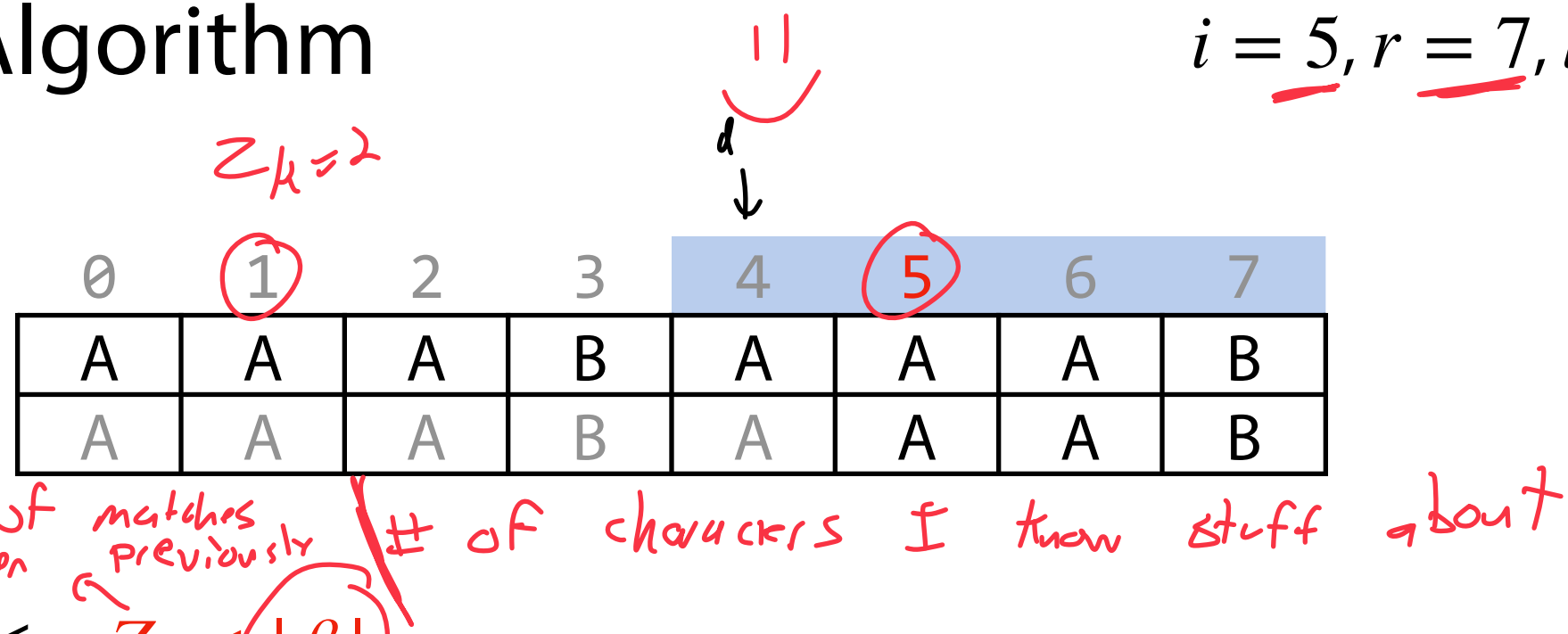
2. The Z-value that matches part or all of the string starting at i

$0 1 2 = 1 2 3 = 5 6 7$

$$k = 6 - 5 = 1$$

The Z-Algorithm

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



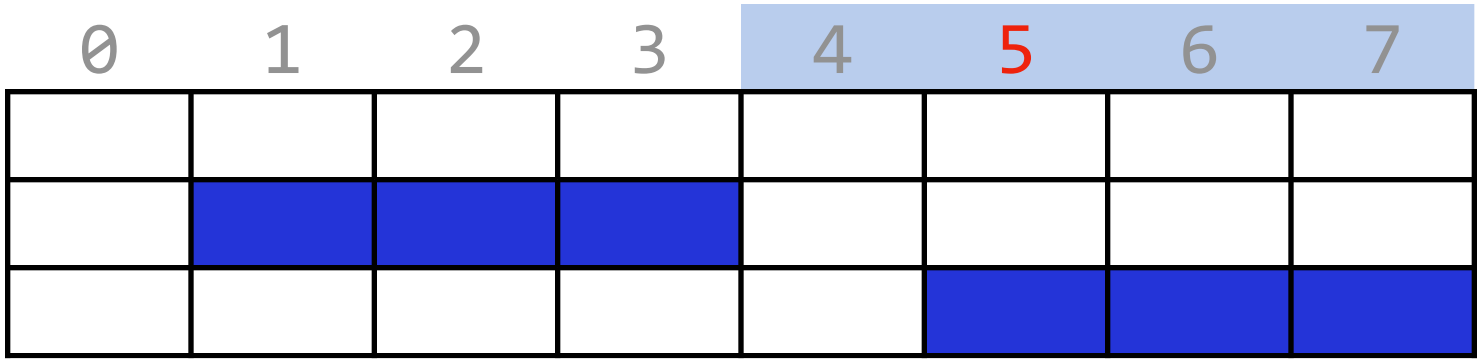
Case 2a: $i \leq r, Z_k < |\beta|$

$|\beta| =$ 3 , $k =$ 1 , $Z_k =$ 2

$Z_i =$ 2

The Z-Algorithm

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

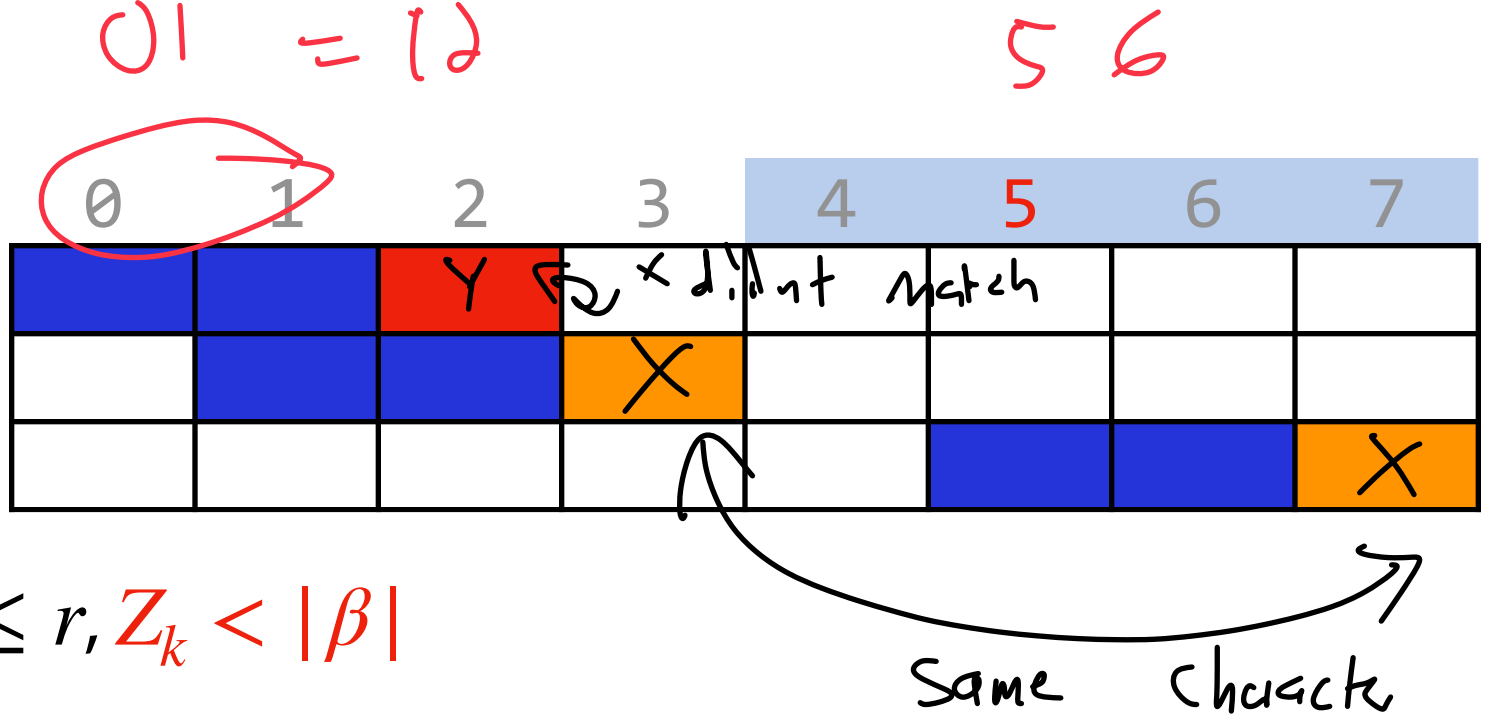


Case 2a: $i \leq r, Z_k < |\beta|$

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

The Z-Algorithm

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

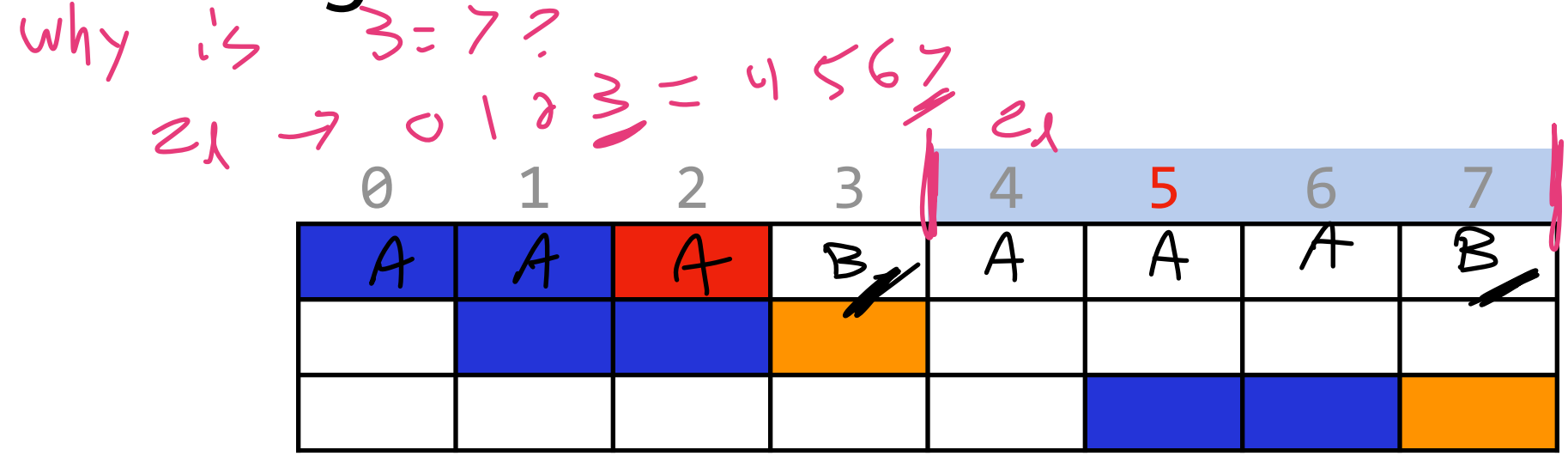


Case 2a: $i \leq r, Z_k < |\beta|$

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

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 $Z_k < |\beta|, Z_i =$ Zk No work needed!

The Z-Algorithm

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

$Z_1 = Z_k = 1$

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

characters matched prev

of characters I can get info about

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

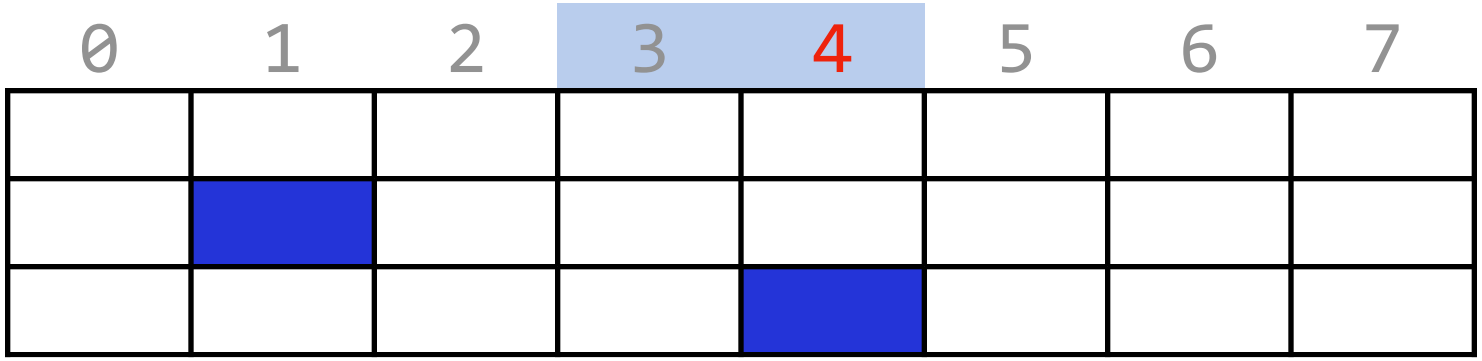
$|\beta| =$ 1 , $k =$ 1 , $Z_k =$ 1

$Z_i =$ 1 + Stuff Starting from 5

$Z_i =$ (Z_k)

The Z-Algorithm

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



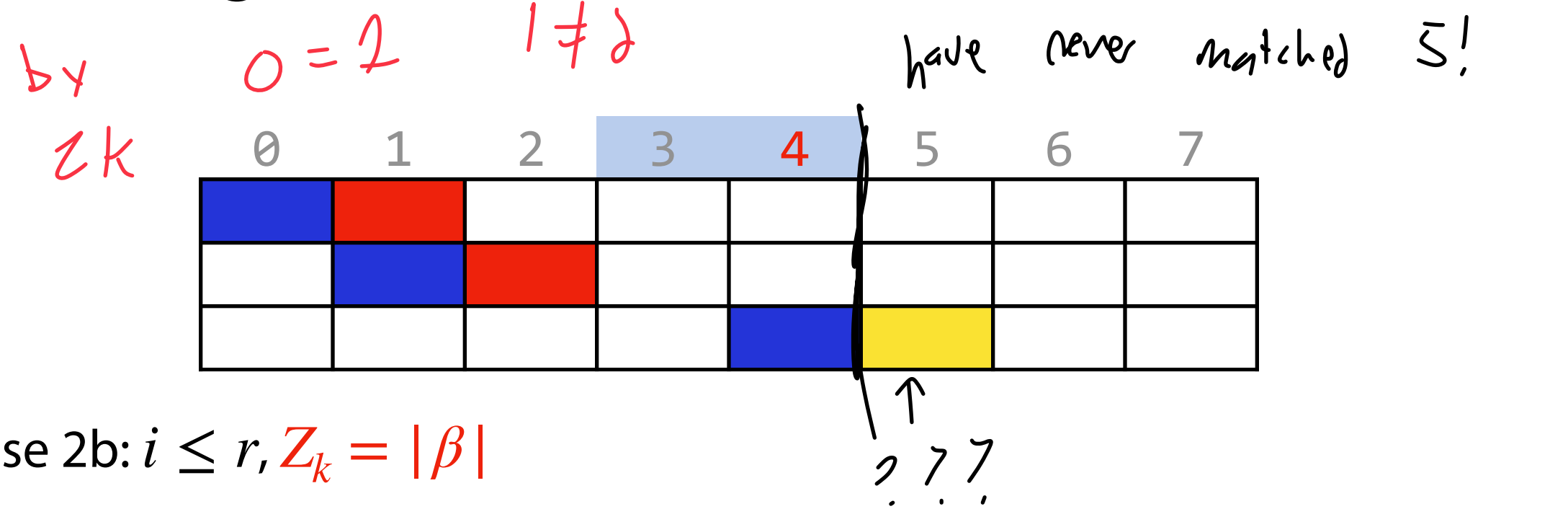
Case 2b: $i \leq r, Z_k = |\beta|$

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

01 = 34

The Z-Algorithm

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



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

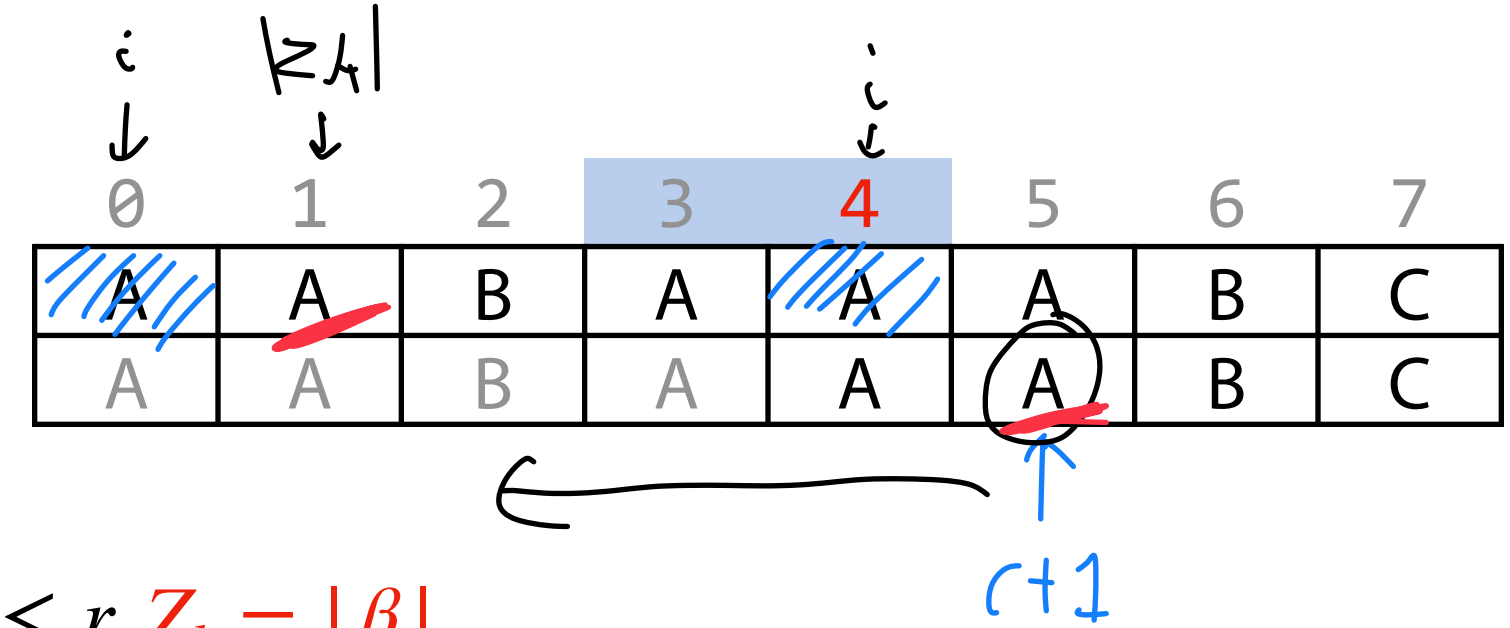
Z_l (defined by r, l) tells us that β matches earlier.

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

The Z-Algorithm

common mistake!

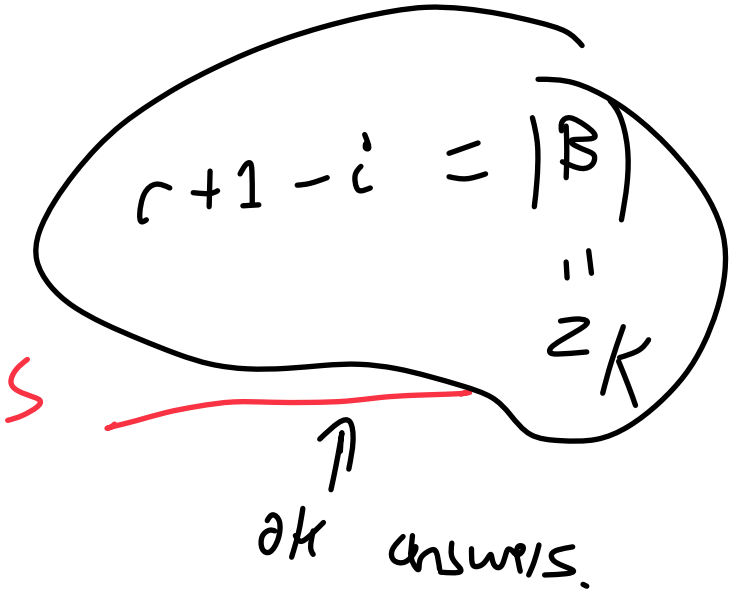
$i = 4, r = 4, l = 3$



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

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

$Z_i = Z\_k +$ comparison starting at c+1 vs



The Z-Algorithm

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

$Z_3 = 4$

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

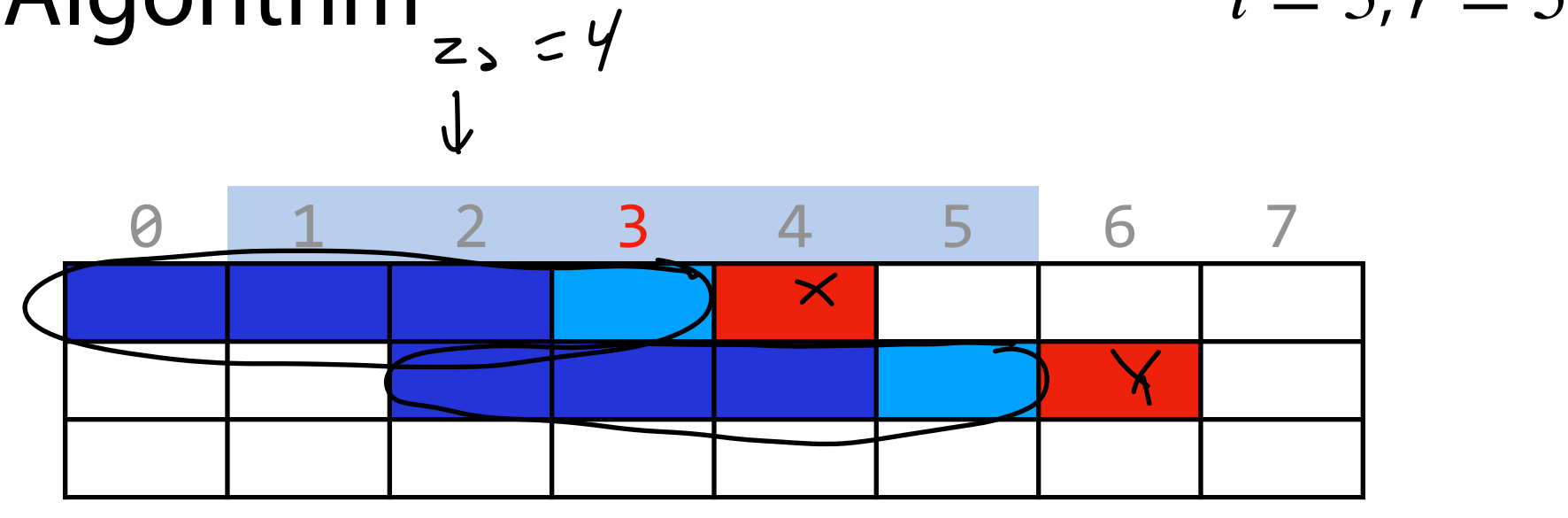
Case 2c: $i \leq r, Z_k > |\beta|$

$|\beta| =$ 3 , $k =$ 2 , $Z_k =$ 4

$Z_i =$ $|\beta|$

The Z-Algorithm

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



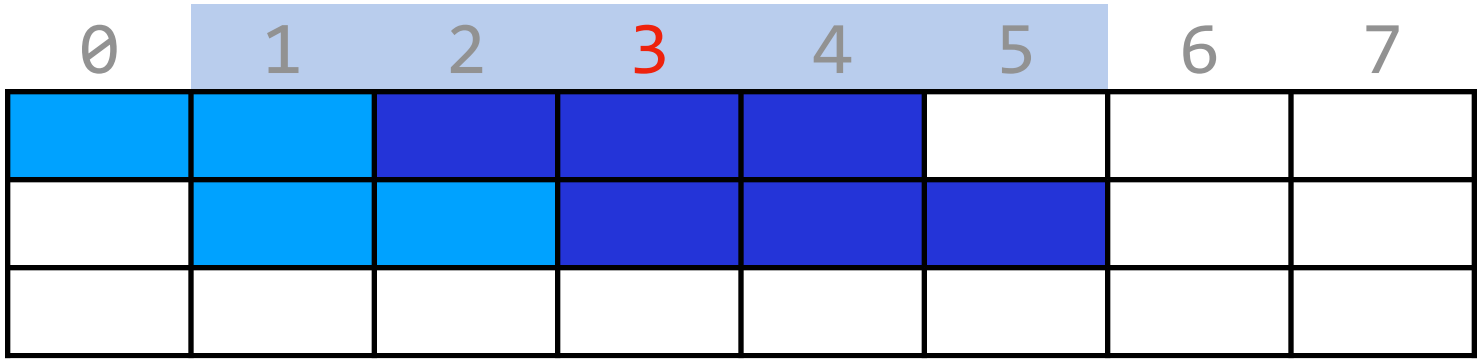
Case 2c: $i \leq r, Z_k > |\beta|$

Z_k tells us how much matches the prefix.

The Z-Algorithm

0 1 2 3 4 = 1 2 3 4 5

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

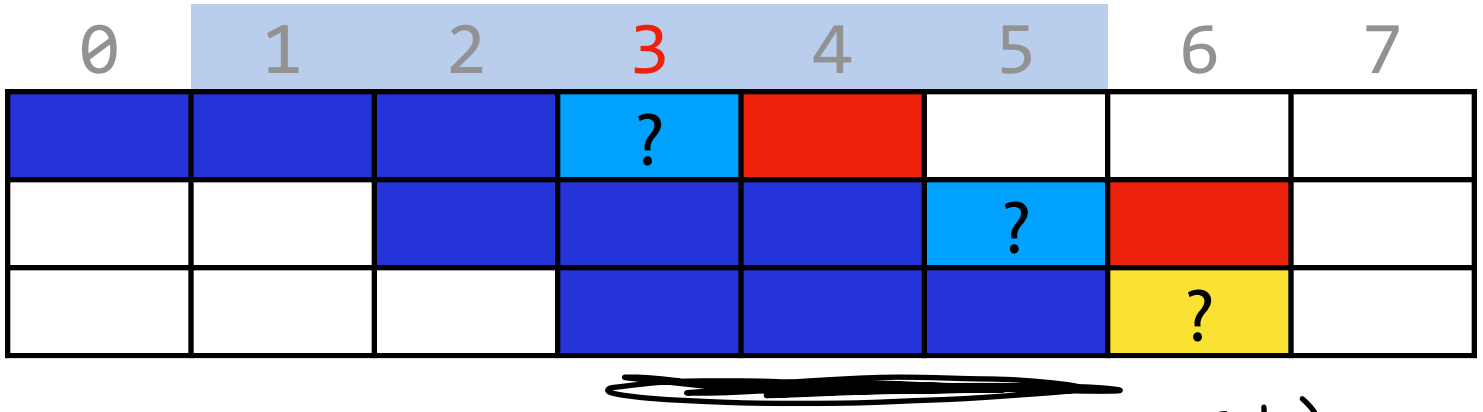


Case 2c: $i \leq r, Z_k > |\beta|$

Z_l tells us that our entire range (β included) matches earlier

The Z-Algorithm

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



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

β = 45 = 234 = 012

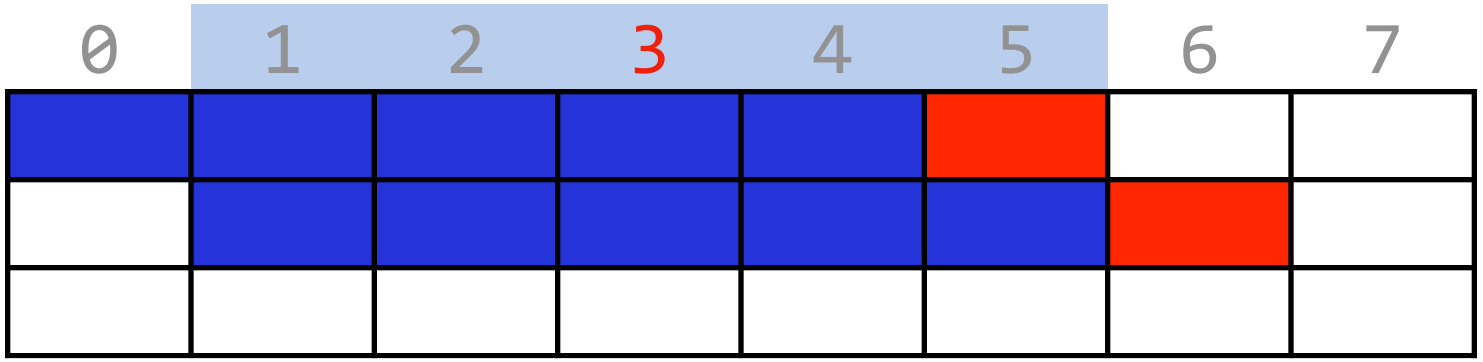
Z_l tells us that β matches earlier. Z_k tells us how much matches the prefix.

Does 6 match 5 or 3

What do we know about yellow?

The Z-Algorithm

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



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

01234 = 12345

5 ≠ 6

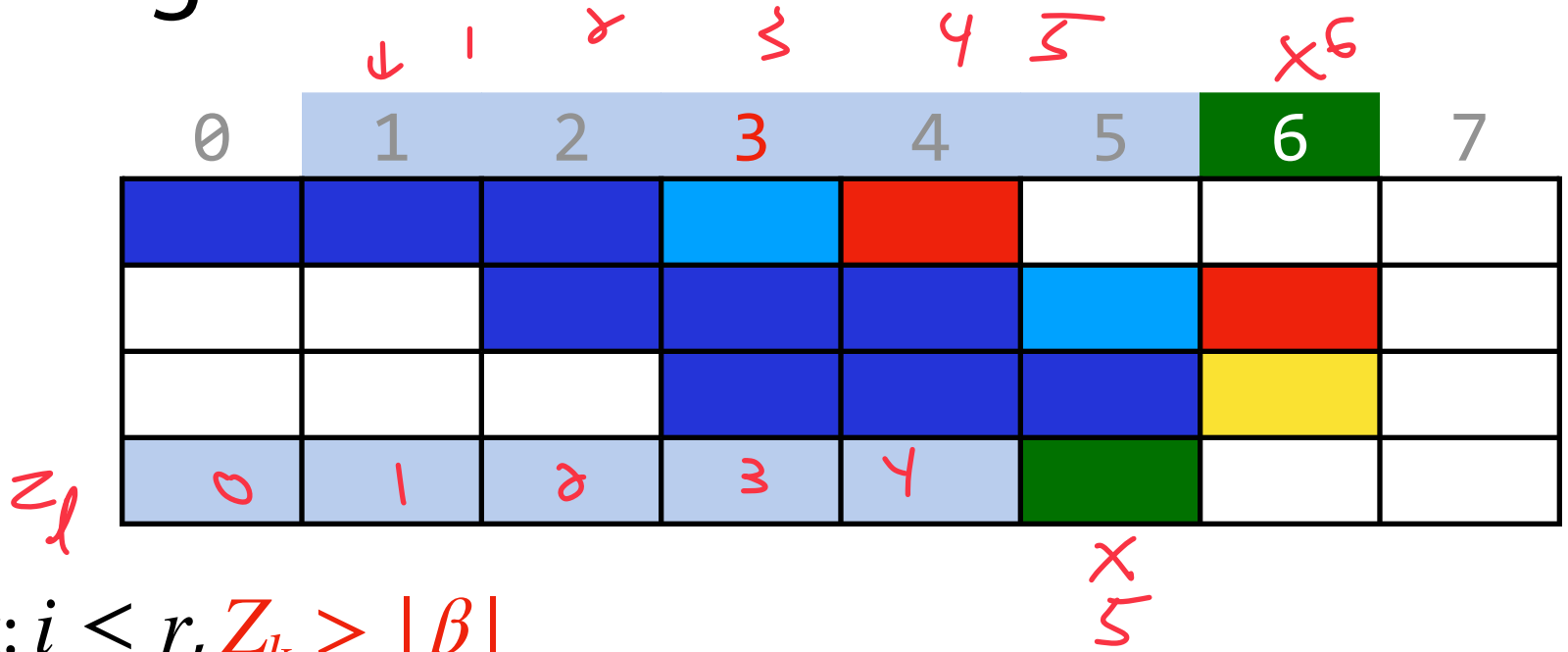
Z_l tells us that our entire range (β included) matches earlier

… and that it failed to match the next character.

are not equal!

The Z-Algorithm

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



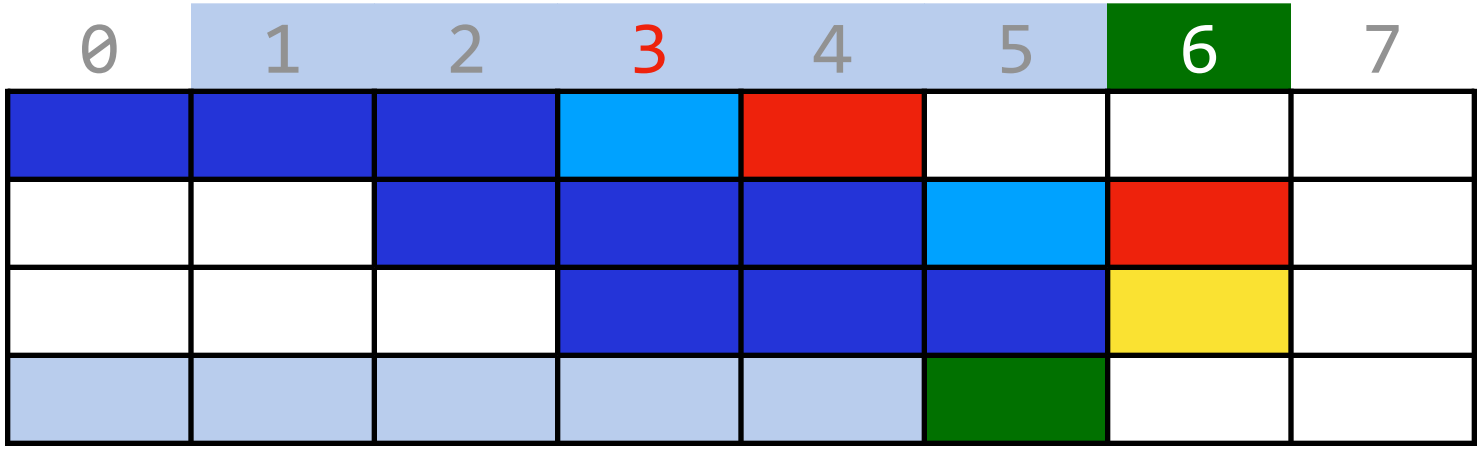
Case 2c: $i \leq r, Z_k > |\beta|$

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*(indices $r + 1$ 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

Demonstrate use of Z-algorithm in pattern matching

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.