Exact Pattern Matching with 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 \(S\).

\[
\begin{align*}
S: & \quad A B C D A B C D A B & Z_4(S) = \\
\end{align*}
\]
The Z-Algorithm

$ 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
 1 0 1 1
 0 1 1
 0 1 1
 1 1
 1
The Z-Algorithm

$Z_1 = 3$

$Z_2 =$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>A</td>
<td>B</td>
<td></td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
The Z-Algorithm

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
The Z-Algorithm

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
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 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 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

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 $Z_6$, we can save time by looking up the value _____________
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 ____________

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

The Z-Algorithm
The Z-Algorithm

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

Compute \( Z_i \) using \( i r l \):

Case 1 \((i > r)\): Compute explicitly; update \( i r l \)

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

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

The amount of work required depends on two pieces of information:

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

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

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 $i, r, l$?
The Z-Algorithm

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 $i, r, l$?
The Z-Algorithm

\[ i = 6, r = 7, l = 5 \]

\[ Z_k = Z_1 = 3 \]

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 \( i \)

\[ k = 6 - 5 = 1 \]
Case 2a: $i \leq r, Z_k < |\beta|$  

$|\beta| = \underline{\underline{\underline{\underline{}}}}, k = \underline{\underline{\underline{\underline{}}}}, Z_k = \underline{\underline{\underline{\underline{}}}}$  

$Z_i = \underline{\underline{\underline{\underline{}}}}$
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 \( \beta \) matches earlier.
The Z-Algorithm

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

$Z_l$ tells us that $\beta$ matches earlier. $Z_k$ tells us how much matches the prefix.
The Z-Algorithm

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

\( Z_l \) tells us that \( \beta \) matches earlier. \( Z_k \) tells us how much matches the prefix.

Because \( Z_k < |\beta| \), \( Z_i = \) __________
The Z-Algorithm

\[ i = 4, r = 4, l = 3 \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

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

\[ |\beta| = \text{__________}, k = \text{__________}, Z_k = \text{__________} \]

\[ Z_i = \text{__________} \]
The Z-Algorithm

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

$Z_l$ (defined by $r, l$) tells us that $\beta$ matches earlier.

\[ i = 4, r = 4, l = 3 \]
The Z-Algorithm

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

$Z_l$ (defined by $r, l$) tells us that $\beta$ matches earlier.

$Z_k$ tells us how much matches the prefix... but not everything!
Case 2b: $i \leq r, Z_k = |\beta|$

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

Because $Z_k = |\beta|$, $Z_i = \underline{__________}$
The Z-Algorithm

\[ i = 3, r = 5, l = 1 \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

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

\[ |\beta| = \underline{\text{________}}, k = \underline{\text{________}}, Z_k = \underline{\text{________}} \]

\[ Z_i = \underline{\text{________}} \]
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

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

$Z_l$ tells us that $\beta$ matches earlier. $Z_k$ tells us how much matches the prefix.

What do we know about yellow?
The Z-Algorithm

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

$Z_l$ tells us that our entire range ($\beta$ included) matches earlier.

... and that it failed to match the next character.
Case 2c: \( i \leq r, Z_k > |\beta| \)

The Z-Algorithm also tells us that yellow and green can't be equal!
Case 2c: $i \leq r, Z_k > |\beta|$

$Z_l$ tells us that $\beta$ is our prefix. $Z_k$ is also a previously computed prefix.

Because $Z_k > |\beta|$, $Z_i = \text{__________}$
The Z-Algorithm

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

Compute $Z_i$ using $i_{rl}$:

Case 1 ($i > r$): Compute explicitly; update $i_{rl}$

Case 2 ($i \leq r$):

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

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

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.