String Algorithms and Data Structures
Z-values and the Z-algorithm
CS 199-225
Brad Solomon
February 7, 2022
A naive reflection

Time

Lecture Helpfulness

Material Understood

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 \textit{w/} Z-algorithm

\textit{Pattern, }P \quad \textit{Text, }T

Naive \approx \theta(|P| + |T|)

Z-Algorithm \approx \theta(|P| + |T|)

Find instances of \textit{P} in \textit{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 \(S\).

\[
\begin{array}{ccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{array}
\]

\(Z_4(S) = \) 

\[
\begin{array}{ccccccccccc}
\end{array}
\]

\(Z_5(S) = 3\)

\[
\begin{array}{ccccccccccc}
\end{array}
\]

\(Z_1(S) = 7\)
Z-value Pattern Matching

\[ P: \quad T \ T \quad T: \quad C \ T \ T \ A \]

\[ S: \quad T \ T \$ \ C \ T \ T \ A \]

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

Z-value search pseudo-code

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

2. Calculate Z-values for \( S \)

3. For \( i < 0 \), match if \( Z_i = |P| \)

Match is \textit{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!

<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><strong>X</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( S: \) TT C G T T A G C G

\[ Z_4 = 2 \]
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 $S[i]$  

<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>$Z_1$</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>
<tr>
<td></td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

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

<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>$Z_2$</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>
<tr>
<td></td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>


The Z-Algorithm

\[ Z_1 = 3 \]
\[ Z_2 = ? \]

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

\( i, \) the current index = 

\( r, \) the furthest match char = 

\( l, \) the furthest reaching Z-value =

<table>
<thead>
<tr>
<th></th>
<th></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>0</td>
<td>1</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>
The Z-Algorithm

\( i \), the current index =

\( r \), the furthest match char =

\( l \), the furthest reaching Z-value =

<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>0</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
</tbody>
</table>
The Z-Algorithm

- The current index $i =$
- The furthest match char $r =$
- The furthest reaching Z-value $l =$
The Z-Algorithm

\( i, \) the current index = 

\( r, \) the furthest match char = 

\( l, \) the furthest reaching Z-value =
The Z-Algorithm

\[ i, \text{ the current index } = \]

\[ r, \text{ the furthest match char } = \]

\[ l, \text{ the furthest reaching Z-value } = \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>3</th>
<th>1</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>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A</th>
<th>B</th>
<th>B</th>
<th>A</th>
<th>A</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>
The Z-Algorithm

\[ i, \text{ the current index} = \]

\[ r, \text{ the furthest match char} = \]

\[ l, \text{ the furthest reaching Z-value} = \]
The Z-Algorithm

\(i\), the current index = 

\(r\), the furthest match char = 

\(l\), the furthest reaching Z-value = 

\[
\begin{array}{cccccccc}
- & 1 & 0 & 0 & 3 & 1 & 0 & 1 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\text{A} & \text{A} & \text{B} & \text{B} & \text{A} & \text{A} & \text{B} & \text{A} \\
\text{A} & \text{A} & \text{B} & \text{B} & \text{A} & \text{A} & \text{B} & \text{A} \\
\end{array}
\]
The Z-Algorithm

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

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 Z-Algorithm

<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>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

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

<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>B</td>
<td>C</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

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

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 $i\text{rl}$:

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

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

\[ i = 5, r = 7, l = 4 \]

<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>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>

Let \( \beta \) 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

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

<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>B</td>
<td>C</td>
<td>D</td>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>

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

$|\beta| = \text{___________}, k = \text{___________}, Z_k = \text{___________}$

$Z_i = \text{___________}$
The Z-Algorithm

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

$Z_l$ 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.

$i = 5, r = 7, l = 4$
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 $Z_k < |\beta|$, $Z_i = \text{__________}$
The Z-Algorithm

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>3</th>
<th>4</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>A</td>
<td>B</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>
</tr>
</tbody>
</table>

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

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

$Z_i = \_\_\_\_\_\_$
The Z-Algorithm

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

$Z_l$ tells us that $\beta$ matches earlier.

$i = 5, r = 6, l = 4$
Case 2b: $i \leq r, Z_k = |\beta|$

$Z_l$ tells us that $\beta$ matches earlier.

$Z_k$ tells us how much matches the prefix… but not everything!
The Z-Algorithm

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 \textbf{Z-Algorithm}

\begin{center}
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
\hline
\hline
\end{tabular}
\end{center}

\textit{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 \( \beta \) matches earlier.
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?
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.

$Z_l$ also tells us that yellow and green can’t be equal!
Case 2c: \( i \leq r, Z_k > |\beta| \)

The \( Z \)-Algorithm tells us that \( \beta \) is our prefix. \( Z_k \) is also a previously computed prefix.

Because \( Z_k > |\beta|, Z_i = \underline{\text{_________}} \)
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 \((i \leq r)\):

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

2b: \((Z_k = |\beta|): Z_i = Z_k + explicit(r+1); update i r l\)

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.