UNIVERSITÄT
LEIPZIG

Asymmetric Numeral System

January 7, 2025

Huo Jiang

Fakultät für Mathematik und Informatik
Universität Leipzig

Outline

1. Introduction
2. Asymmetric Numeral System
3. rANS
4. Streaming-rANS
5. Summary
6. Further Study
7. Reference

Introduction

Outline

1. Introduction
   1.1 Information Theory
   1.2 Source Coding Theorem
   1.3 Entropy Coding

Information Theory[1]

- 1948 Claude Shannon: ”A Mathematical Theory of Communication”
- Self-information
  a symbol of probability p contains $log_2 \frac{1}{p}$ bits of information.
- Shannon Entropy
  Let X be a random variable, which takes values in the set $\mathcal{X}$ and is distributed according to p

$$H(X) = \sum_{x \in \mathcal{X}} p(x) log_2 \frac{1}{p(x)}$$

Source Coding Theorem (Shannon) [1]

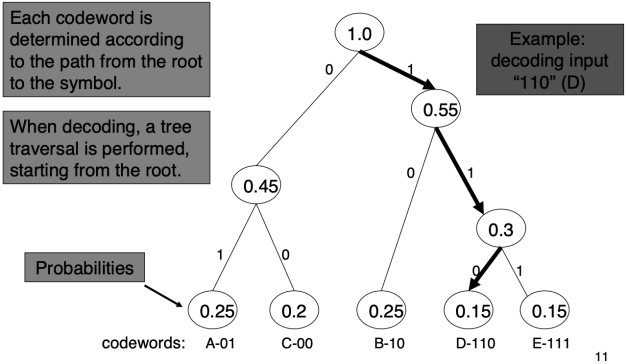
- prerequisite: lossless data compression
- $H(X)$ is compression limit, i.e. lower bound $L_{\text{avg}} \geq H(X)$
  which measures **the minimum average number of bits** needed to encode **a symbol** from a data source
- $H(X)$ is achievable, $L_{\text{avg}} \to H(X)$
  for a sufficiently long sequence of symbols it is possible to achieve an average code length close to $H(X)$
- Shannon's Source Coding Theorem [1]

$$N \cdot H(X)$$

Entropy Coding

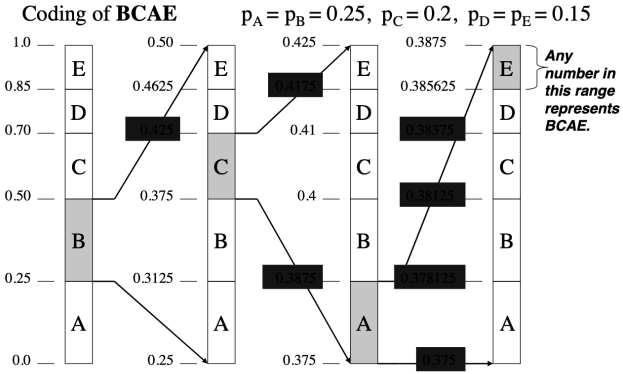
- Implementation of Shannon's Source Coding Theorem
- High-probability symbols get shorter codes
- Low-probability symbols get longer codes.
- A method that attempts to approach the lower bound $H(X)$
- Entropy Coding Family
  - Huffman Coding: ZIP, JPEG, PNG
  - Arithmetic Coding/Range Coding: JPEG, H.264, HEVC.
  - Asymmetric Numeral System (ANS): Zstandard, JPEG XL, AVIF

Huffman Coding



source: http://www.math.tau.ac.il/ dcor/Graphics/adv-slides/entropy.pdf

Arithmetic Coding / Range Coding



source: http://www.math.tau.ac.il/ dcor/Graphics/adv-slides/entropy.pdf

Asymmetric Numeral System

Outline

2. Asymmetric Numeral System
- 2.1 Motivation
- 2.2 Key Concepts
  - 2.2.1 Symbol spread function
  - 2.2.2 Representation of information
  - 2.2.3 Symmetric or Asymmetric

Motivation

- Huffman coding is efficient for symbol-based compression but cannot achieve optimal compression in many cases due to its restriction to integer-length codes
- Arithmetic coding achieves near-optimal compression but has a higher computational cost, especially in encoding and decoding (it uses multiplication and division operations for each encoded symbol)
- ANS combines the compression ratio of arithmetic coding (integer arithmetic, modular arithmetic, and table lookups), with a processing cost similar to that of Huffman coding [4]

Motivation

	Huffman Coding	**Arithmetic Coding**	**ANS**
Compression Efficiency	Suboptimal: Restricted to integer-length codes, resulting in slightly larger output than $H(X)$	Near-optimal: Matches $H(X)$ with fractional-length codes.	Near-optimal: Matches $H(X)$ with fine-grained fractional codes.
Arithmetic Operations	Simple: Uses addition and binary tree traversal.	Complex: Requires interval management with high precision.	Simple: Uses integer arithmetic and avoids floating-point operations.
Encoding/Decoding Speed	Fast: Uses binary trees for encoding/decoding (logarithmic complexity).	Slower: Requires multiplication/division operations.	Faster: Relies on modular arithmetic and table lookups.

Table 1

Key Concepts

- **Symbol spread function** → ”Numeral System”, e.g. standard binary system

Key Concepts

- **Symbol spread function** → ”Numeral System”, e.g. standard binary system
- **Representation of information**, e.g. standard decimal system represents an entire message as a single number

Key Concepts

- **Symbol spread function** → ”Numeral System”, e.g. standard binary system
- **Representation of information**, e.g. standard decimal system represents an entire message as a single number
- **Symmetric or ”Asymmetric”**

Symbol spread function

- the function that determines the split of the $\mathbb{N}$ subsets corresponding to different symbols [3]

Symbol spread function

- the function that determines the split of the $\mathbb{N}$ subsets corresponding to different symbols [3]
- $\bar{s} : \mathbb{N} \rightarrow \mathcal{A}, \bar{s}(x) = \text{mod}(x, b) = s$

Symbol spread function

- the function that determines the split of the $\mathbb{N}$ subsets corresponding to different symbols [3]
- $\bar{s} : \mathbb{N} \rightarrow \mathcal{A}, \bar{s}(x) = \text{mod}(x, b) = s$
- In the standard binary system example: $\bar{s}(x) = \text{mod}(x, 2)$
  i.e. splits natural numbers into subsets of even/odd numbers [3]

Symbol spread function (Exa. 1)

$\mathcal{A} = \{0, 1\}$, $\bar{s}(x) = \text{mod}(x, 2)$

Exa. 1: Standard Binary System

$\mathbb{N}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$s = 0$																			
$s = 1$																			

Fig.1

Symbol spread function (Exa. 2)

$\mathcal{A} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$x_{i+1} = 10x_i + s$

$\mathbb{N}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...	202	...	2025
$s = 0$																									
$s = 1$																									
$s = 2$																									
$s = 3$																									
$s = 4$																									
$s = 5$																									
$s = 6$																									
$s = 7$																									
$s = 8$																									
$s = 9$																									

Fig.2

Representation of information (Exa. 2)

- a finite sequence of symbols/digits:

$$\mathcal{A} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Representation of information (Exa. 2)

- a finite sequence of symbols/digits:

$$\mathcal{A} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

- the sequence of symbols/digits to be encoded:

$$S_{in} = (2, 0, 2, 5)$$

Representation of information (Exa. 2)

- a finite sequence of symbols/digits:

$$\mathcal{A} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

- the sequence of symbols/digits to be encoded:

$$S_{in} = (2, 0, 2, 5)$$

- How can we represent this sequence as a single number?

Representation of information (Exa. 2)

- a finite sequence of symbols/digits:

$$\mathcal{A} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

- the sequence of symbols/digits to be encoded:

$$S_{in} = (2, 0, 2, 5)$$

- How can we represent this sequence as a single number?
- The easiest way is to concatenate the symbols/digits:

$$x_{out} = 2025$$

Representation of Information (Exa. 2)

- encoding the sequence of digits $S_{in} = (2, 0, 2, 5)$

$$
\begin{aligned}
x_{\text{init}} &= 0 \\
x_1 &= x_{\text{init}} \times 10 + 2 = 2 \\
x_2 &= x_1 \times 10 + 0 = 20 \\
x_3 &= x_2 \times 10 + 2 = 202 \\
x_{\text{out}} &= x_3 \times 10 + 5 = 2025
\end{aligned}
$$

Representation of Information (Exa. 2)

– What is the encoding function?

Representation of Information (Exa. 2)

- What is the encoding function?
- Encoding function

$$x_{i+1} = x_i \times 10 + s_i$$

Representation of Information (Exa. 2)

- What is the encoding function?
- Encoding function

$$x_{i+1} = x_i \times 10 + s_i$$

- x_{i+1} is the x_i -th appearance of symbol s [3]

$$x_{i+1} = C(s, x_i) = x_i \cdot 10 + s$$

Representation of Information (Exa. 2)

$$x_{i+1} = 10x_i + s$$

$s = 0$, $s = 2$, $s = 2$, $s = 5$

x_{i+1} $\mathbb{N}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	202	2025
x_i $s = 0$	0										1										2		
$s = 1$		0										1											
$s = 2$			0										1										
$s = 3$				0										1									
$s = 4$					0										1						...		...
$s = 5$						0										1							
$s = 6$							0										1						
$s = 7$								0										1					
$s = 8$									0										1				
$s = 9$										0										1			

Fig.3

Representation of Information (Exa. 1)

Exa. 1: Standard Binary System

$$x_{i+1} = 2x_i + s$$

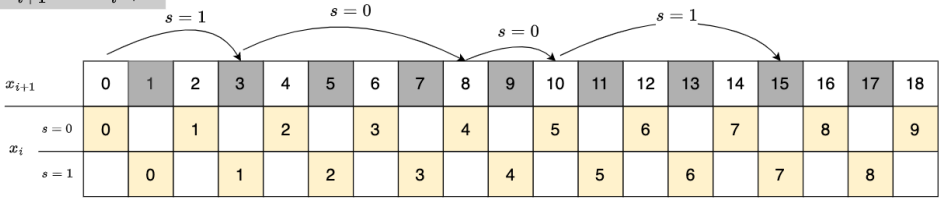


Fig.4

$x_{i+1} = 2x_i + s$

$s = 1$ $s = 0$ $s = 0$ $s = 1$

x_{i+1}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
x_i, $s = 0$	0		1		2		3		4		5		6		7		8		9
x_i, $s = 1$		0		1		2		3		4		5		6		7		8	

$x_{i+1} = 10x_i + s$

$s = 2$ $s = 0$ $s = 2$ $s = 5$

x_{i+1}	$\mathbb{N}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20		202		2025
x_i, $s = 0$		0										1										2				
x_i, $s = 1$			0										1													
x_i, $s = 2$				0										1												
x_i, $s = 3$					0										1											
x_i, $s = 4$						0										1							...		...	
x_i, $s = 5$							0										1									
x_i, $s = 6$								0										1								
x_i, $s = 7$									0										1							
x_i, $s = 8$										0										1						
x_i, $s = 9$											0										1					

Fig.5

What is asymmetric?

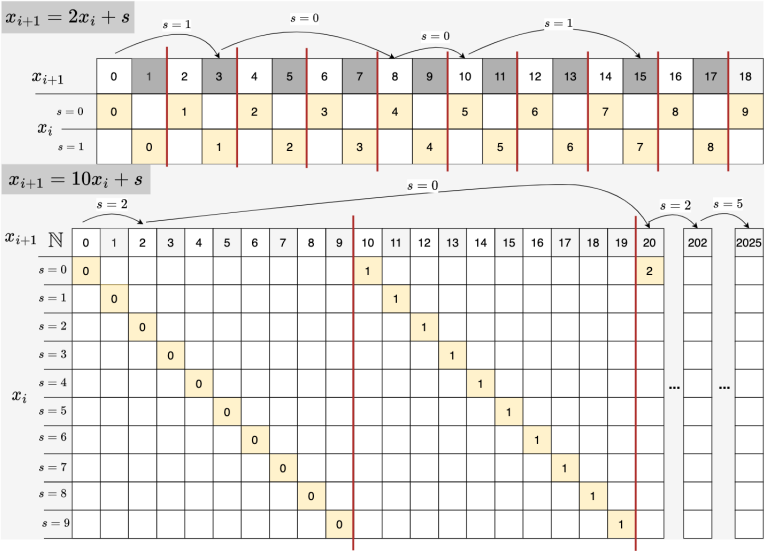

$x_{i+1} = 2x_i + s$
$s = 1$
$s = 0$
$s = 0$
$s = 1$
x_{i+1}
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
x_i
$s = 0$: 0 1 2 3 4 5 6 7 8 9
$s = 1$: 0 1 2 3 4 5 6 7 8
$x_{i+1} = 10x_i + s$
$s = 0$
$s = 2$
$s = 2$
$s = 5$
x_{i+1} $\mathbb{N}$
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 202 2025
x_i
$s = 0$: 0 1 2
$s = 1$: 0 1
$s = 2$: 0 1
$s = 3$: 0 1
$s = 4$: 0 1 ... ...
$s = 5$: 0 1
$s = 6$: 0 1
$s = 7$: 0 1
$s = 8$: 0 1
$s = 9$: 0 1

Fig.6

Exa. 3: Asymmetric Binary System [3]

- Let $\mathcal{A} = \{0, 1\}$
- $Pr(0) = 1/4, Pr(1) = 3/4$

Exa. 3: Asymmetric Binary System [3]

- Let $\mathcal{A} = \{0, 1\}$
- $Pr(0) = 1/4, Pr(1) = 3/4$
- a cycle of length 4, which contains 1 zero and 3 ones

Exa. 3: Asymmetric Binary System [3]

- Let $\mathcal{A} = \{0, 1\}$
- $Pr(0) = 1/4, Pr(1) = 3/4$
- a cycle of length 4, which contains 1 zero and 3 ones
- Assumption:

$x_{i+1} = C(s, x_i) \approx x_i / p_s$, where p_s is the probability of symbol s

Exa. 3: Asymmetric Binary System [3]

$x_{i+1} \approx x_i / Pr(s)$ $\quad Pr(0) = 1/4, Pr(1) = 3/4$

x_{i+1}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	...	52
x_i $s=0$																				...	
x_i $s=1$																					

Fig.7_0

Exa. 3: Asymmetric Binary System [3]

$x_{i+1} \approx x_i / Pr(s)$ $\quad$ $Pr(0) = 1/4, Pr(1) = 3/4$

x_{i+1}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	...	52
x_i $s=0$																				...	
x_i $s=1$																					

Fig.7_1

Exa. 3: Asymmetric Binary System [3]

$x_{i+1} \approx x_i / Pr(s)$ $\quad Pr(0) = 1/4, Pr(1) = 3/4$

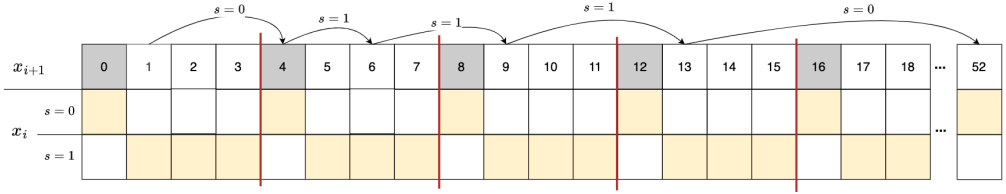


Fig.7_2

Exa. 3: Asymmetric Binary System [3]

$x_{i+1} \approx x_i / Pr(s)$ $\quad Pr(0) = 1/4, Pr(1) = 3/4$

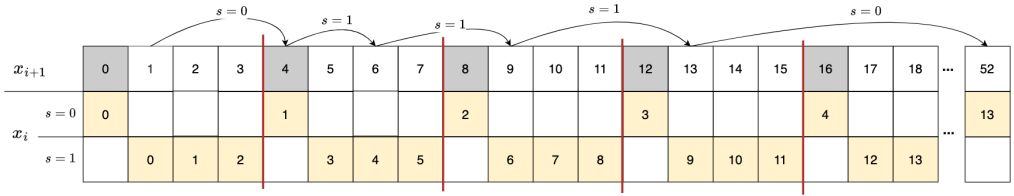


Fig.7

We refine even/odd number by changing their densities

Symmetric or Asymmetric

- Symmetric: uniform distribution
- Asymmetric: non-uniform distribution

Symmetric or Asymmetric

Assumption: $x_{i+1} = C(s, x_i) \approx x_i / p_s$, where p_s is the probability of symbol s

- To change the symmetric behavior so that the information added to x_i by coding a new symbol depends on the probability distribution of the symbols

Symmetric or Asymmetric

Assumption: $x_{i+1} = C(s, x_i) \approx x_i / p_s$, where p_s is the probability of symbol s

- To change the symmetric behavior so that the information added to x_i by coding a new symbol depends on the probability distribution of the symbols
- $log_2(x_i)$ bits

Symmetric or Asymmetric

Assumption: $x_{i+1} = C(s, x_i) \approx x_i / p_s$, where p_s is the probability of symbol s

- To change the symmetric behavior so that the information added to x_i by coding a new symbol depends on the probability distribution of the symbols
- $log_2(x_i)$ bits
- A symbol of probability pcarries $log_2(1/p)$ bits of information (Shannon)

Symmetric or Asymmetric

Assumption: $x_{i+1} = C(s, x_i) \approx x_i/p_s$, where p_s is the probability of symbol s

- To change the symmetric behavior so that the information added to x_i by coding a new symbol depends on the probability distribution of the symbols
- $log_2(x_i)$ bits
- A symbol of probability pcarries $log_2(1/p)$ bits of information (Shannon)
- How many bits of new information? $log_2 x_i + log_2(1/p) \rightarrow log_2 x_{i+1}$

Symmetric or Asymmetric

Assumption: $x_{i+1} = C(s, x_i) \approx x_i/p_s$, where p_s is the probability of symbol s

- To change the symmetric behavior so that the information added to x_i by coding a new symbol depends on the probability distribution of the symbols
- $log_2(x_i)$ bits
- A symbol of probability pcarries $log_2(1/p)$ bits of information (Shannon)
- How many bits of new information? $log_2 x_i + log_2(1/p) \rightarrow log_2 x_{i+1}$
- $log_2 x_{i+1} = log_2 x_i + log_2(1/p) = log_2(x_i/p) \Rightarrow x_{i+1} \approx x_i/p$

rANS

Outline

3. rANS
   3.1 Key Concepts
   3.2 Example: 101110
   3.3 Notations
   3.4 Encoding Function
   3.5 Implementation of Encoder
   3.6 Decoding Function
   3.7 Implemenation of Decoder

rANS

- ”r” = ”range”: similar to range coding (RC)
- symbol spread function
- representation of information
- asymmetric

Exa.1: Standard Binary System

– How to encode 101110? What is the encoding function $C(s, x_i)$?

$x_{i+1} = 2x_i + s$

$s = 0$ $s = 1$ $s = 1$ $s = 1$ $s = 0$

x_{i+1}		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
x_i	$s = 0$	0		1		2		3		4		5		6		7		8		9	
	$s = 1$		0		1		2		3		4		5		6		7		8		9
x_{i+1}		20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
x_i	$s = 0$	10		11		12		13		14		15		16		17		18		19	
	$s = 1$		10		11		12		13		14		15		16		17		18		19
x_{i+1}		40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
x_i	$s = 0$	20		21		22		23		24		25		26		27		28		29	
	$s = 1$		20		21		22		23		24		25		26		27		28		29

Fig.4: 101110

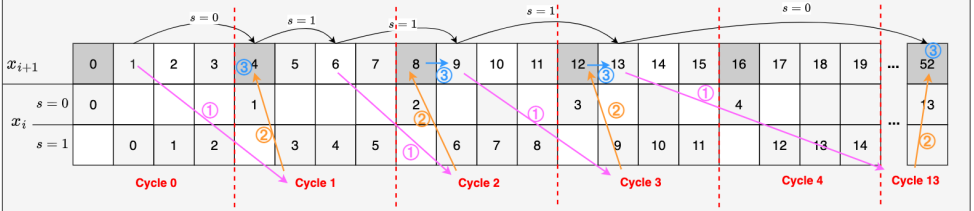
Exa.3: $Pr(0) = 1/4, Pr(1) = 3/4$

- How to encode 101110? What is the encoding function $C(s, x_i)$?
- How can we find the x -th appearance of symbol s?
- $x_{i+1} = C(s, x_i) \approx x_i / p_s$

Some Notations

- $\mathcal{A} = \{s_1, s_2, \ldots, s_k\}$ is the alphabet set
- s is the symbol, if it is used in subscript: s is the current symbol, $s-1$ is the previous symbol, $s+1$ is the next symbol
- F_s is the absolute frequency of symbol s
- $m = \sum_{s \in \mathcal{A}} F_s$ is the total frequency
- $c_s = \sum_{s \in \mathcal{A}} F_{s-1}$ is the cumulative frequency (c_{s+1} is the cumulative frequency of the next symbol)
- $p_s = F_s / m$ is the probability of symbol s
- x_i is the current state (next state: x_{i+1}, previous state: x_{i-1})

$x_{i+1} \approx x_i / Pr(s)$ | $Pr(0) = 1/4, Pr(1) = 3/4$



Given current state x_i and next symbol s. How to find next state x_{i+1}, which is the x_i-th appearance of symbol s ?

STEPS / EXAMPLE	$Pr(0) = 1/4, Pr(1) = 3/4$		General
	$s = 0$	$s = 1$	
Step ①: Find cycle number k	$k = \lfloor x_i/1 \rfloor$	$k = \lfloor x_i/3 \rfloor$	$k = \lfloor x_i/F_s \rfloor$
Step ②: Navigate to the beginning (first number) of that cycle	$4 \cdot k$	$4 \cdot k$	$m \cdot k$
Step ③: Find the position of the symbol s in that cycle	part 1: first of one + 0	part 1: + mod$(x_i, 3)$	part 1: + mod(x_i, F_s)
- part 1: position of the symbol s in the sequence of the same symbol - part 2: how many other symbols before the symbol s	part 2: no other symbols + 0	part 2: only one other symbol (0) and with one of that symbol in one cycle + 1	part 2: only one other symbol (0) and with one of that symbol in one cycle $+c_s$
	$x_{i+1} = m \cdot k$	$x_{i+1} = 4 \cdot \lfloor x_i/3 \rfloor + \text{mod}(x_i, 3) + 1$	$x_{i+1} = m \cdot \lfloor x/F_s \rfloor + \text{mod}(x, F_s) + c_s$

$$x_{i+1} = C(s, x_i) = m \cdot \lfloor x_i/F_s \rfloor + \text{mod}(x_i, F_s) + c_s$$

Fig.8

Encoding Function [3] [5]:

$$x_{i+1} = m \cdot \lfloor x_i / F_s \rfloor + \mathrm{mod}(x_i, F_s) + c_s$$

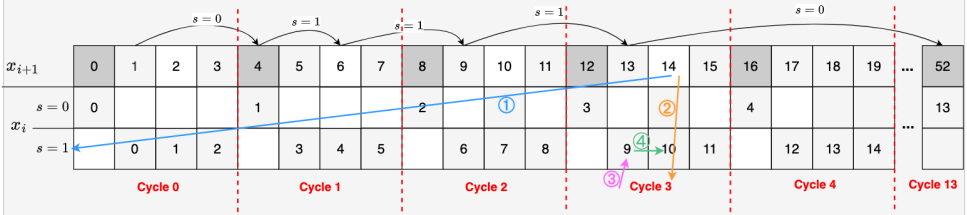
implementation with python (dash app)

```python
def rANS_encoder(symbol_counts, s_input):
    cumul_freq = [] # c_s
    sum_freq = 0 # m
    x = 0 # initial state
    output = [] # list to store encoded (symbols, state)

    for count in symbol_counts:
        cumul_freq.append(sum_freq)
        sum_freq += count
    for s in s_input:
        F_s = symbol_counts[s]
        c_s = cumul_freq[s]
        x = sum_freq * (x // F_s) + (x % F_s) + c_s
        output.append((s, x))

    L_avg = math.ceil(math.log(x) / math.log(2.0)) / len(s_input)
    return output, x, L_avg, entropy(symbol_counts)
```

$x_{i+1} \approx x_i / Pr(s)$ $\boxed{Pr(0) = 1/4, Pr(1) = 3/4}$



Given current state x_i. How to find previous state x_{i-1}, where x_i is the x_{i-1}-th appearance of symbol s ?

STEPS / EXAMPLE	$Pr(0) = 1/4, Pr(1) = 3/4$		General
	$s = 0$	$s = 1$	
Step ① : Find the symbol s for the state x_i	if $x_i \bmod 4 = 0$	otherwise or if $x_i \bmod 4 = 1, 2, 3$	s such that $c_s \leq \mathrm{mod}(x_i, m) < c_{s+1}$
Step ②: Find the cycle k of the current state x_i	$k = x_i/4$	$k = \lfloor x_I/4 \rfloor$	$k = \lfloor x_i/m \rfloor$
Step ③: Find the first number of that symbol s in that cycle k	$x_i/4 \cdot 1$	$\lfloor x_i/4 \rfloor \cdot 3$	$\lfloor x_i/m \rfloor \cdot F_s$
Step ④: Allocate to the exact position	$x_{i-1} = x_i/4 \cdot 1 + 0 - 0$	$x_{i-1} = \lfloor x_i/4 \rfloor \cdot 3 + 2 - 1$	$x_{i+1} = \lfloor x_i/m \rfloor \cdot F_s + \mathrm{mod}(x_i, F_s) - c_s$

1. s such that $c_s \leq \mathrm{mod}(x_1, m) < c_{s+1}$
2. $x_{i-1} = D(x_i) = \lfloor x_i/m \rfloor \cdot F_s + \mathrm{mod}(x_i, F_s) - c_s$

Fig.9

Decoding Function [3] [5]:

- s

$$s \text{ such that } c_s \leq \text{mod}(x_i, m) < c_{s+1}$$

- x_{i-1}

$$x_{i-1} = D(x_i) = \lfloor x_i / m \rfloor \cdot F_s + \text{mod}(x_i, F_s) - c_s$$

implementation with python (dash app)

```python
def rANS_decoder(symbol_counts, x):
    cumul_freq = []  # c_s
    sum_freq = 0 # m
    output = []  # List to store decoded symbols and states
    # compute m and c_s
    for count in symbol_counts:
        cumul_freq.append(sum_freq)
        sum_freq += count
    while x > 0:
        slot = x % sum_freq  # Find the slot in the cumulative frequency range
        s = bisect_right(cumul_freq, slot) - 1 # Binary search to find the symbol s (slot).
        F_s = symbol_counts[s]          # Frequency of the symbol
        c_s = cumul_freq[s]             # Cumulative count of the symbol
        output.append((s,x))            # Append decoded symbol to the output
        x = (x // sum_freq) * F_s + x % sum_freq - c_s  # Update the state
    return output, state
```

Streaming-rANS

Outline

4. Streaming-rANS
- 4.1 Problem
- 4.2 Idea
- 4.3 Solution
  - 4.3.1 Renormalization
- 4.4 Implementation
  - 4.4.1 Encoding
  - 4.4.2 Dncoding

Streaming-rANS

To avoid using large number arithmetic, in practice stream variants are used: which enforce by renormalization: sending the least significant bits of x to or from the bitstream (usually L and b are powers of 2). [4]

Example of the problem //TODO: add example

- ANS (Asymmetric Numeral Systems) uses a single integer x, known as the "state," to represent the encoded data.
- Over time, as more symbols are encoded, the state x can grow arbitrarily large. This can lead to:
  - Overflow: If the value exceeds the precision limits of your system (e.g., a 32-bit or 64-bit register).
  - Inefficient Encoding: A larger x means you need more memory or bits to represent it.
- Using the C function from the previous section to encode succeeding symbols, we can encode a sequence into a very large natural number x, which can be stored as a length $\lfloor lg(x) \rfloor + 1$ bit sequence. [2]

Renormalization in Range Coding

- Range Coding
  - 1
  - 2
- we use the similar idea for rANS
  - range $[L, H]$
  - Encoding: shrink the current state x until it is in the range $[L, H]$
  - Decoding: expand the current state x until it is in the range $[L, H]$
- In AC, renormalization is used to allow finite precision, an analogous approach should be used for ANS. Specifically, we will enforce xto remain in some fixed range I by transferring the least significant bits to the stream. For generality, let us transfer a base b digit, for example $b = 2$ means transferring single bits, while $b = 2^8$ means transferring complete bytes.

Methods from Jarek Duda: Method 1 - Decoding [2]

ANS decoding step from state *x*

$(s, x) = D(x)$ *# decode the current state x*

useSymbol(s)

while $x < L$ **do** *# if x is too small that not in the range [L,H]*

 # expand the x by reading the outstreamed digits back from the stream (an array)

expand$(x, b) = b \cdot x +$ readDigit$() \rightarrow x$

end while

 # return the new state

What is L, H ?

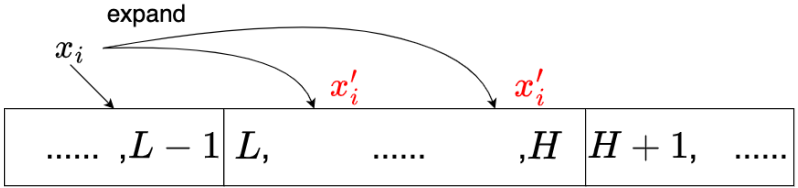


Fig 11

What is L, H ?

- $x_i = b \cdot x_{i-1} +$ bit
- Either x_i in $[L, H]$ or x_{i-1} in $[L, H]$. Assume x_{i-1} in $[L, H]$.
  - $x_{i-1} >= L$
  - $x_i >= b \cdot L + bit >= b \cdot L$
  - x_i can't be in $[L, H]$
  - derive: $H <= b \cdot L - 1$
- $[L, H]$, where $H <= b \cdot L - 1$ (Constraint 1)
- $I = \{L, ..., b \cdot L - 1\}$, which is defined by Jarek Duda b-**unique interval** and $L \in \mathbb{N}$
- This will ensure that the loop in Method 1 will always return to the original value

Methods from Jarek Duda: Method 2 - Encoding [2]

shrink the current state x by stream out the least significant bit (x will be stored in binary and we take out the bit from right side)

ANS encoding step of symbol sfrom state x

 # once the current state x is greater than max value that we define later

while $x > maxX[s]$ **do**

 # write the least significant digit to the stream (stream could be an array in implementaion)

writeDigit(mod(x, b))

shrink$(x, b) = \lfloor x/b \rfloor \rightarrow x$ *# shrink the x*

end while

$x = C(s, x)$ *# encode with the shrunk x*

What is $maxX[s]$?

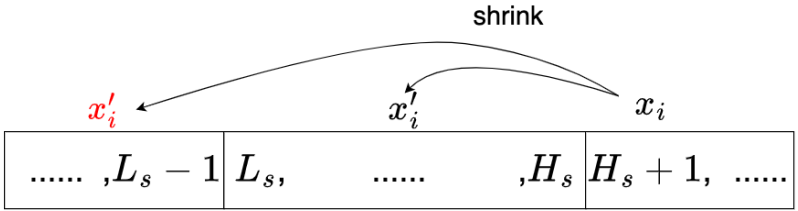


Fig 10

What is $maxX[s]$?

- to make sure not to jump over the range
- Assume that $x_i = H_s + 1$ and it will be shrinked to $x'_i = \lfloor x_i/b \rfloor$
- $x'_i = \lfloor x_i/b \rfloor = \lfloor (H_s + 1)/b \rfloor >= L_s$
- $L_s <= (H_s + 1)/b <= \lfloor (H_s + 1)/b \rfloor < (H_s + 1)/b + 1$
- $H_s >= b \cdot L_s - 1, \forall s \in \mathcal{A}$ (Constraint 2)

Final Form of $[L, H]$ in practice

- Contraints

$$[L, H]: \text{ where } H <= b \cdot L - 1$$
$$[L_s, H_s] : H_s >= b \cdot L_s - 1, \forall s \in \mathcal{A}$$

- In practice ($b = 2^1$): we choose $L = k \cdot m$, $L_s = k \cdot F_s$ ($k \in \mathbb{N}$)

$$\mathcal{I} = [km, 2 \cdot km - 1]$$
$$\mathcal{I}_s = [kF_s,\ 2 \cdot kF_s - 1] \text{ for } \forall s \in \mathcal{A}$$

implementation with python (dash app)

```python
def rANS_stream_encoder(symbol_counts, s_input):
    # Init: c_s, m, x, output, bit_stream (omit the code here)
    for s in s_input:
        F_s = symbol_counts[s]
        c_s = cumul_freq[s]
        L_s = F_s # k = 1
        maxX_s = 2 * L_s - 1
        out_bits = ""

        while x > maxX_s:
            out_bits += str(x % 2)
            x = x // 2
        x = (x // F_s) * sum_freq + (x % F_s) + c_s
        bit_stream += out_bits
        output.append((s, x, out_bits, bit_stream))

    return output, x, bit_stream, L_avg, entropy(symbol_counts)
```

implementation with python (dash app)

```python
def rANS_stream_decoder(symbol_counts, x, bit_stream, num_symbols):
    # Init: c_s, m, x, output, bit_stream (omit the code here)
    bit_stream = list(map(int, bit_stream.split(',')))  # Convert bit_stream to a list of integers
    for _ in range(num_symbols):
        slot = x % sum_freq
        s = bisect_right(cumul_freq, slot) - 1
        F_s = symbol_counts[s]
        c_s = cumul_freq[s]
        output.append((s, x))
        x = (x // sum_freq) * F_s + slot - c_s

        while x < sum_freq and len(bit_stream) > 0:
            x = x * 2 + bit_stream.pop()

    return output, x
```

Summary

Summary

- To summarize, this may help you to make a Leipzig university beamer presentation.

Further Study

Further Study | |

1. Huffman Codes: An Information Theory Perspective - Reducible
2. What is Asymmetric Numeral Systems? - Kedar Tatwawadi
3. Asymmetric Numeral Systems - EE274: Data Compression, course notes
4. Stanford Compression Library
5. Asymmetric Numeral Systems Toolkit - Jarek Duda
6. Presentation and Implementation - Huo Jiang

Reference

1. C. Shannon, “A mathematical theory of communication,” The Bell Sys- tem Technical Journal, vol. 27, pp. 379–423, 623–656, July, October
2. J. Duda, K. Tahboub, N. J. Gadgil and E. J. Delp, ”The use of asymmetric numeral systems as an accurate replacement for Huffman coding,” 2015 Picture Coding Symposium (PCS), Cairns, QLD, Australia, 2015, pp. 65-69, doi: 10.1109/PCS.2015.7170048.
3. J. Duda, “Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding,” arXiv:1311.2540.
4. Asymmetric numeral systems - Wikipedia
5. What is Asymmetric Numeral Systems? - Kedar Tatwawadi
6. Leipzig Beamer Template

Thank you!