Introduction to Source Coding

Dr. Arup Kumar Pal
Department of Computer Science & Engineering
Indian Institute of Technology (ISM)
Dhanbad, Jharkhand-826004
E-mail: arupkrpal@gmail.com

Outline

- Purpose of Compression
- Fundamentals of Data Compression
- Some Source Coding Techniques
- References

Purpose of Compression

- Gray Scale Image: 720×480 = 337.5KB
- Color Image: 720×480 = 0.99 MB
- 90 minutes Color Video: 720×480×30frames/sec=157 GB
- Internet Speed: 56kbps
- Approx. download time: 49 seconds, 148 seconds and 278 days respectively.
- Outcomes of Compression:
  - Reduce Storage Space
  - Reduce Transmission Time
  - Reduce Data access Time

Data Compression

- Compression consists in reducing the physical size of information blocks.
- Compression is achieved by removing data redundancy while preserving information content.
- To adopt suitable coding and approaches for achieving the more compact representation of the data.

Example-1

- Consider the following sequence
  BDBCACDBCDBDBDACDA

  Total No of Characters: 18

  Representation of Code:

  00 for A, 01 for B, 10 for C and 11 for D

- Here the sequence is generated by a process that follows the following rules: (i) ***A*** is followed by ***B*** or ***C***; (ii) ***B*** is followed by ***C*** or D; (iii) ***C*** is followed by ***A or D***; (iv) ***D*** is followed by ***A or B***

B	**D**	**B**	**C**	**A**	**C**	**D**	**B**	**C**	**D**	**B**	**D**	**B**	**D**	**A**	**C**	**D**	**A**
01	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**0**

Example-2

- Consider the following sequence

9	11	11	11	14	13	15	17	16	17	20	21

$$\hat{x}_n = n + 8 \quad n = 1, 2, \cdots, 12$$

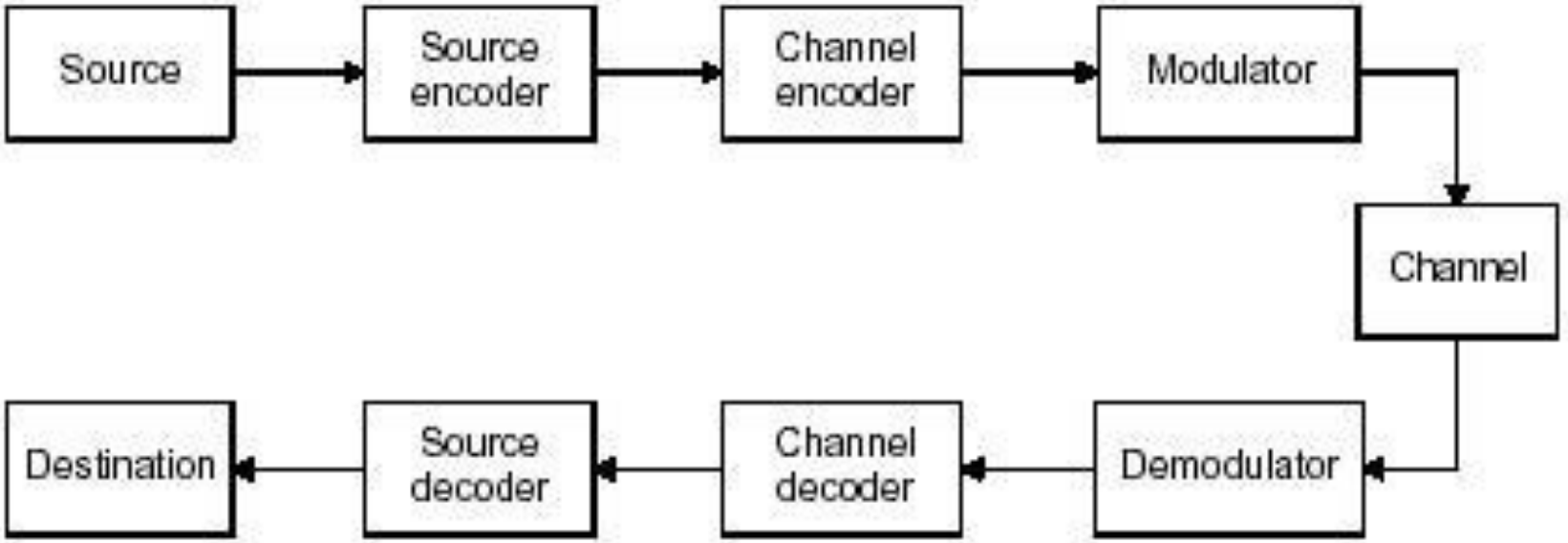
$$e_n = x_n - \hat{x}_n$$

0	1	0	-1	1	-1	0	1	-1	-1	1	1

Representation of Code:

*00 for **-1**, 01 for **0** and 10 for **1***

Communication System



General model of a communication system

Source Coding

- The goal of source coding for digital systems is two fold:
  - Uniquely map any arbitrary source message to a binary code and back again.
  - Efficiently map a source message to the most compact binary code and back again.
- Techniques are
  - Fixed-length code
  - Variable-length code (Goal: Minimizing the average code word length)

Channel Coding

- Channel coding is considered for reliable communication between transmitter and receiver.
- Two policies to deal with channel coding are:
  - Forward Error Correction (FEC)
  - Automatic Repeat Request (ARQ)

Uniquely Decodable Codes

- Variable length code assigns a bits of string (codeword) of variable length to every message value

  e.g. $a = 1, b = 01, c = 101, d = 011$

  What if you get the sequence of bits 1011 ?

  Is it aba, ca, or, ad?

- Uniquely decodable code is a variable length code in which bit strings can always be uniquely decomposed into its codewords.

Prefix Codes

- Prefix code is a variable length code in which no codeword is a prefix of another word
- It is possible to decode each message in an encoded string without reference to succeeding code symbols.

Example

Source	Code 1	Code 2
a_1	0	0
a_2	10	01
a_3	110	011
a_4	1110	0111

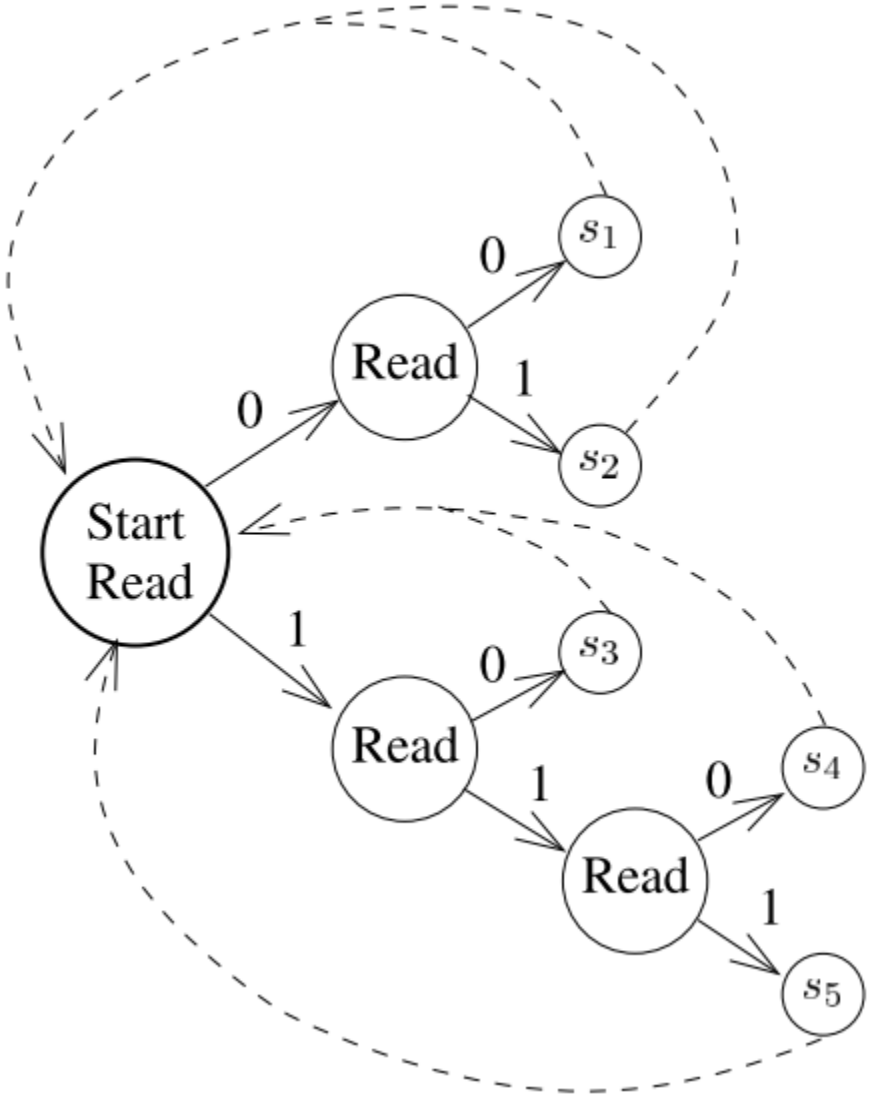
Code1 is Prefix code.

DEFINITION 3.5 Prefix of a Code *Let* $C(s_i) = x_{j1}x_{j2}\ldots x_{jn}$ *be a code word of length* n. *A sub-string of code characters from* $C(s_i)$, $x_{j1}x_{j2}\ldots x_{jm}$, *where* $m < n$, *is called a* prefix *of the code word* $C(s_i)$.

DEFINITION 3.6 Prefix Condition *A necessary and sufficient condition for a code to be instantaneous is that no complete code word of the code be a prefix of some other code word.*

Decoding Prefix Codes

Source	Code
s_1	00
s_2	01
s_3	10
s_4	110
s_5	111



The Kraft Inequality

- If the individual code word lengths are specified there is no guarantee that an instantaneous code can be designed with those lengths.
- The Kraft Inequality theorem provides a limitation on the code word lengths for the design of instantaneous codes.
- It only indicates whether an instantaneous code can be designed from the given code word lengths.

THEOREM ***Kraft Inequality***

A necessary and sufficient condition for the existence of an instantaneous code with alphabet size r *and* q *code words with individual code word lengths of* $l_1, l_2, \ldots, l_q$ *is that the following inequality be satisfied:*

$$K = \sum_{i=1}^{q} r^{-l_i} \leq 1$$

Conversely, given a set of code word lengths that satisfy this inequality, then there exists an instantaneous code with these word lengths.

Example

Source	Code A	Code B	Code C
s_1	0	0	0
s_2	100	100	10
s_3	110	110	110
s_4	111	11	11

Compact Code

Definition: The code will be compact code if its average length is less than or equal to the average length of all other uniquely decodable codes for the same source and code alphabet.

Source	P_i	Code A	Code B
s_1	0.5	00	1
s_2	0.1	01	000
s_3	0.2	10	001
s_4	0.2	11	01

$$L_A = (0.5\times 2)+(0.1\times 2)+(0.2\times 2)+(0.2\times 2)$$
$$= 2 \text{ bits per symbol}$$
$$L_B = (0.5\times 1)+(0.1\times 3)+(0.2\times 3)+(0.2\times 2)$$
$$= 1.8 \text{ bits per symbol}$$

Entropy

Definition: Shannon's measure of information is the number of bits to represent the amount of uncertainty (randomness) in a data source, and is defined as entropy.

$$H(S) = -\sum_{i=1}^{n} p(S_i) \log(p(S_i))$$

Theorem: Shannon's Noiseless Coding Theorem Let L be the average length of a compact code for the source S, then $H(S) \leq L < H(S)+1$

Efficiency of Code

- The efficiency of the code is given by

$$\eta = \frac{H(S)}{\overline{R}} \times 100\%$$

Source	P_i	Code A	Code B
s_1	0.5	00	1
s_2	0.1	01	000
s_3	0.2	10	001
s_4	0.2	11	01

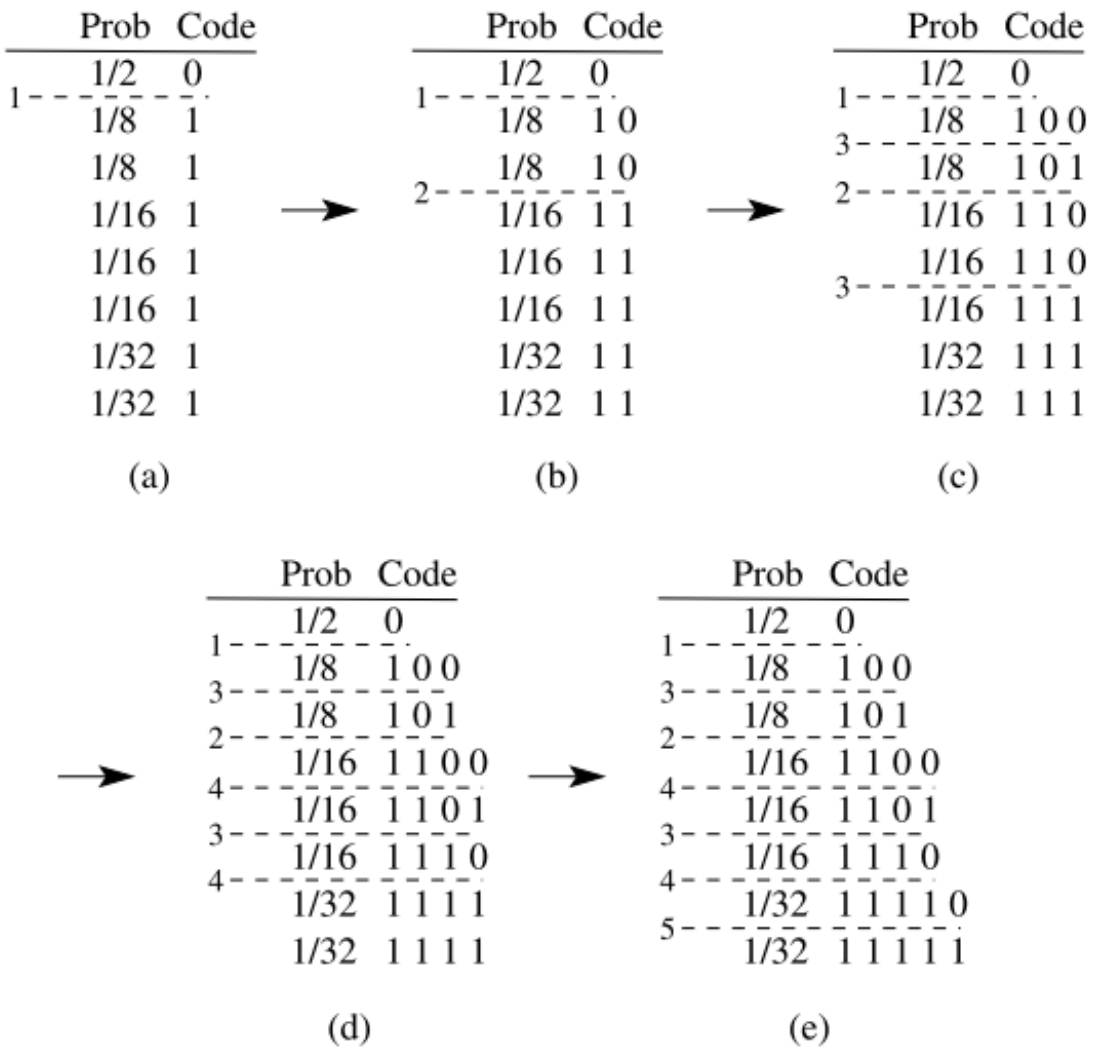
$$H(S) = 1.76, R_A = 2, R_B = 1.8$$

$$\eta_A = 88\%$$

$$\eta_B = 98\%$$

Conclusion: Code B can not be improved more than 2%

Shannon Fano Codes



It does not give optimal variable length code

Huffman Codes

- Key idea: Assign fewer bits to symbols that occur more frequently and more bits to symbols appear less often.
- Huffman codes are compact codes.

Frequency of characters

Character	A	B	C	D	E
Frequency	17	12	12	27	32

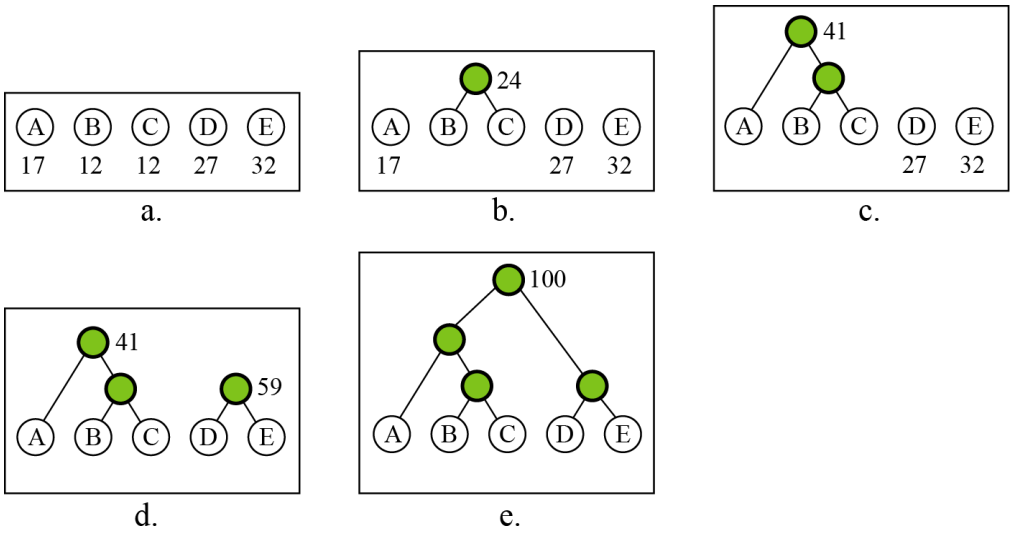
Huffman Code Design

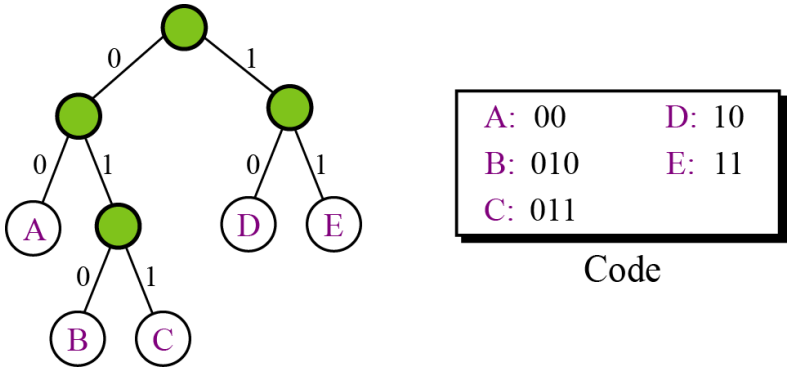
- Requirement: The source probability distribution.

  (Not Available in many cases)

- Procedure:
  - Step 1: Sort the probability of all source symbols in descending order.
  - Step 2: Merge the last two into new symbol, add their probabilties
  - Step 3: Repeat Step 1 and Step 2 until one symbol is left
  - Step4: Code assignment, traverse the tree from the root to each leaf node

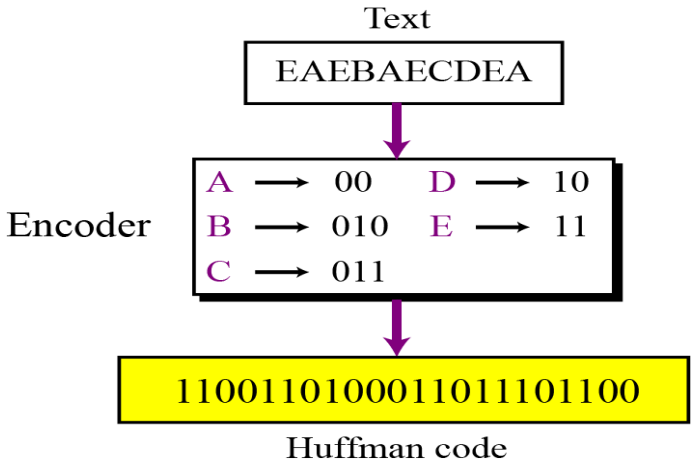
Binary Huffman Coding





Binary Huffman Coding (Cont'd)

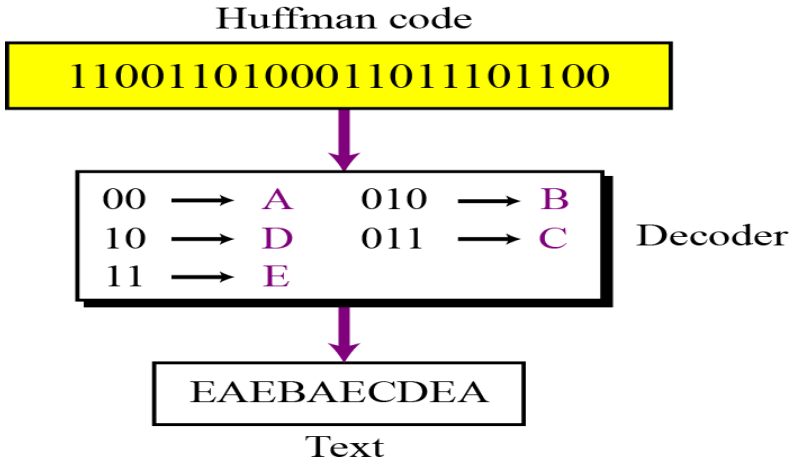
- Encoding



Total No of bits:50

Total No of bits:22

- Decoding

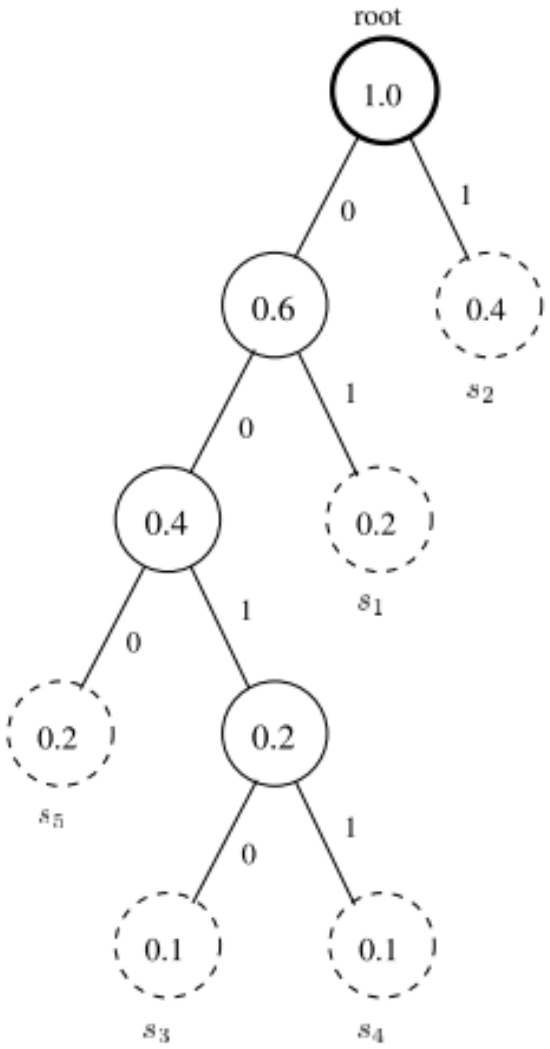


Minimum Variance Huffman Codes

- Same probabilities of source may give a number of Huffman trees.
- Approach: Put the combined symbol as high as possible in the sorted list
- Average length variance, $\sigma^2 = \sum P_i (l_i - L)^2$

Minimum Variance Huffman Codes

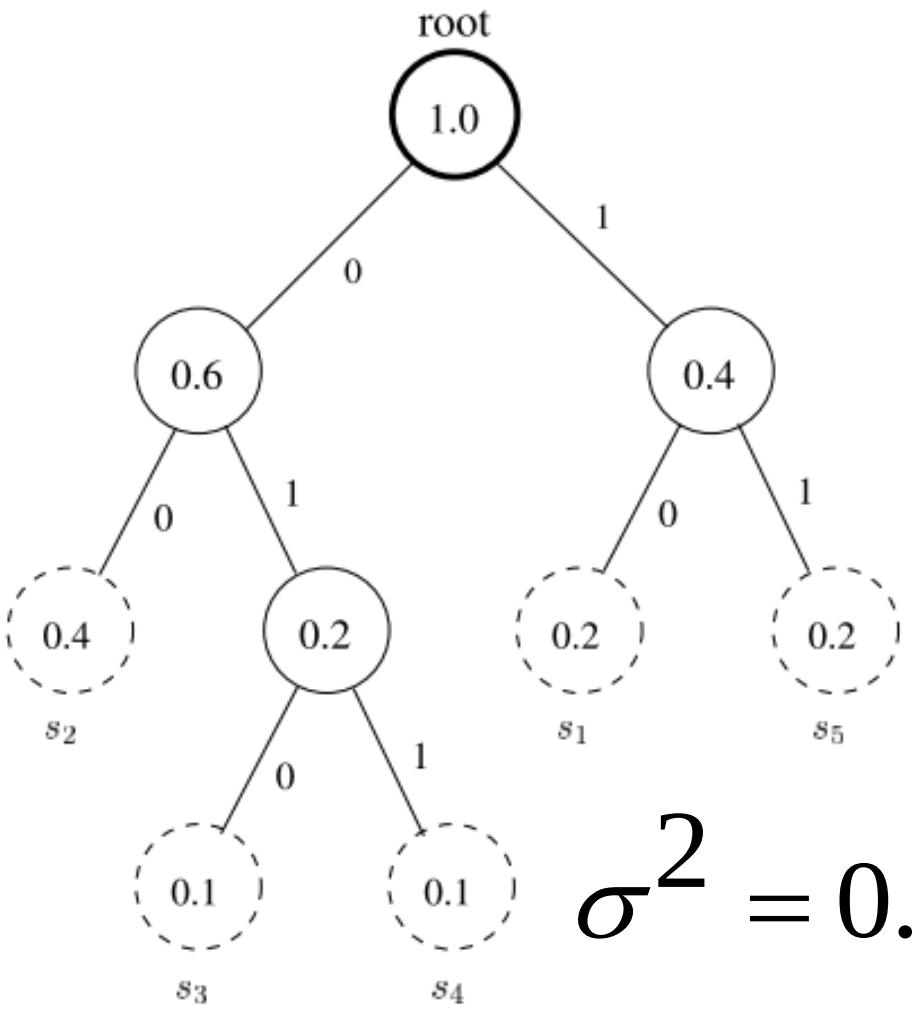
Symbol	$P(s_i)$	Huffman Code
s_1	0.2	01
s_2	0.4	1
s_3	0.1	0010
s_4	0.1	0011
s_5	0.2	000



$$\sigma^2 = 1.88$$

Minimum Variance Huffman Codes

Symbol	$P(s_i)$	Huffman Code
s_1	0.2	10
s_2	0.4	00
s_3	0.1	010
s_4	0.1	011
s_5	0.2	11



$$\sigma^2 = 0.16$$

Canonical Huffman Codes

- Canonical Huffman Code is well structured.
- Rules:
  - Assign 0 to left branch and 1 to right branch
  - Build the tree from left to right in increasing order of depth
  - Each leaf is placed at the first available position

Canonical Huffman Codes

Symbol (i)	Probability	Huffman Code 1	Huffman Code 2	Canonical Code
A (1)	0.1	100	011	001
B (2)	0.15	001	110	010
C (3)	0.05	0000	1111	0000
D (4)	0.09	0001	1110	0001
E (5)	0.14	101	010	011
F (6)	0.27	01	10	10
G (7)	0.2	11	00	11

Performance of Huffman Codes

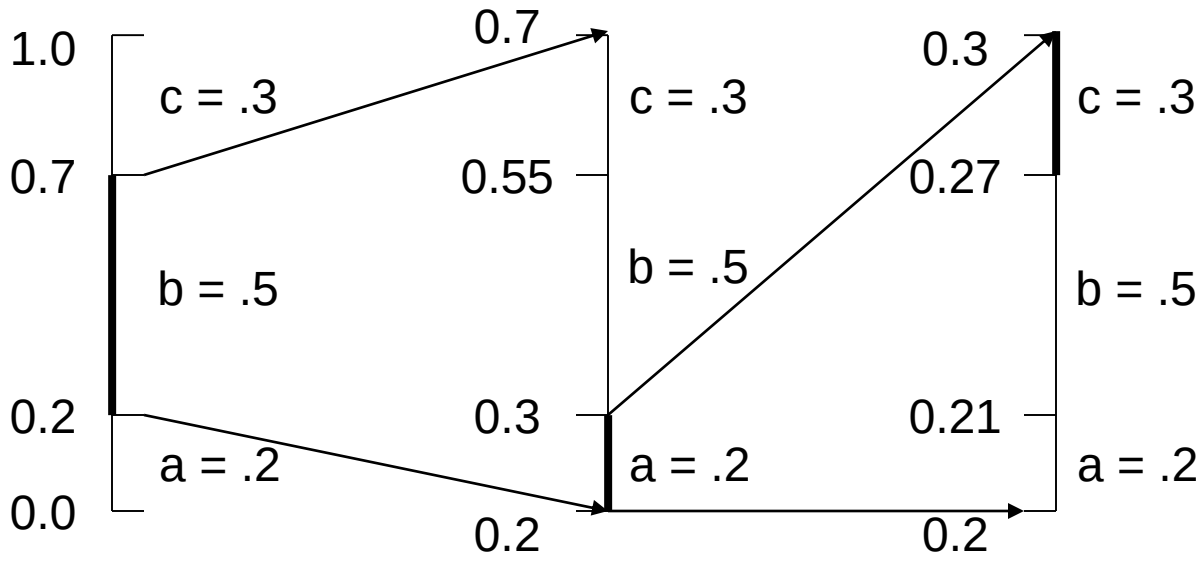
$$H(S) \leq \bar{R} < H(S) + 1$$

Arithmetic Coding

- Huffman – needs to transmit Huffman tables with compressed data
- Arithmetic – needs to transmit length of encoded string with compressed data

Arithmetic Coding: Encoding Example

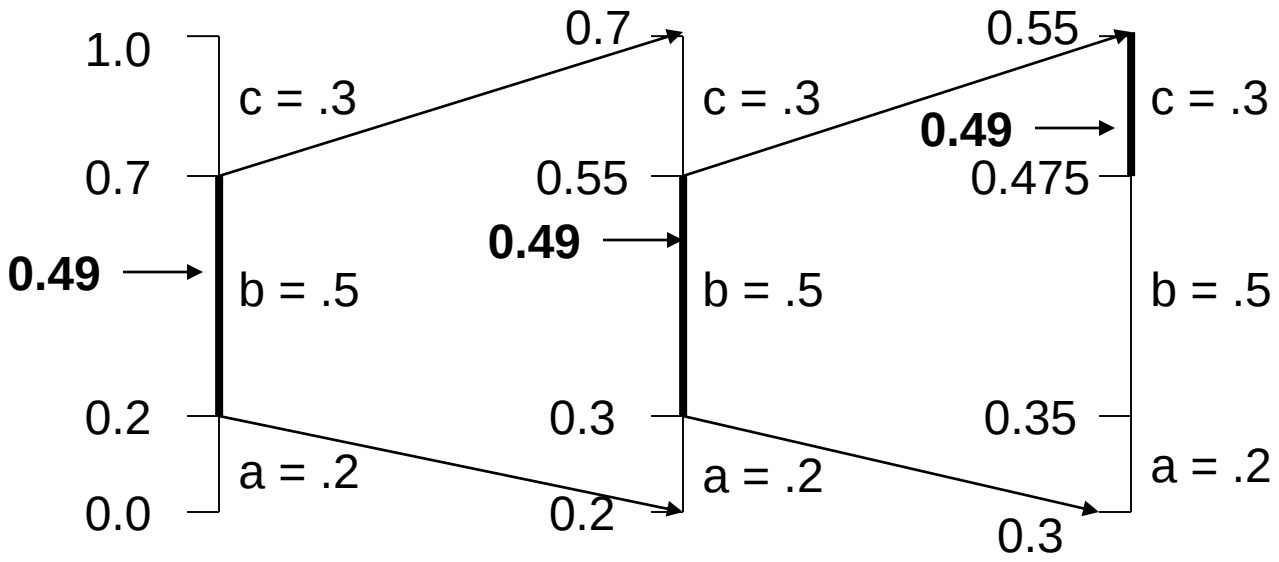
Coding the message sequence: **bac**



The final interval is **[.27,.3)**

Arithmetic Coding: Decoding Example

Decoding the number 0.49, knowing the message is of length 3:



The message is **bbc.**