Module 10: Compression - Enriched

T. Biedl  M. Petrick  O. Veksler
Based on lecture notes by many previous cs240 instructors

David R. Cheriton School of Computer Science, University of Waterloo

Winter 2019

References: wikipedia
Outline

1 Compression
Huffman with a different base

Example text: nobanana$, $\Sigma_S = \{\$, b, o, a, n\}$

Character frequencies: $\$: 1, $b$: 1, $o$: 1, $a$: 3, $n$: 3

![Huffman Tree](image)

$\left(\begin{array}{c}10010001011011010000\end{array}\right)_{20 \text{ bits}}$ vs. $(202011212100)_3 = \left(\begin{array}{c}11000000111110010110\end{array}\right)_{19 \text{ bits}}$
Arithmetic compression

This was skipped in W19 and you do not need to know it.

Subdivide \([0, 1]\) by frequencies of letters.

\[
\begin{array}{ccccccc}
0 & \frac{1}{3} & \frac{2}{3} & \frac{7}{9} & \frac{8}{9} & 1 \\
\hline
a \ (33\%) & n \ (33\%) & b \ (11\%) & o \ (11\%) & $ \ (11\%) \\
\end{array}
\]

divide each letter’s range in turn

\[
\begin{align*}
\frac{1}{3} & + \frac{7}{9} \cdot \frac{1}{3} \\ & \approx 0.593
\end{align*}
\]

\[
\begin{align*}
\frac{1}{3} & + \frac{8}{9} \cdot \frac{1}{3} \\ & \approx 0.630
\end{align*}
\]

\[
\begin{align*}
0.593 & + \frac{8}{9} \cdot 0.037 \\ & \approx 0.626
\end{align*}
\]

Any number in interval \((0.626, 0.63)\) represents ‘no$’.
Arithmetic compression code

**ArithmeticEncoding**\((f, S)\)
\(f\): frequencies of \(\Sigma_S\), \(S\): text to encode (ends with $)
1. \(I \leftarrow [0, 1] \quad //\) interval
2. \(\text{for } i = 0 \ldots |S| - 1\)
3. \(\text{subdivide } I \text{ by frequencies } f \text{ of } \Sigma_S\)
4. \(I \leftarrow \text{interval that corresponds to } S[i]\)
5. \(C \leftarrow \text{number in } I \text{ that needs few bits}\)
6. \(\text{return } C \text{ (or its binary encoding)}\)

**ArithmeticDecoding**\((f, C)\)
\(f\): frequencies of \(\Sigma_S\), \(C\): number to decode
1. \(S \leftarrow \text{empty string}, I \leftarrow [0, 1]\)
2. \(\text{repeat}\)
3. \(\text{subdivide } I \text{ by frequencies } f \text{ of } \Sigma_S\)
4. \(S.\text{append}(\text{character whose interval contains } C)\)
5. \(\text{until } c = $\)
6. \(\text{return } S\)
Move-To-Front Transform

- Keep alphabet $\Sigma_S$ in an unsorted array $L$ in pre-agreed order. (Typically $L$ is initialized as ASCII.)
- Encode each character of $S$ by its index in $L$.
- After such an encoding, $L$ is updated with Move-To-Front heuristic.

$$S = \text{GOOD}$$

$C = 1, 2, 0, 2$

- $S$ has substring $A...A \iff C$ has run $0...0$ of length $k - 1$.
- $C$ contains lots of small numbers and few big ones.
Fast Burrows-Wheeler Encoding

Text $S = \text{tarantula}$

Suffix array $A_s$:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$i$th suffix</th>
<th>$A_s$ suffix/cyclic shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\text{tarantula}$$</td>
<td>9 $\text{a}$ $\text{tarantula}$$</td>
</tr>
<tr>
<td>1</td>
<td>$\text{arantula}$$</td>
<td>8 $\text{l}$ $\text{tarantula}$$</td>
</tr>
<tr>
<td>2</td>
<td>$\text{rantula}$$</td>
<td>3 $\text{rantula}$ $\text{tara}$$</td>
</tr>
<tr>
<td>3</td>
<td>$\text{antula}$$</td>
<td>1 $\text{tarantula}$ $\text{tari}$$</td>
</tr>
<tr>
<td>4</td>
<td>$\text{ntula}$$</td>
<td>7 $\text{ula}$ $\text{tarantul}$$</td>
</tr>
<tr>
<td>5</td>
<td>$\text{tula}$$</td>
<td>4 $\text{antula}$ $\text{tarar}$$</td>
</tr>
<tr>
<td>6</td>
<td>$\text{ula}$$</td>
<td>2 $\text{rantula}$ $\text{tarat}$$</td>
</tr>
<tr>
<td>7</td>
<td>$\text{l}$$</td>
<td>6 $\text{tula}$ $\text{tarant}$$</td>
</tr>
<tr>
<td>8</td>
<td>$\text{a}$$</td>
<td>0 $\text{tarantula}$$</td>
</tr>
<tr>
<td>9</td>
<td>$$$</td>
<td>5 $\text{ntula}$ $\text{tarana}$$</td>
</tr>
</tbody>
</table>

$BWTFastEncoding(S)$

1. $A_s \leftarrow$ compute suffix array of $S$
2. for $i = 0 \ldots |S| - 1$
3. \hspace{1em} if $A_s[i] = 0$ do $C$.append($\$$)
4. \hspace{1em} else $C$.append($T[A_s[i] - 1]$)
5. return $C$