CS 240 – Data Structures and Data Management

Module 10: Compression - Enriched

T. Biedl E. Kondratovsky M. Petrick O. Veksler

Based on lecture notes by many previous cs240 instructors

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

Winter 2022

version 2022–03–24 10:54

Outline

1. Compression
   - Compression ratio
   - Huffman encoding
   - Modified run-length encoding

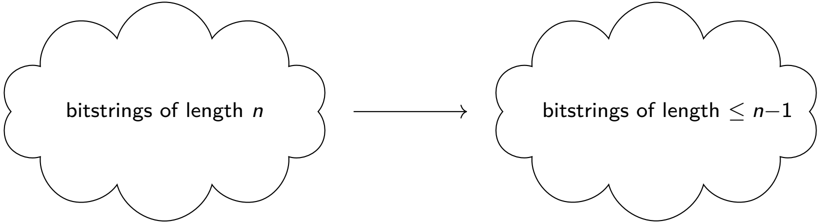
Outline

1. Compression
   - Compression ratio
   - Huffman encoding
   - Modified run-length encoding

Compression ratio

Theorem: No compression algorithm can have compression ratio < 1 for *all* input strings.

Proof: Assume $\Sigma_S = \Sigma_C = \{0, 1\}$.



- How big are these sets?

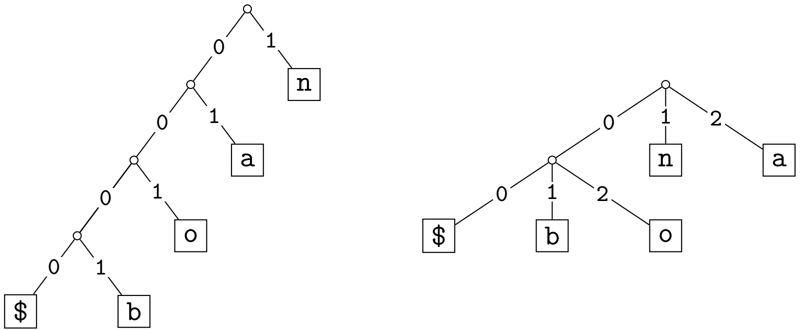
Outline

1. Compression
   - Compression ratio
   - Huffman encoding
   - Modified run-length encoding

Huffman with a different base

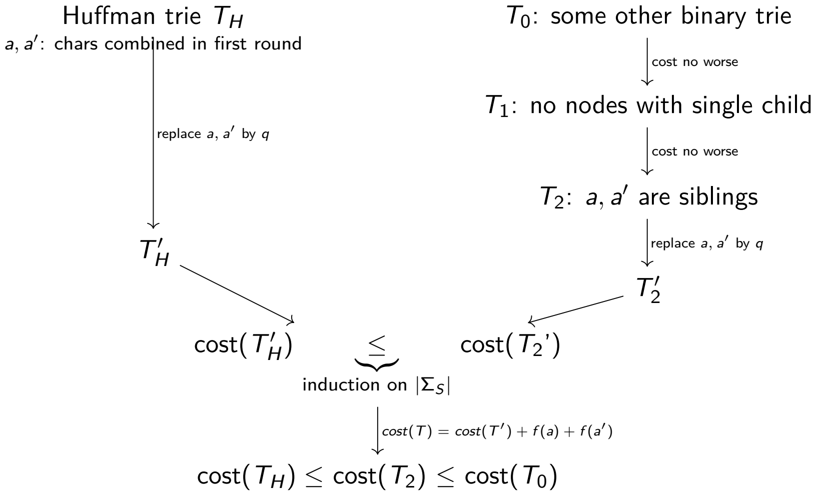
Example text: `nobanana$`, $\Sigma_S = \{\$, b, o, a, n\}$

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

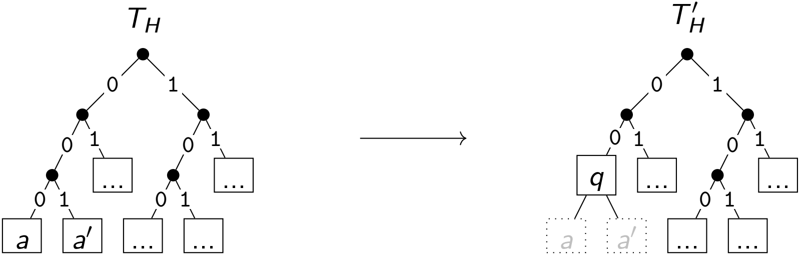


$$\left(\underbrace{10010001011011010000}_{20\ \text{bits}}\right) \text{vs.} (202011212100)_3 = \left(\underbrace{1100000111110010110}_{19\ \text{bits}}\right)$$

Huffman optimality – outline



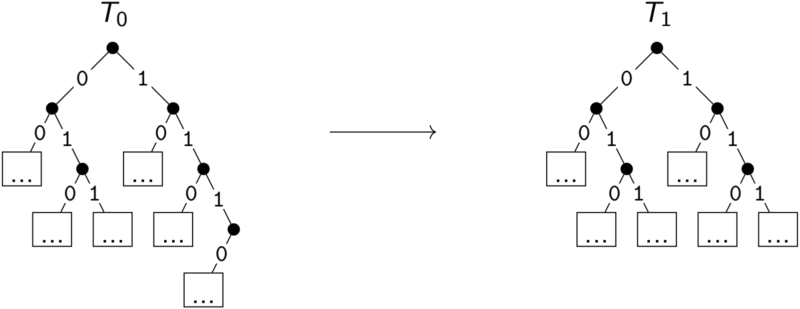
Huffman optimality – from T_H to T'_H



Observation: $cost(T_H) = cost(T'_H) + f(a) + f(a')$

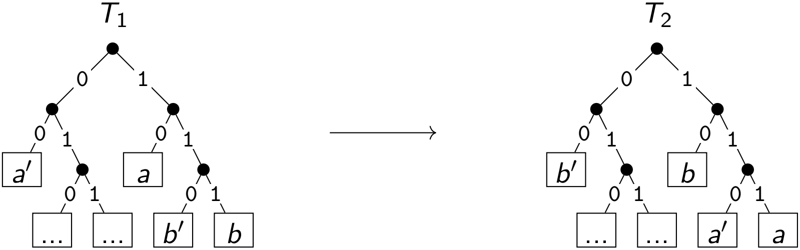
Observation: T'_H is the Huffman-encoding trie for $\Sigma_S \setminus \{a, a'\} \cup \{q\}$

Huffman optimality – from T_0 to T_1



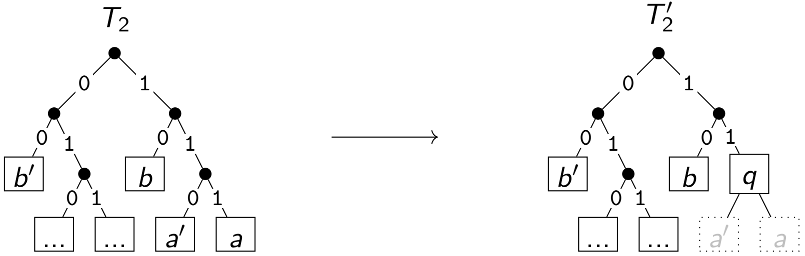
Observation: $cost(T_1) \leq cost(T_0)$

Huffman optimality – from T_1 to T_2



Observation: $cost(T_2) \leq cost(T_1)$

Huffman optimality – from T_2 to T_2'



Observation: $cost(T_2) = cost(T_2') + f(a) + f(a')$

Outline

1. Compression
   - Compression ratio
   - Huffman encoding
   - Modified run-length encoding

Modified run-length encoding

Binary bijective numeration (use here $\{A', B'\}$ rather than $\{0, 1\}$):

k	1	2	3	4	5	6	7	8	...
$E(k)$	A'	B'	$A'A'$	$A'B'$	$B'A'$	$B'B'$	$A'A'A'$	$A'A'B'$	...

- $E(k)$ has length $\lfloor \log(k+1) \rfloor$
- Base-2 representation has length $\lfloor \log k \rfloor + 1$, so this is (slightly) shorter.

Modified RLE: Encode only runs of 0, using binary bijection numeration:

Example:

$$S = 110, 114, 100, \underbrace{0, 0, 0, 0}_{4}, 1, 6, 100, 2, \underbrace{0, 0}_{2}, 103, 2, \underbrace{0, 0, 0, 0}_{4}$$

$$C = 110, 114, 100, A', B', 1, 6, 100, 2, B', 103, 2, A', B'$$