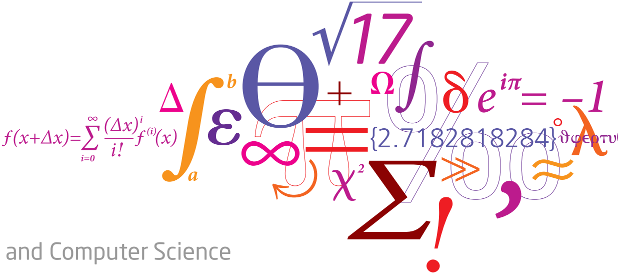
Compression

02282 Algorithms for Massive Data Sets

Patrick Hagge Cording



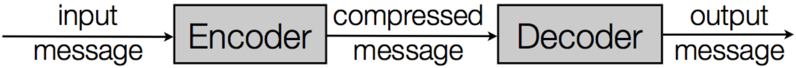
DTU Compute

Department of Applied Mathematics and Computer Science

Outline

- Introduction to compression
- Burrows-Wheeler Transform
- Lempel-Ziv compression
- Grammar-compression

Encoding and decoding



- Lossless: input message = output message.
- Lossy: input message ≈ output message.

How much can we compress?

One lossless compression scheme can not compress all messages.

- Consider all messages of length 2^n.
- Suppose all messages are encoded to $n-1$ or fewer bits.
- From $n-1$ bits, the decoder can distinguish at most 2^{n-1} messages.

If one message is compressed, others must expand.

Quality

Quality of compression usually measured by:

- Time used to compress/decompress
- Size of encoded message
- Generality of the technique
- Lossy compression: also quality of reconstructed approximation

Warm-up

Think of a compression scheme that compresses the string

`aaaaaabbbbcccccc`

Warm-up

Think of a compression scheme that compresses the string

`aaaaaabbbbccccc`

Run-length encoding:

$$(\mathtt{a}, 6)(\mathtt{b}, 4)(\mathtt{c}, 5)$$

How does run-length encoding perform on english text?

Burrows-Wheeler Transform (BWT)

- Idea: Group characters according to their *context*.
- The letter “t” often occurs followed by “he” in english.
- The BWT is reversible!

Algorithm

- Sort all cyclic rotations of S.
- Store the last character of each cyclic rotation.

BWT example

$S =$ bananas

Cyclic rotations	Sorted
bananas	ananasb
ananasb	anasban
nanasba	asbanan
anasban	bananas
nasbana	nanasba
asbanan	nasbana
sbanana	sbanana

$BWT(S) =$ bnnsaaa

Efficient computation of the BWT

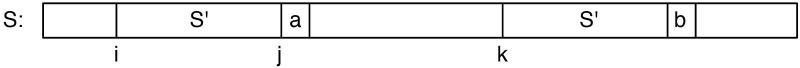
- Append special character (\$) to S.
- Sort the suffixes by constructing a suffix tree or suffix array.

Suffixes	Sorted	Sorted order
`bananas$`	`ananas$`	2
`ananas$`	`anas$`	4
`nanas$`	`as$`	6
`anas$`	`bananas$`	1
`nas$`	`nanas$`	3
`as$`	`nas$`	5
`s$`	`s$`	7
`$`	`$`	8

- Let $s_1 s_2 \dots s_n$ be the sorted order of suffixes.
- $BWT(S) = S[s_1 - 1]S[s_2 - 1] \dots S[s_n - 1]$
- $O(n)$ time and space.

LZ77

- Encode substrings as references to previously seen substrings.



- The LZ77 parse of a string S is a sequence of triplets $(p_1, n_1, c_1)(p_2, n_2, c_2)\dots(p_z, n_z, c_z)$ where
  - p_k is a position in S,
  - n_k is the length,
  - and c_k is a single character.
- For any $1 \leq k \leq z$: $p_k + n_k \leq k - 1 + \sum_{i=1}^{k-1} n_i$.

LZ77 encoding algorithm

- $p = 1$
- While $p \leq n$
  - Let $S[i..j]$ be the longest substring of $S[1..p-1]$ s.t. $S[i..j] = S[p..p+(j-i)]$
  - If $S[i..j] \neq \varepsilon$ then output $(i, j-i, S[p+(j-i)+1])$ otherwise output $(-,-,S[p+(j-i)+1])$
  - Set $p = p + (j-i) + 2$

Efficient implementation

- Use suffix tree for step 1.
- $O(n)$ time and space.

LZ78

- The LZ78 parse of a string S is a sequence of pairs (phrases) $(r_1, c_1)(r_2, c_2) \dots (r_z, c_z)$ where
  - r_k is a pointer to an earlier phrase,
  - and c_k is a single character.
- Differ from LZ77 in the way it finds matches.

LZ78 encoding algorithm

The algorithm uses a trie to represent the pairs. Nodes are labelled by phrase numbers and edges by characters.

- Let T be the LZ78 trie, initially having just one node.
- $p = 1$
- $k = 1$
- While $p \leq n$
    - Let $S[p..j]$ be the longest prefix of $S[p..n]$ that is also a string in T
    - Update T to contain $S[p..j+1]$. Insert new node with label k and label its ingoing edge $S[j+1]$
    - Set $p = j + 2$
    - Set $k = k + 1$

Algorithm runs in $O(n)$ time and $O(z)$ space.

Summary of LZ77 and LZ78

- Used in gzip, png, rar, zip, and gif.
- Many variants:
    - Encode phrases.
    - Sliding window.
    - Non-greedy.
    - Self-referential.

- Greedy LZ77 parse is optimal w.r.t. number of phrases but not w.r.t. total number of bits required to encoded phrases.

Grammar compression

Straight Line Program (SLP)

- Context-free grammar in Chomsky normal form:
    - All production rules have the form $X = YZ$ or $X = c$.
- Generates one string only.

- In compression: redundancies are replaced by production rules.

Grammar-compression example

`abaababaabaababaababaabaab`

Grammar-compression example

$\underline{\texttt{ab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{ab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{ab}}\texttt{a}\underline{\texttt{ab}}$ $X_1 = \texttt{ab}$

Grammar-compression example

$\underline{\texttt{ab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{ab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{ab}}\texttt{a}\underline{\texttt{ab}}$ $X_1 = \texttt{ab}$

$X_1\texttt{a}X_1X_1\texttt{a}X_1\texttt{a}X_1X_1\texttt{a}X_1X_1\texttt{a}X_1\texttt{a}X_1$

Grammar-compression example

$\underline{\texttt{ab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{ab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{ab}}\texttt{a}\underline{\texttt{ab}}$ $\qquad X_1 = \texttt{ab}$

$\underline{X_1\texttt{a}}X_1\underline{X_1\texttt{a}X_1\texttt{a}}X_1\underline{X_1\texttt{a}}X_1\underline{X_1\texttt{a}X_1\texttt{a}}X_1$ $\qquad X_2 = X_1\texttt{a}$

Grammar-compression example

$\underline{\texttt{ab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{ab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{ab}}\texttt{a}\underline{\texttt{ab}}$ $\qquad X_1 = \texttt{ab}$

$\underline{X_1\texttt{a}}X_1\underline{X_1\texttt{a}X_1\texttt{a}}X_1\underline{X_1\texttt{a}}X_1\underline{X_1\texttt{a}X_1\texttt{a}}X_1$ $\qquad X_2 = X_1\texttt{a}$

$X_2X_1X_2X_2X_1X_2X_1X_2X_2X_1$

Grammar-compression example

$$\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{ab}} \qquad X_1 = \mathtt{ab}$$

$$\underline{X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}X_1\mathtt{a}}X_1 \qquad X_2 = X_1\mathtt{a}$$

$$\underline{X_2X_1}X_2\underline{X_2X_1X_2X_1}X_2\underline{X_2X_1} \qquad X_3 = X_2X_1$$

Grammar-compression example

$\underline{\texttt{ab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{ab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{ab}}\texttt{a}\underline{\texttt{ab}}$ $\quad X_1 = \texttt{ab}$

$\underline{X_1\texttt{a}}X_1\underline{X_1\texttt{a}X_1\texttt{a}}X_1\underline{X_1\texttt{a}}X_1\underline{X_1\texttt{a}X_1\texttt{a}}X_1$ $\quad X_2 = X_1\texttt{a}$

$\underline{X_2X_1}X_2\underline{X_2X_1X_2X_1}X_2\underline{X_2X_1}$ $\quad X_3 = X_2X_1$

$X_3X_2X_3X_3X_2X_3$

Grammar-compression example

$$\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{ab}}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{ab}}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{ab}}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{ab}} \qquad X_1 = \mathtt{ab}$$

$$\underline{X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}}\underline{X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}}\underline{X_1\mathtt{a}}X_1 \qquad X_2 = X_1\mathtt{a}$$

$$\underline{X_2X_1}X_2\underline{X_2X_1}\underline{X_2X_1}X_2\underline{X_2X_1} \qquad X_3 = X_2X_1$$

$$\underline{X_3X_2}X_3\underline{X_3X_2}X_3 \qquad X_4 = X_3X_2$$

Grammar-compression example

$$\underline{\texttt{ab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{ab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{ab}}\texttt{a}\underline{\texttt{ab}} \qquad X_1 = \texttt{ab}$$

$$\underline{X_1\texttt{a}}X_1\underline{X_1\texttt{a}X_1\texttt{a}}X_1\underline{X_1\texttt{a}}X_1\underline{X_1\texttt{a}X_1\texttt{a}}X_1 \qquad X_2 = X_1\texttt{a}$$

$$\underline{X_2X_1}X_2\underline{X_2X_1X_2X_1}X_2\underline{X_2X_1} \qquad X_3 = X_2X_1$$

$$\underline{X_3X_2}X_3\underline{X_3X_2}X_3 \qquad X_4 = X_3X_2$$

$$X_4X_3X_4X_3$$

Grammar-compression example

$$\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{ab}} \qquad X_1 = \mathtt{ab}$$

$$\underline{X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}X_1\mathtt{a}}X_1 \qquad X_2 = X_1\mathtt{a}$$

$$\underline{X_2X_1}X_2\underline{X_2X_1X_2X_1}X_2\underline{X_2X_1} \qquad X_3 = X_2X_1$$

$$\underline{X_3X_2}X_3\underline{X_3X_2}X_3 \qquad X_4 = X_3X_2$$

$$\underline{X_4X_3X_4X_3} \qquad X_5 = X_4X_3$$

Grammar-compression example

$$\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{ab}} \qquad X_1 = \mathtt{ab}$$

$$\underline{X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}X_1\mathtt{a}}X_1 \qquad X_2 = X_1\mathtt{a}$$

$$\underline{X_2X_1}X_2\underline{X_2X_1X_2X_1}X_2\underline{X_2X_1} \qquad X_3 = X_2X_1$$

$$\underline{X_3X_2}X_3\underline{X_3X_2}X_3 \qquad X_4 = X_3X_2$$

$$\underline{X_4X_3X_4X_3} \qquad X_5 = X_4X_3$$

$$X_5X_5$$

Grammar-compression example

$\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{ab}}$ $X_1 = \mathtt{ab}$

$\underline{X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}X_1\mathtt{a}}X_1$ $X_2 = X_1\mathtt{a}$

$\underline{X_2X_1}X_2\underline{X_2X_1X_2X_1}X_2\underline{X_2X_1}$ $X_3 = X_2X_1$

$\underline{X_3X_2}X_3\underline{X_3X_2}X_3$ $X_4 = X_3X_2$

$\underline{X_4X_3X_4X_3}$ $X_5 = X_4X_3$

$\underline{X_5X_5}$ $X_6 = X_5X_5$

Grammar-compression example

$$\underline{\texttt{ab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{ab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{abab}}\texttt{a}\underline{\texttt{ab}}\texttt{a}\underline{\texttt{ab}} \qquad X_1 = \texttt{ab}$$

$$\underline{X_1\texttt{a}}X_1\underline{X_1\texttt{a}X_1\texttt{a}}X_1\underline{X_1\texttt{a}}X_1\underline{X_1\texttt{a}X_1\texttt{a}}X_1 \qquad X_2 = X_1\texttt{a}$$

$$\underline{X_2X_1}X_2\underline{X_2X_1X_2X_1}X_2\underline{X_2X_1} \qquad X_3 = X_2X_1$$

$$\underline{X_3X_2}X_3\underline{X_3X_2}X_3 \qquad X_4 = X_3X_2$$

$$\underline{X_4X_3X_4X_3} \qquad X_5 = X_4X_3$$

$$\underline{X_5X_5} \qquad X_6 = X_5X_5$$

$$X_6$$

Grammar-compression example

$$\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{abab}}\mathtt{a}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{ab}} \qquad X_1 = \mathtt{ab}$$

$$\underline{X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}}X_1\underline{X_1\mathtt{a}X_1\mathtt{a}}X_1 \qquad X_2 = X_1\mathtt{a}$$

$$\underline{X_2X_1}X_2\underline{X_2X_1X_2X_1}X_2\underline{X_2X_1} \qquad X_3 = X_2X_1$$

$$\underline{X_3X_2}X_3\underline{X_3X_2}X_3 \qquad X_4 = X_3X_2$$

$$\underline{X_4X_3X_4X_3} \qquad X_5 = X_4X_3$$

$$\underline{X_5X_5} \qquad X_6 = X_5X_5$$

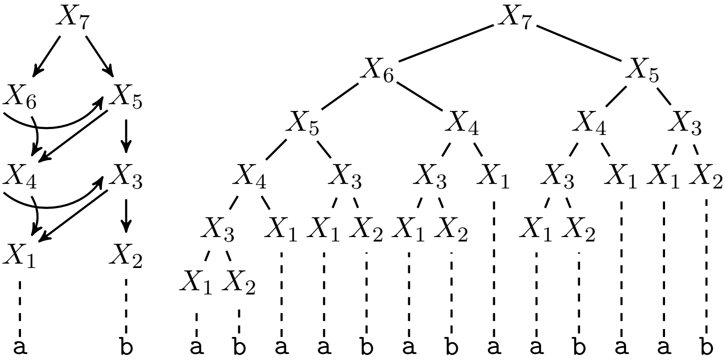
$$X_6$$

- Original data: 26 characters. Compressed data: 6 rules.

Representation and decompression

$S =$ `abaababaabaab`

$X_1 = \texttt{a}$
$X_2 = \texttt{b}$
$X_3 = X_1X_2$
$X_4 = X_3X_1$
$X_5 = X_4X_3$
$X_6 = X_5X_4$
$X_7 = X_6X_5$



(Left) the grammar,
(center) Directed acyclic graph (DAG) representation,
(right) the parse tree.

Straight Line Programs

Straight Line Programs are mainly of theoretical interest.

- Efficient computation: Smallest grammar problem is NP-hard.
- We can convert an LZ77 parse of size z to an SLP of size $O(z \log N/z)$.
- We can convert an LZ78 parse of size z to an SLP of size $O(z)$ (exercise).
- A data structure for an SLP is also a data structure an LZ77 or LZ78 compressed string (with some overhead).

Be sure you understand Straight Line Programs – next time we design data structures for them!