

CS 414 – Multimedia Systems Design
Lecture 7 – Basics of
Compression (Part 2)

Klara Nahrstedt
Spring 2008



Administrative

- MP1 is posted
- Discussion meeting on Thursday, January 31, at 7pm, 3403 SC.



Huffman Encoding

- Statistical encoding
- To determine Huffman code, it is useful to construct a binary tree
- Leaves are characters to be encoded
- Nodes carry occurrence probabilities of the characters belonging to the sub-tree



Huffman Encoding (Example)

**Step 1 : Sort all Symbols according to their probabilities
(left to right) from Smallest to largest
these are the leaves of the Huffman tree**

$$P(B) = 0.51$$

$$P(C) = 0.09$$

$$P(E) = 0.11$$

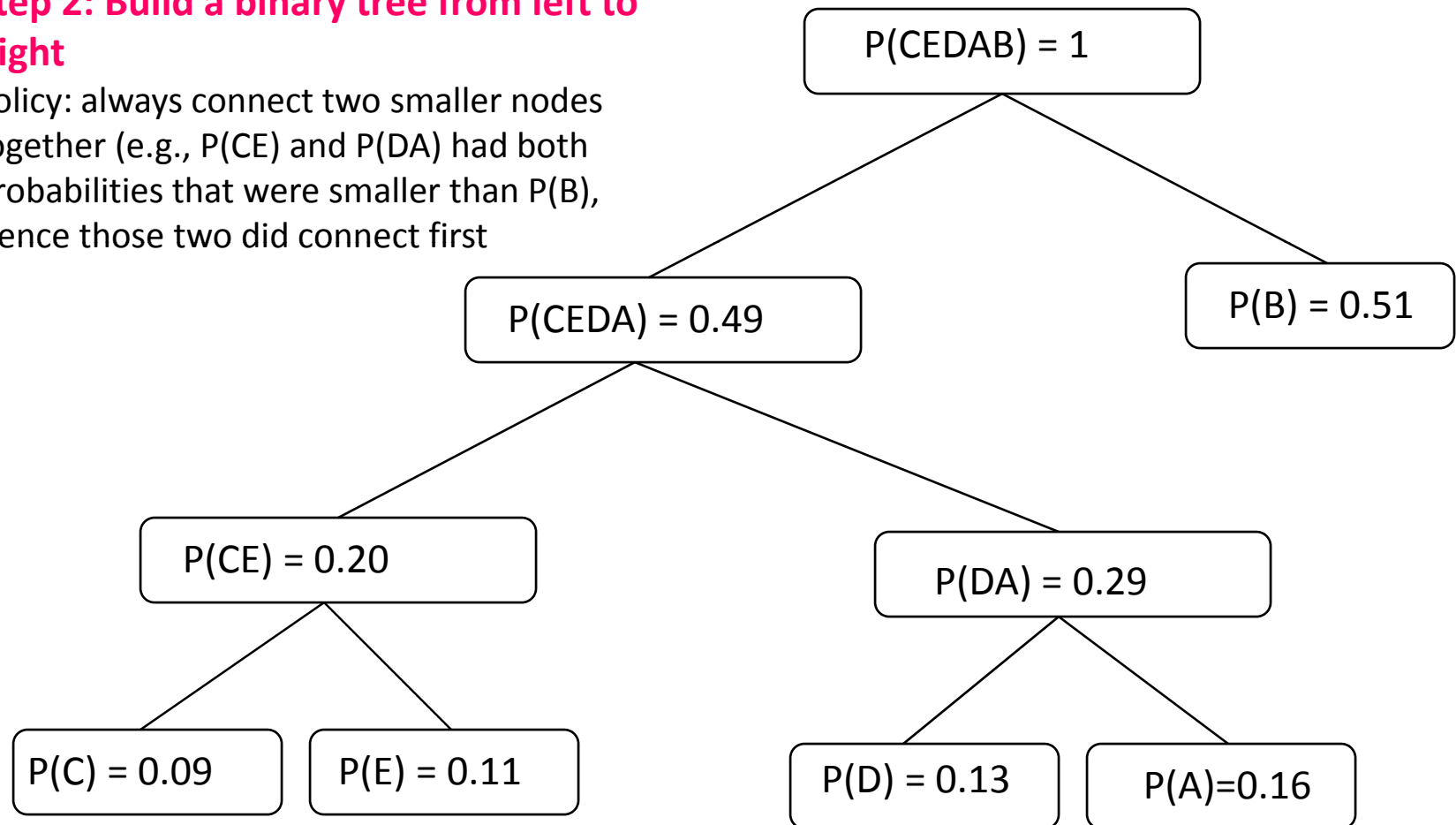
$$P(D) = 0.13$$

$$P(A) = 0.16$$

Huffman Encoding (Example)

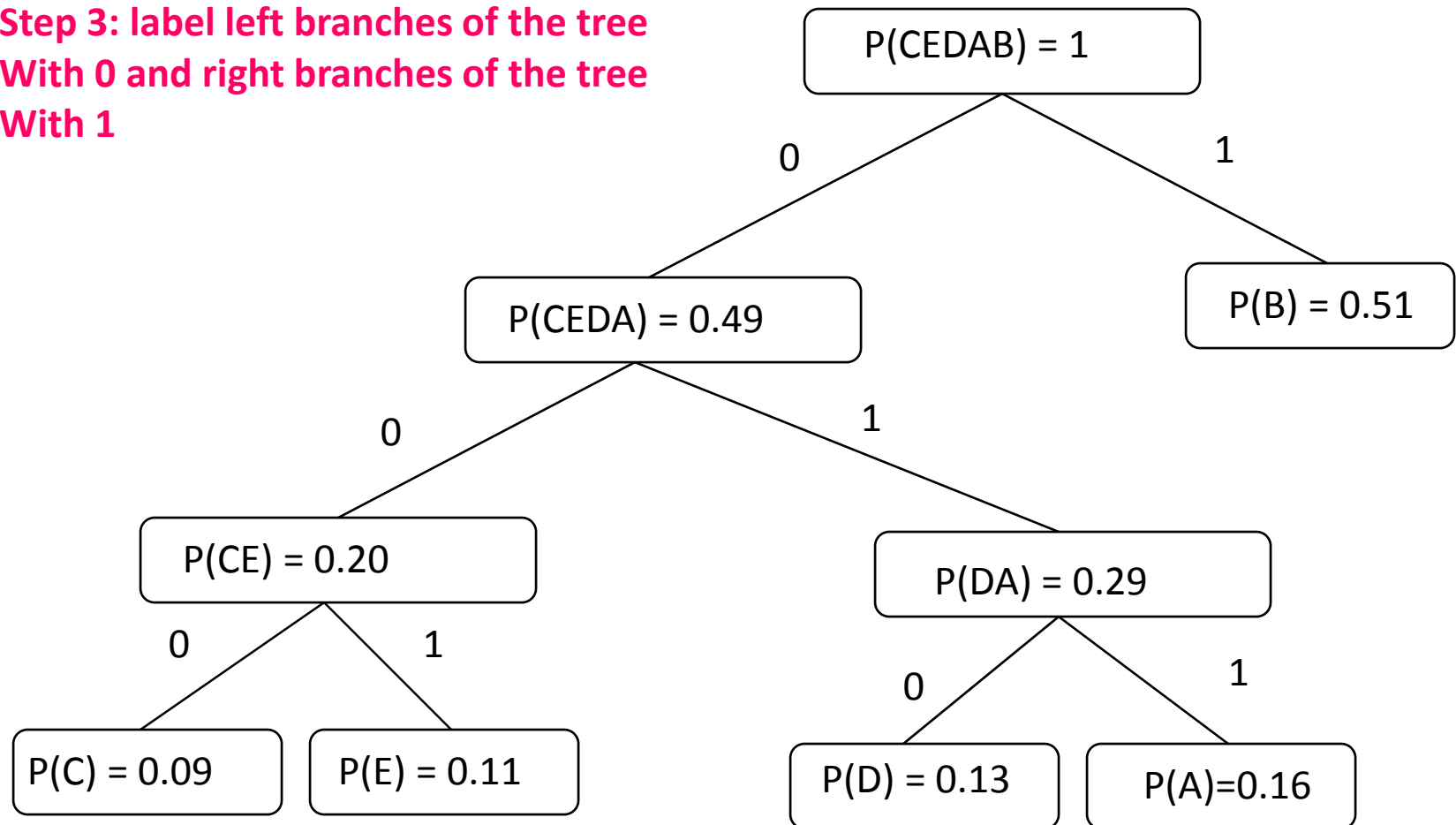
Step 2: Build a binary tree from left to Right

Policy: always connect two smaller nodes together (e.g., $P(CE)$ and $P(DA)$ had both Probabilities that were smaller than $P(B)$, Hence those two did connect first



Huffman Encoding (Example)

Step 3: label left branches of the tree
With 0 and right branches of the tree
With 1



Huffman Encoding (Example)

Step 4: Create Huffman Code

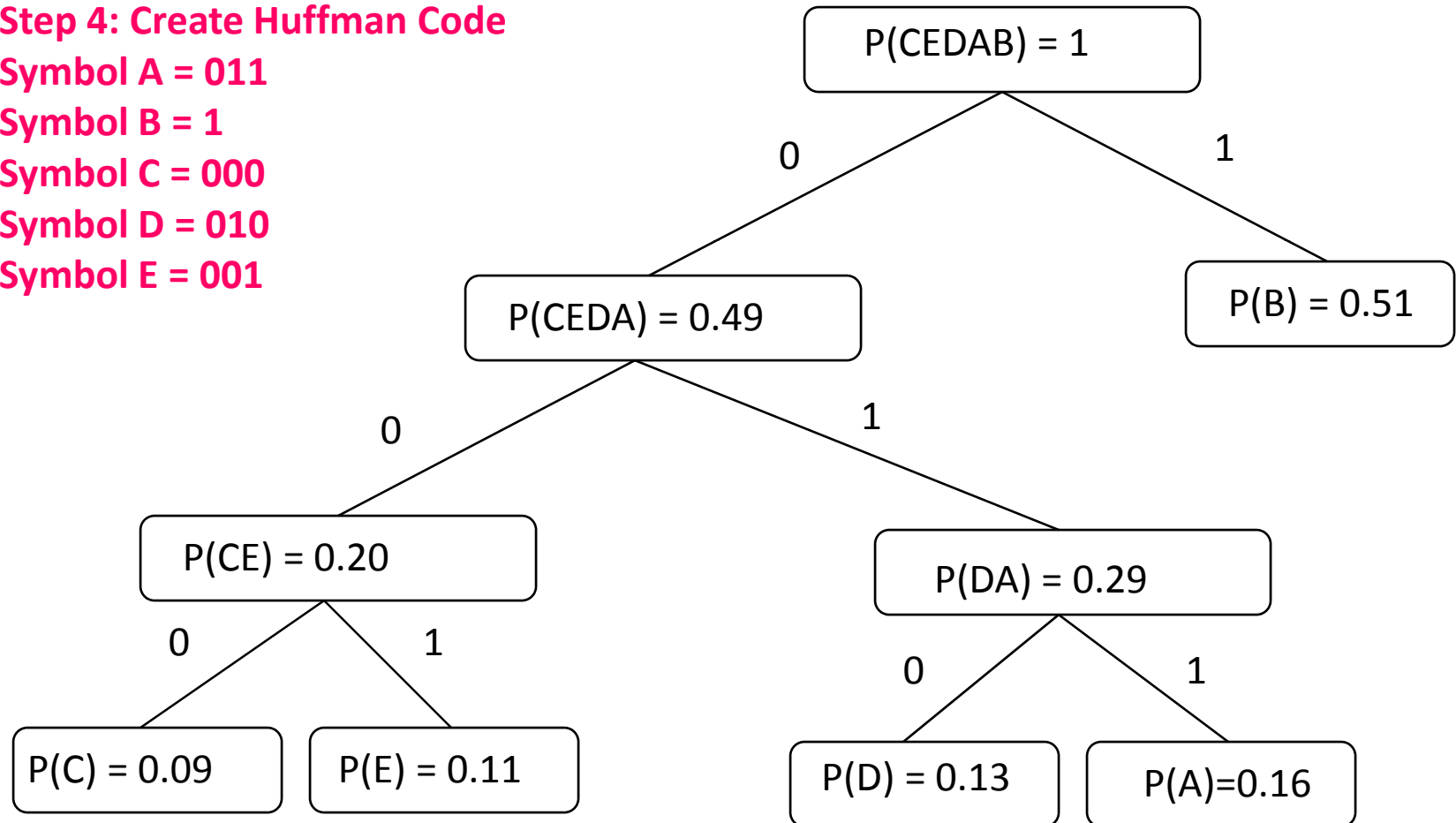
Symbol A = 011

Symbol B = 1

Symbol C = 000

Symbol D = 010

Symbol E = 001

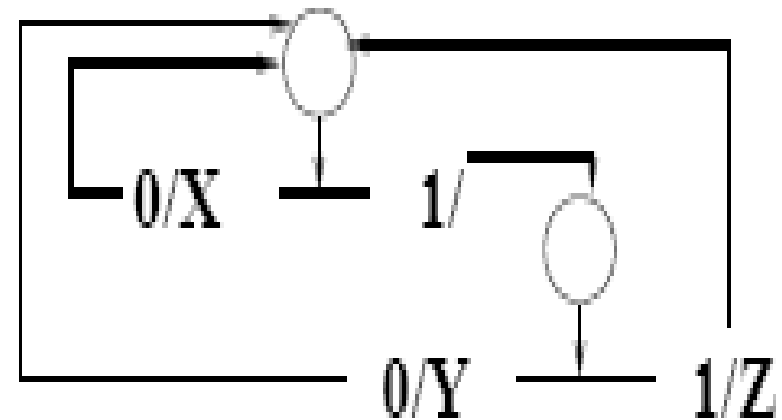


Huffman Decoding

- Assume Huffman Table

Symbol	Code
X	0
Y	10
Z	11

Consider encoded
bitstream:
000101011001110



What is the decoded string?



Huffman Example

- Construct the Huffman coding tree (in class)

Symbol (S)	$P(S)$
A	0.25
B	0.30
C	0.12
D	0.15
E	0.18



Characteristics of Solution

Symbol (S)	Code
A	01
B	11
C	100
D	101
E	00



Example Encoding/Decoding

Encode "BEAD"

⇒ 110001101

⇒ Decode "0101100"

Symbol (S)	Code
A	01
B	11
C	100
D	101
E	00

Entropy (Theoretical Limit)

$$H = \sum_{i=1}^N -p(s_i) \log_2 p(s_i)$$

$$\begin{aligned} &= -.25 * \log_2 .25 + \\ &\quad -.30 * \log_2 .30 + \\ &\quad -.12 * \log_2 .12 + \\ &\quad -.15 * \log_2 .15 + \\ &\quad -.18 * \log_2 .18 \end{aligned}$$

$$H = 2.24 \text{ bits}$$

Symbol	$P(S)$	Code
A	0.25	01
B	0.30	11
C	0.12	100
D	0.15	101
E	0.18	00

Average Codeword Length

$$L = \sum_{i=1}^N p(s_i) \text{codelength}(s_i)$$

$$\begin{aligned} &= .25(2) + \\ &\quad .30(2) + \\ &\quad .12(3) + \\ &\quad .15(3) + \\ &\quad .18(2) \end{aligned}$$

$$L = 2.27 \text{ bits}$$

Symbol	$P(S)$	Code
A	0.25	01
B	0.30	11
C	0.12	100
D	0.15	101
E	0.18	00



Code Length Relative to Entropy

$$L = \sum_{i=1}^N p(s_i) \text{codelength}(s_i)$$

$$H = \sum_{i=1}^N -p(s_i) \log_2 p(s_i)$$

- Huffman reaches entropy limit when all probabilities are negative powers of 2
 - i.e., 1/2; 1/4; 1/8; 1/16; etc.
- $H \leq \text{Code Length} \leq H + 1$



Example

$$\begin{aligned} H &= -.01 \cdot \log_2 .01 + \\ &\quad -.99 \cdot \log_2 .99 \\ &= .08 \end{aligned}$$

$$\begin{aligned} L &= .01(1) + \\ &\quad .99(1) \\ &= 1 \end{aligned}$$

Symbol	$P(S)$	Code
A	0.01	1
B	0.99	0



Group Exercise

- Compute Entropy (H)
- Build Huffman tree
- Compute average code length
- Code “BCCADE”

Symbol (S)	$P(S)$
A	0.1
B	0.2
C	0.4
D	0.2
E	0.1



Limitations

- Diverges from lower limit when probability of a particular symbol becomes high
 - always uses an integral number of bits
- Must send code book with the data
 - lowers overall efficiency
- Must determine frequency distribution
 - must remain stable over the data set



Arithmetic Coding

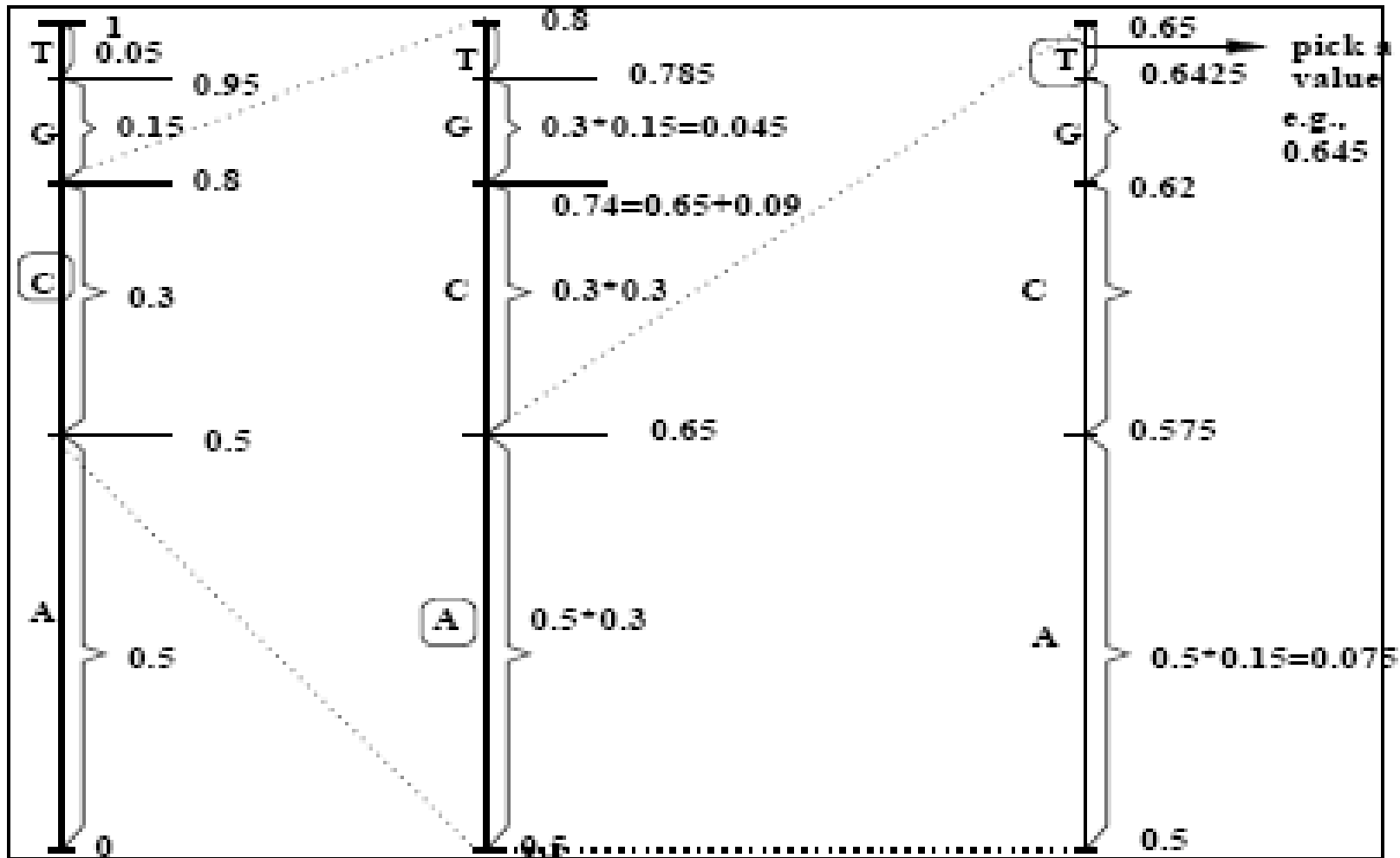
- Optimal algorithm as Huffman coding wrt compression ratio
- Better algorithm than Huffman wrt transmitted amount of information
 - Huffman – needs to transmit Huffman tables with compressed data
 - Arithmetic – needs to transmit length of encoded string with compressed data



Arithmetic Coding

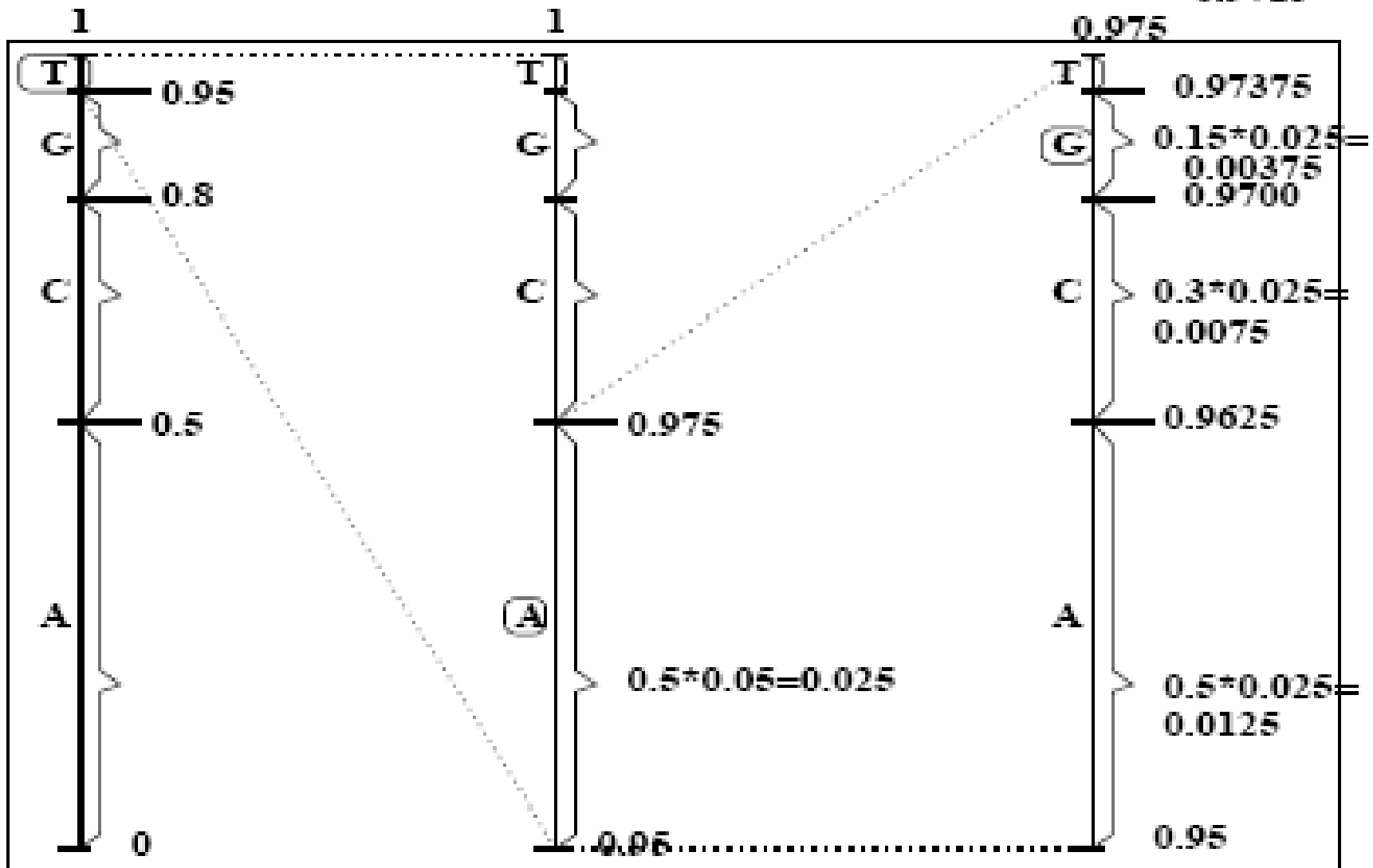
- Each symbol is coded by considering the prior data
- Encoded data must be read from the beginning, there is no random access possible
- Each real number (< 1) is represented as binary fraction
 - $0.5 = 2^{-1}$ (binary fraction = 0.1); $0.25 = 2^{-2}$ (binary fraction = 0.01), $0.625 = 0.5 + 0.125$ (binary fraction = 0.101)

$P(A)=0.5, P(C) = 0.3, P(G) = 0.15, P(T) = 0.05 \Rightarrow$ Encode CAT



Send Value: 0.645

Given: $P(A)=0.5$, $P(C) = 0.3$, $P(G) = 0.15$, $P(T) = 0.05 \Rightarrow$ Decode:
0.9715



Decoded Word: TAG



Adaptive Encoding (Adaptive Huffman)

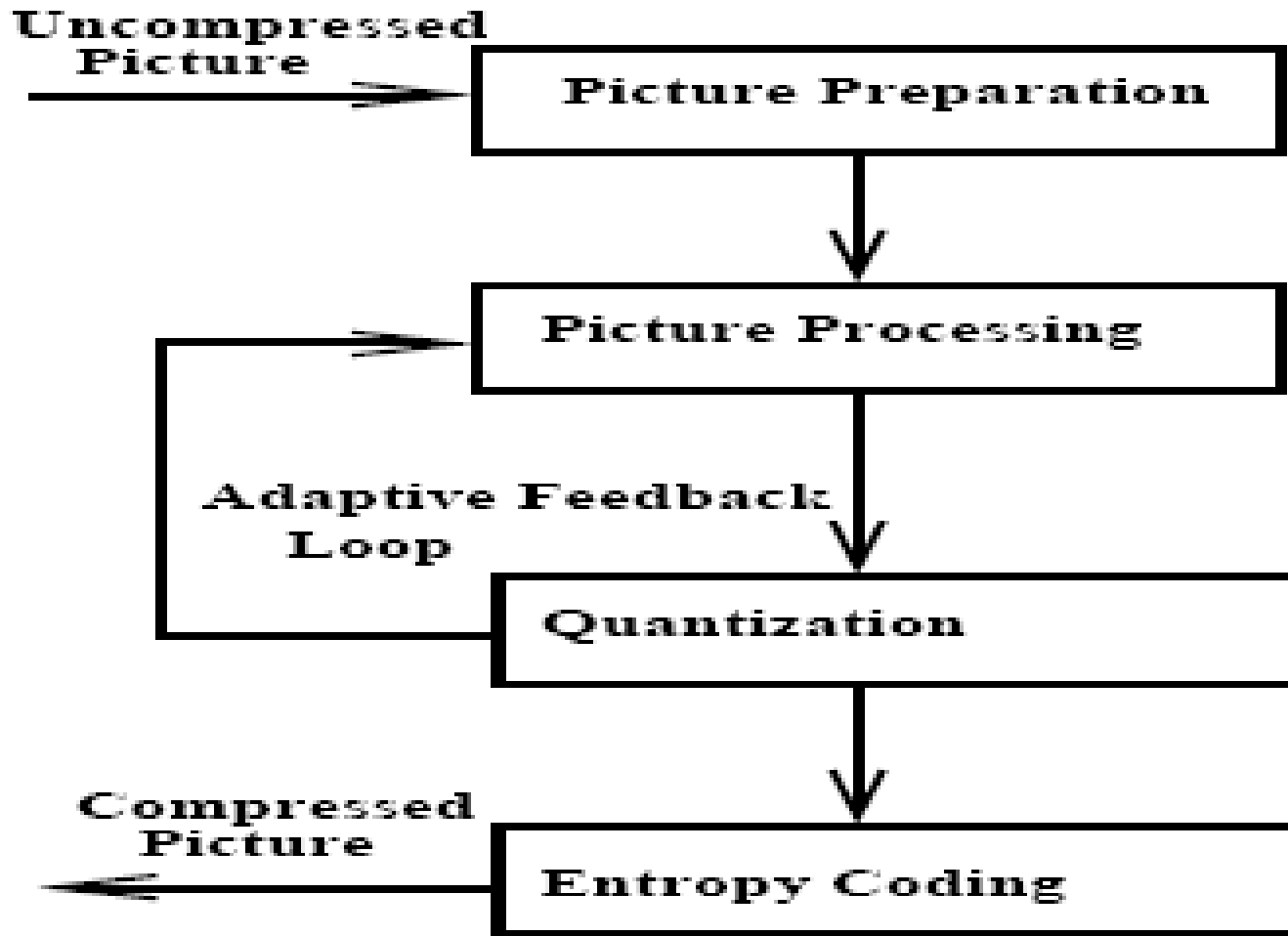
- Huffman code change according to usage of new words and new probabilities can be assigned to individual letters
- If Huffman tables adapt, they must be transmitted to receiver side

Adaptive Huffman Coding Example

Symbol	Code	Original probabilities
A	001	$P(A) = 0.16$
B	1	$P(B) = 0.51$
C	011	$P(C) = 0.09$
D	000	$P(D) = 0.13$
E	010	$P(E) = 0.11$

Symbol	Code	New Probabilities (based on new word BACAAB)
A	1	$P(A) = 0.5$
B	01	$P(B) = 1/3$
C	001	$P(C) = 1/6$
D	0000	$P(D) = 0$
E	0001	$P(E) = 0$

Hybrid Coding





Picture Preparation

- Generation of appropriate digital representation
- Image division into 8x8 blocks
- Fix number of bits per pixel (first level quantization – mapping from real numbers to bit representation)



Other Compression Steps

- Picture processing (Source Coding)
 - Transformation from time to frequency domain (e.g., use Discrete Cosine Transform)
 - Motion vector computation in video
- Quantization
 - Reduction of precision, e.g., cut least significant bits
 - Quantization matrix, quantization values
- Entropy Coding
 - Huffman Coding + RLE



Audio Compression and Formats

- MPEG-3
- ADPCM
- u-Law
- Real Audio
- Windows Media (.wma)

- Sun (.au)
- Apple (.aif)
- Microsoft (.wav)



Image Compression and Formats

- RLE
- Huffman
- LZW
- GIF
- JPEG
- Fractals

- TIFF, PICT, BMP, etc.



Video Compression and Formats

- H.261/H.263
- Cinepak (early 1992 Apple's video codec in Quick-time video suite)
- Sorensen (Sorenson Media, used in Quick-time and Macromedia flash)
- Indeo (early 1992 Intel video codec)
- Real Video (1997 RealNetworks)
- MPEG-1, MPEG-2, MPEG-4, etc.

- QuickTime, AVI, WMV (Windows Media Video)



Summary

- Important Lossless (Entropy) Coders
 - RLE, Huffman Coding and Arithmetic Coding
- Important Lossy (Source) Coders
 - Differential PCM (DPCM) – calculate difference from previous values – one has fewer values to encode
 - Loss occurs during quantization of sample values, hence loss on difference precision occurs as well

Solution

- Compute Entropy (H)

- $H = 2.1$ bits

- Build Huffman tree

- Compute code length

- $L = 2.2$ bits

Symbol	P(S)	Code
A	0.1	100
B	0.2	110
C	0.4	0
D	0.2	111
E	0.1	101

- Code “BCCADE” \Rightarrow 11000100111101