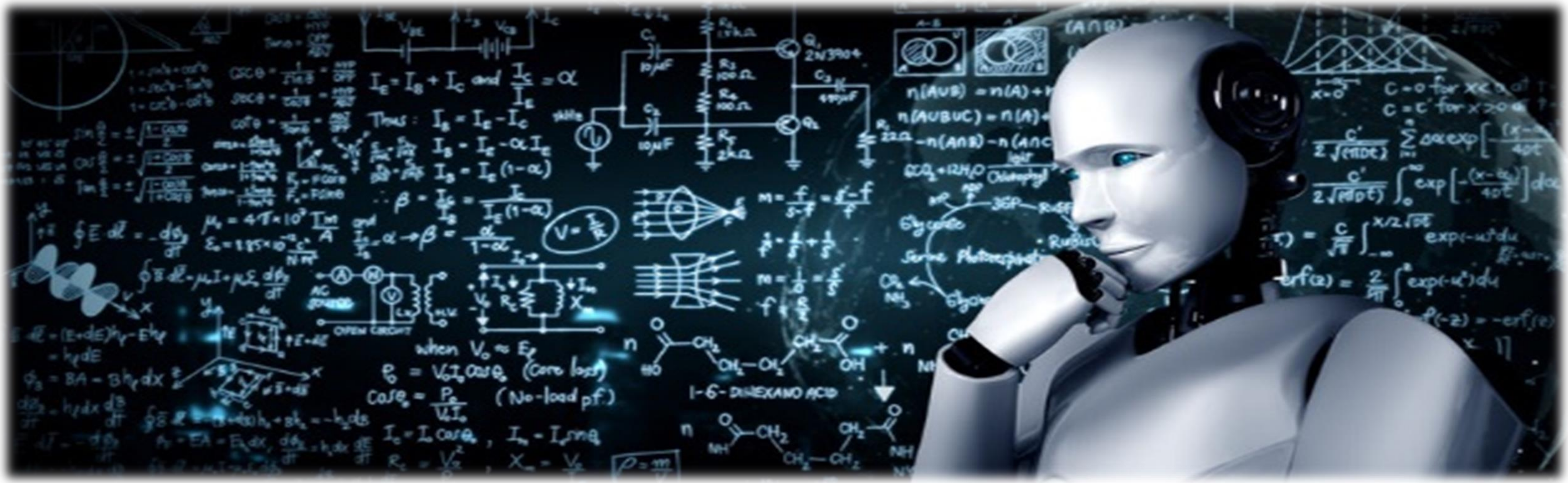


Lecture 07 – Data Compression

Design and Analysis of Algorithms – IT1205

Year 02 Semester 02

SCU
SLIIT CITY UNI

Data Compression

- What is data compression?
  - Data compression is a way of storing data in a format that requires less space than usual.

- Why do we compress data?
  - To save the space.
  - To speed up transferring data.

How to compress?

- For example:-

  00000000000000000000000000000000000000000000000000

  - compress the string by simply saying:
    - Repeat '0' 50 times
    - **X0#50**

Data compression.

- Two kinds of data compression.

  1. *Lossless compression*
  2. *Lossy compression*

Lossless compression

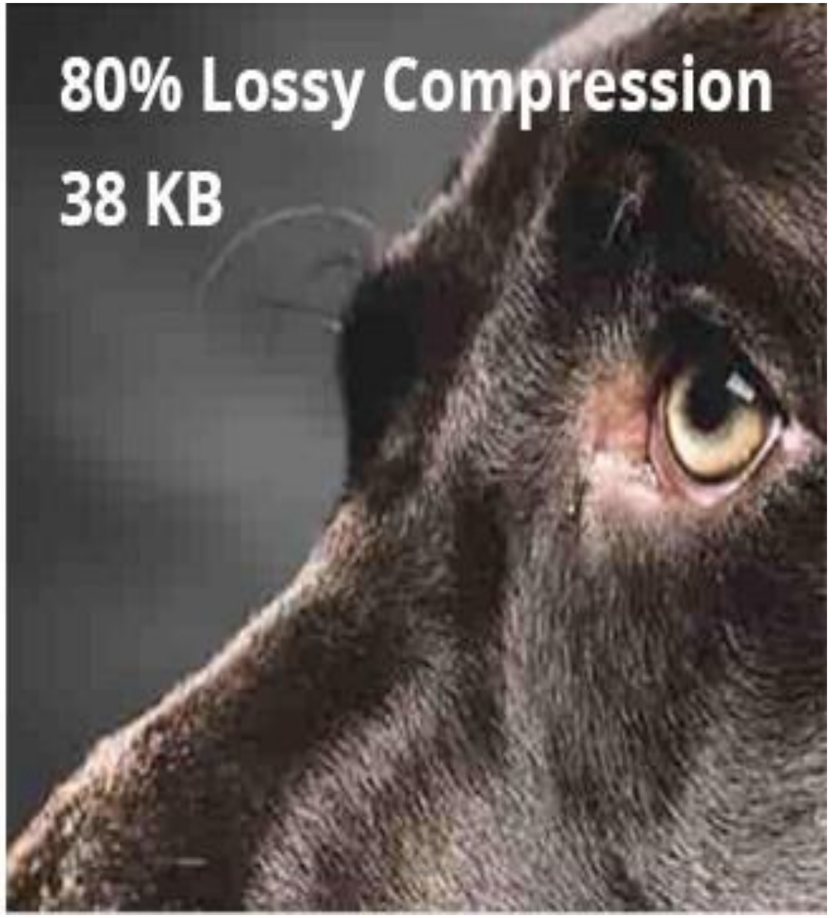
- Lossless data compression is used when the data has to be uncompressed exactly as it was before the compression.
- Generally required for text compression.
- Lossless compression never removes any information from the original file. It relies instead on representing data in mathematical formulas.

ex: WinZip.GIF

Lossy compression

- Lossy compression reduces a file by permanently eliminating certain information, especially redundant or unnecessary information.
- When the file is uncompressed, only a part of the original information is still there.(although the user may not notice it).
- Example : jpeg, bmp, mpeg, mp3, wave

Lossy compression



Codes

- A code is a mapping of source messages into code words.

ex: ASCII code (8 bits per character)

Letter	Binary code	Decimal value
A	01000001	65
B	01000010	66
a	01100001	97
b	01100010	98

Type of Codes

- Fixed Length Coding
- Variable length Coding

Fixed Length Coding(FLC)

- A code in which a fixed number of source symbols are encoded into a fixed number of output symbols.
  - All the binary codes are example for fixed length encoding
  - ASCII
  - Unicode

Unicode

- Unicode is a information technology standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems.
  - Universal character encoding scheme for written characters and text.
  - Using a 16-bit encoding means that code values are available for more than 65,000 characters.
  - Provides the capacity to encode all characters used for the written languages of the world.

Unicode

ASCII/8859-1 Text

A	0100 0001
S	0101 0011
C	0100 0011
I	0100 1001
I	0100 1001
/	0010 1111
8	0011 1000
8	0011 1000
5	0011 0101
9	0011 1001
-	0010 1101
1	0011 0001
	0010 0000
t	0111 0100
e	0110 0101
x	0111 1000
t	0111 0100

Unicode Text

A	0000 0000 0100 0001
S	0000 0000 0101 0011
C	0000 0000 0100 0011
I	0000 0000 0100 1001
I	0000 0000 0100 1001
	0000 0000 0010 0000
天	0101 1001 0010 1001
地	0101 0111 0011 0000
	0000 0000 0010 0000
س	0000 0110 0011 0011
ل	0000 0110 0100 0100
ا	0000 0110 0010 0111
م	0000 0110 0100 0101
	0000 0000 0010 0000
α	0000 0011 1011 0001
≰	0010 0010 0111 0000
γ	0000 0011 1011 0011

Fixed Length Coding(FLC)

- In a fixed-length code each code word has the same length
- If there are n symbols, need **$\lceil \log_2 n \rceil$** bits.
  - Example : - {A,B,C,D,E} need

$$= \lceil \log_2 5 \rceil \text{ bits}$$

$$= \lceil 2.3219 \rceil$$

$$= 3$$

Simple Example (Contd.)

Symbol	Codeword
A	000
B	001
C	100
D	101
R	111

Message is

ABRACADABRA

Then the code is as follows

A	B	R	A	C	A	D	A
000	001	111	000	100	000	101	000

ABRACADABRA= 000001111000100000101000001111000 **(33 bits)**

Variable Length Coding

- In a variable-length code codewords may have different lengths
- Example: codeword for A is a prefix of the codeword for B.
  - use a prefix-free code where no codeword is a prefix of any other codeword.

Variable Length Coding (Contd.)

Symbol	Codeword
A	00
B	100
C	101
D	110
R	111

Symbol	Codeword
A	0
B	100
C	101
D	110
R	111

Variable Length Coding (Contd.)

Message is

ABRACADABRA

Symbol	Codeword	Frequency	Bits need
A	0	5	5
B	100	2	6
C	101	1	3
D	110	1	3
R	111	2	6
		Total bits = 23	

Comparison of different codes

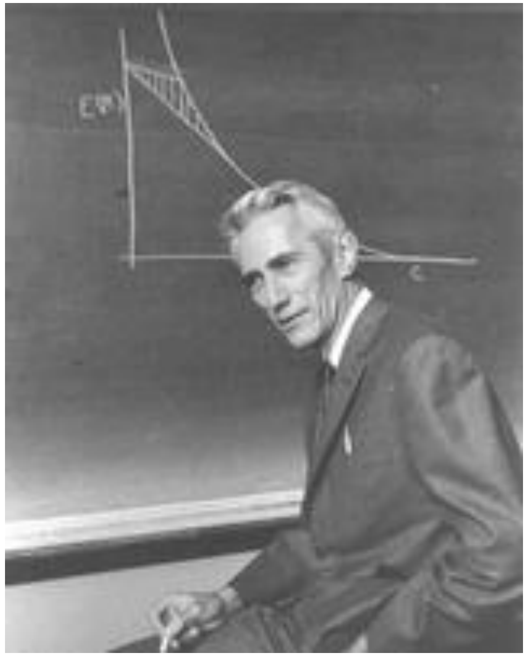
Code	Bits	Bytes
ASCII	88	11
Binary	33	5
A=00	28	4
A=0	23	3
Entropy	22.44	3

Entropy

- Shannon formulated the theory of data compression and established that there is a fundamental limit to lossless data compression.

- This limit, called the **entropy** rate, is denoted by **H**.

- **H** is the smallest number of bits per symbol you can use.



Shannon in 1948

Entropy (Contd.)

Let n be the size of the alphabet and let p_i be the probability of the i th letter in the alphabet. The entropy rate is

$$H = \sum_{i=1}^{n} - p_i \log_2 p_i$$

The entropy rate of English text would have been 4.07 bits/character

Entropy (Contd.)

Symbol	Frequency
A	5
B	2
C	1
D	1
R	2

message is

ABRACADABRA

$$H = -\left(\frac{5}{11}\log_2\frac{5}{11} + \frac{2}{11}\log_2\frac{2}{11} + \frac{2}{11}\log_2\frac{2}{11} + \frac{1}{11}\log_2\frac{1}{11} + \frac{}{11}\log_2\frac{1}{11}\right)$$

$$11H = -\left(5\log_2\frac{5}{11} + 2\log_2\frac{2}{11} + 2\log_2\frac{2}{11} + \log_2\frac{1}{11} + \log_2\frac{1}{11}\right)$$

$$= 22.444$$

H = 2.04

Shannon Fano Code

- Shannon-Fano coding in data compression is a variable-length encoding based on the frequency of each character.
- Shannon-Fano is a minimal prefix code.
- Divide symbols into two groups, with each group having as close to half the total frequency of each.
- Give each group one bit ($-\log_2 0.5$).

Shannon Fano Algorithm.

1. Line up the symbols by falling probability of incidence.
2. Divide the symbols in two groups, so that both groups have equal or almost equal sum of the probabilities.
3. Assign value 0 to the first group, and value 1 to the second.
4. For each of the both groups go to step 2.

Shannon Fano(Contd.)

"EXAMPLE OF SHANNON FANO"

- First, calculate the probabilities of each symbol in the text as follows:

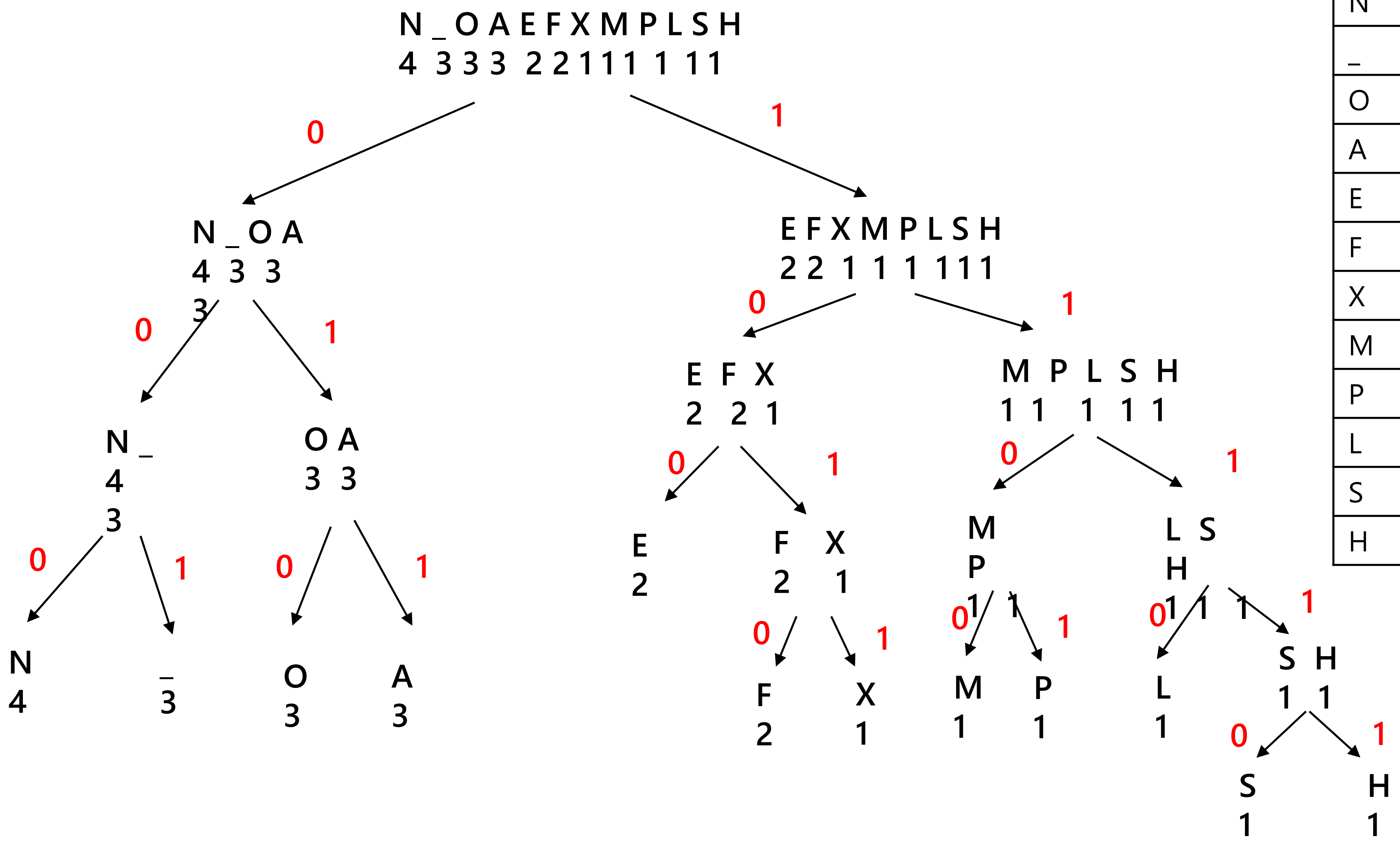
Symbol	Probability
E	2/23
X	1/23
A	3/23
M	1/23
P	1/23
L	1/23
O	3/23
F	2/23
S	1/23
H	1/23
N	4/23
_	3/23

Shannon Fano(Contd.)

- **Line up the symbols.**

Symbol	Probability
N	4/23
_	3/23
O	3/23
A	3/23
E	2/23
F	2/23
X	1/23
M	1/23
P	1/23
L	1/23
S	1/23
H	1/23

Shannon Fano Coding Tree



N	000
_	001
O	010
A	011
E	100
F	1010
X	1011
M	1100
P	1101
L	1110
S	11110
H	11111

Huffman Coding

- Huffman encoding, an elegant algorithm invented by David Huffman in 1952
- The concept is based on replacing strings that appear most frequently by shorter strings even thought strings that appear less frequently by longer strings.
- Variable code word length.
- Uniquely decodable

Huffman's Algorithm

1. Line up the symbols by increasing probabilities.(Ascending order)

2. Link two symbols with least probabilities into one new symbol which probability is a sum of probabilities of two symbols.

3. Go to step 2. Until you generate a single symbol which probability is 1

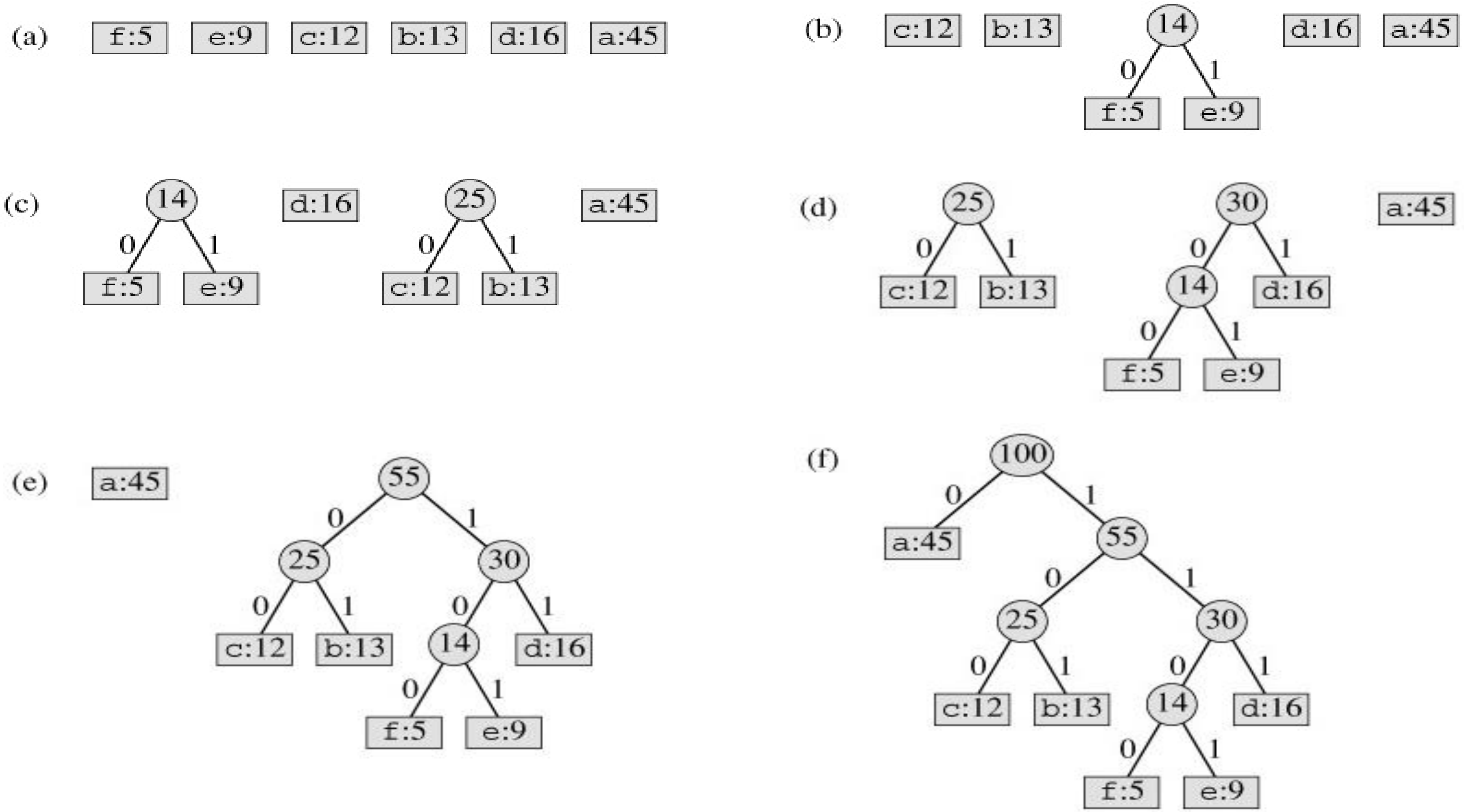
4. Trace the coding tree from a root (the generated symbol with probability 1) to origin symbols.

Constructing a Huffman code

```
HUFFMAN(C)
n ← |C|
Q ← C
for i 1 to n - 1
    do allocate a new node z
        left[z] ← x ← EXTRACT-MIN (Q)
        right[z] ← y ← EXTRACT-MIN (Q)
        f [z] ← f [x] + f [y]
        INSERT(Q, z)
return EXTRACT-MIN(Q)  ▹ Return the root of the tree.
```

Example :

Symbol	a	b	c	d	e	f
Probabilities	0.45	0.13	0.12	0.16	0.09	0.05



Symbol	Codeword
A	0
B	101
C	100
D	111
E	1101
F	1100

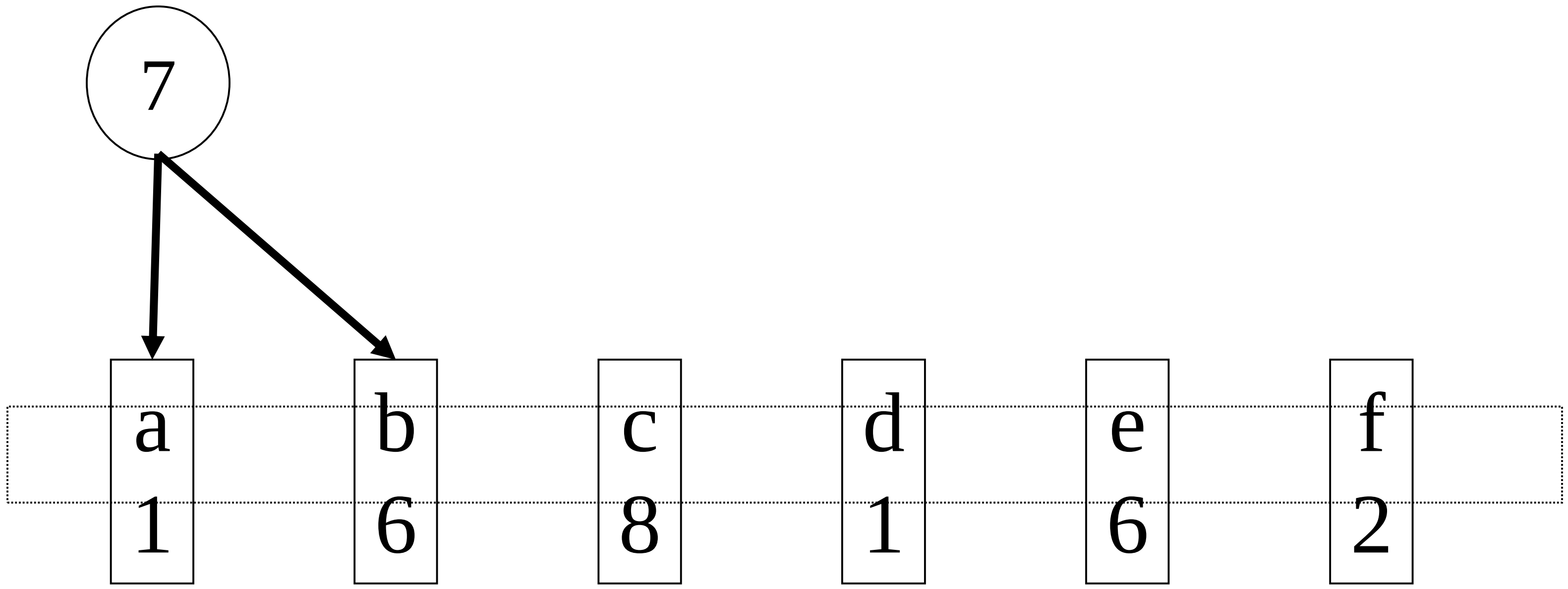
Hu – Tucker Algorithm

- More like Huffman, but sub-trees must replace their left child in a sorted list.

- Only allowed to join sub-trees that are not separated by leaves.

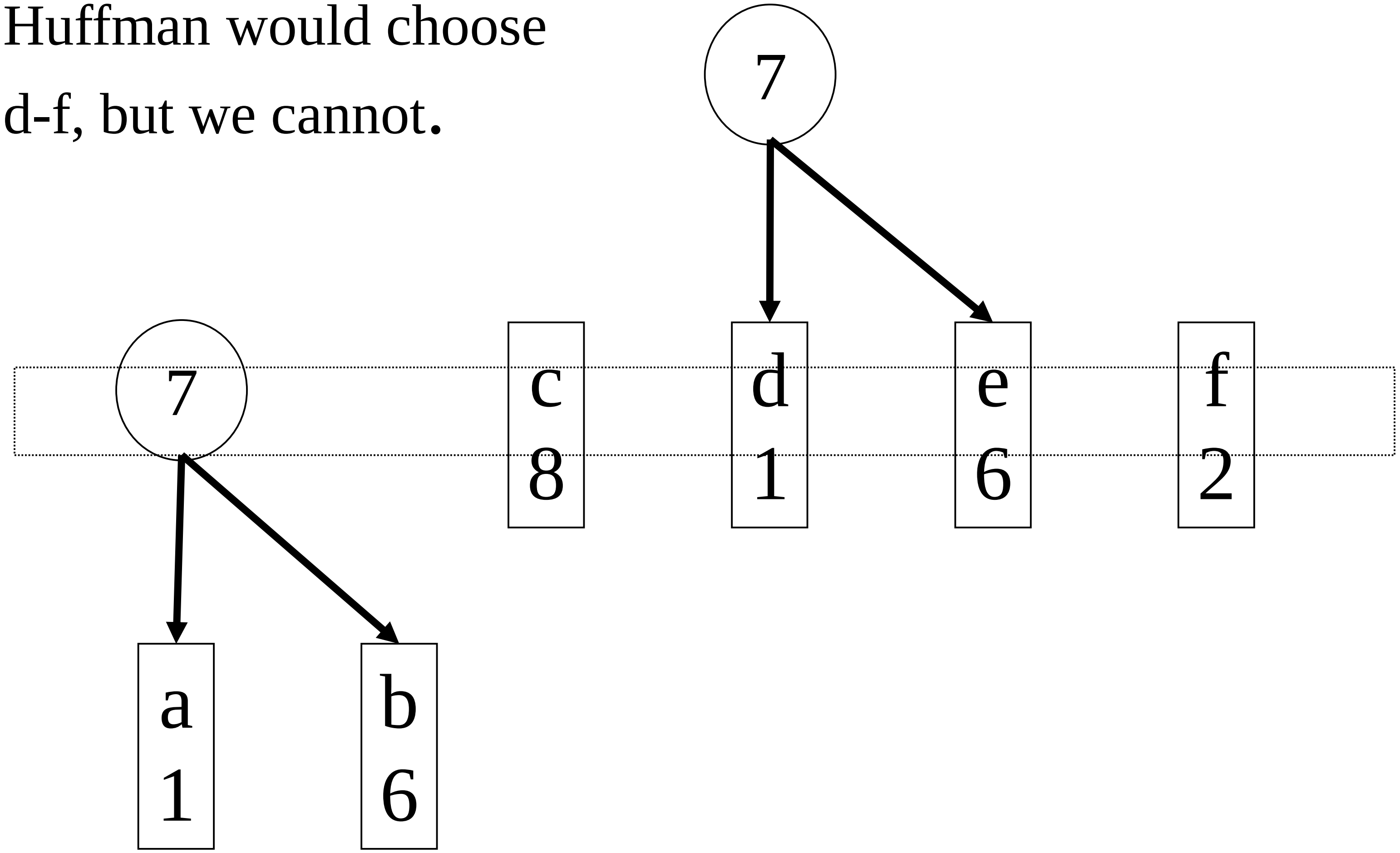
Hu – Tucker Algorithm

1. Huffman’s algorithm says join a-d, but they are separated by a leaf, so we cannot.
2. Either a-b or d-e are available.
3. Choose the leftmost.

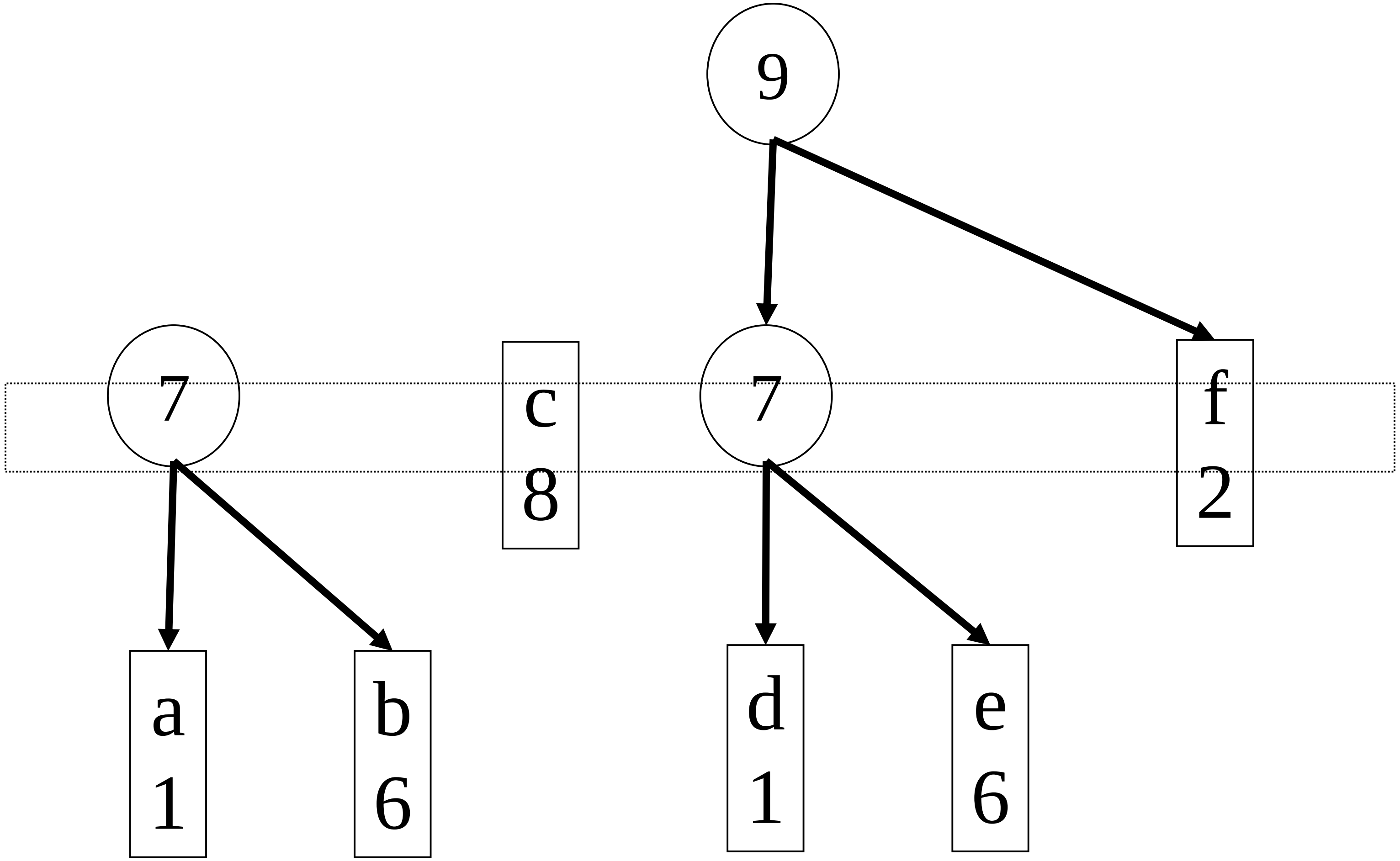


Hu – Tucker Algorithm

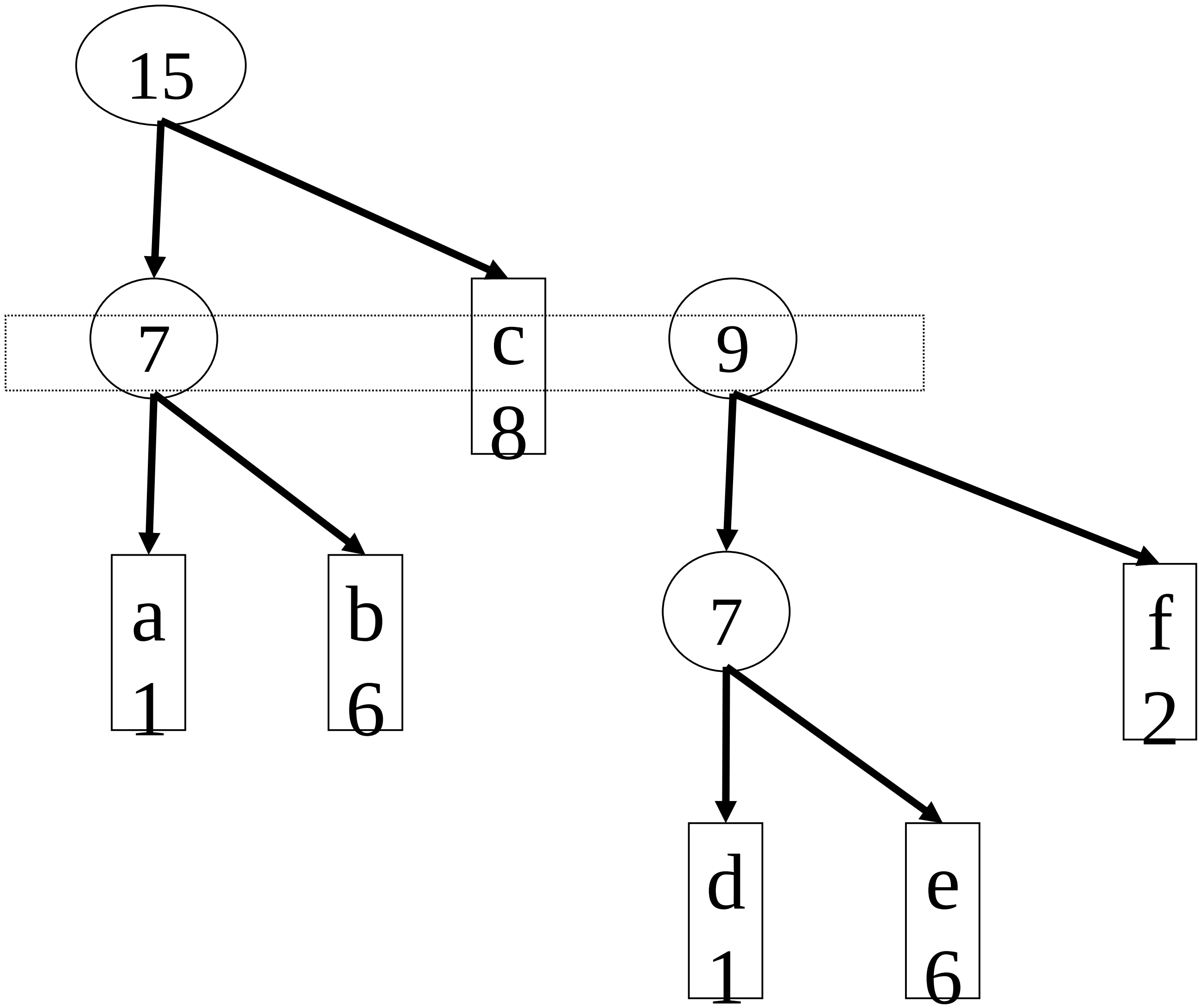
Huffman would choose d-f, but we cannot.



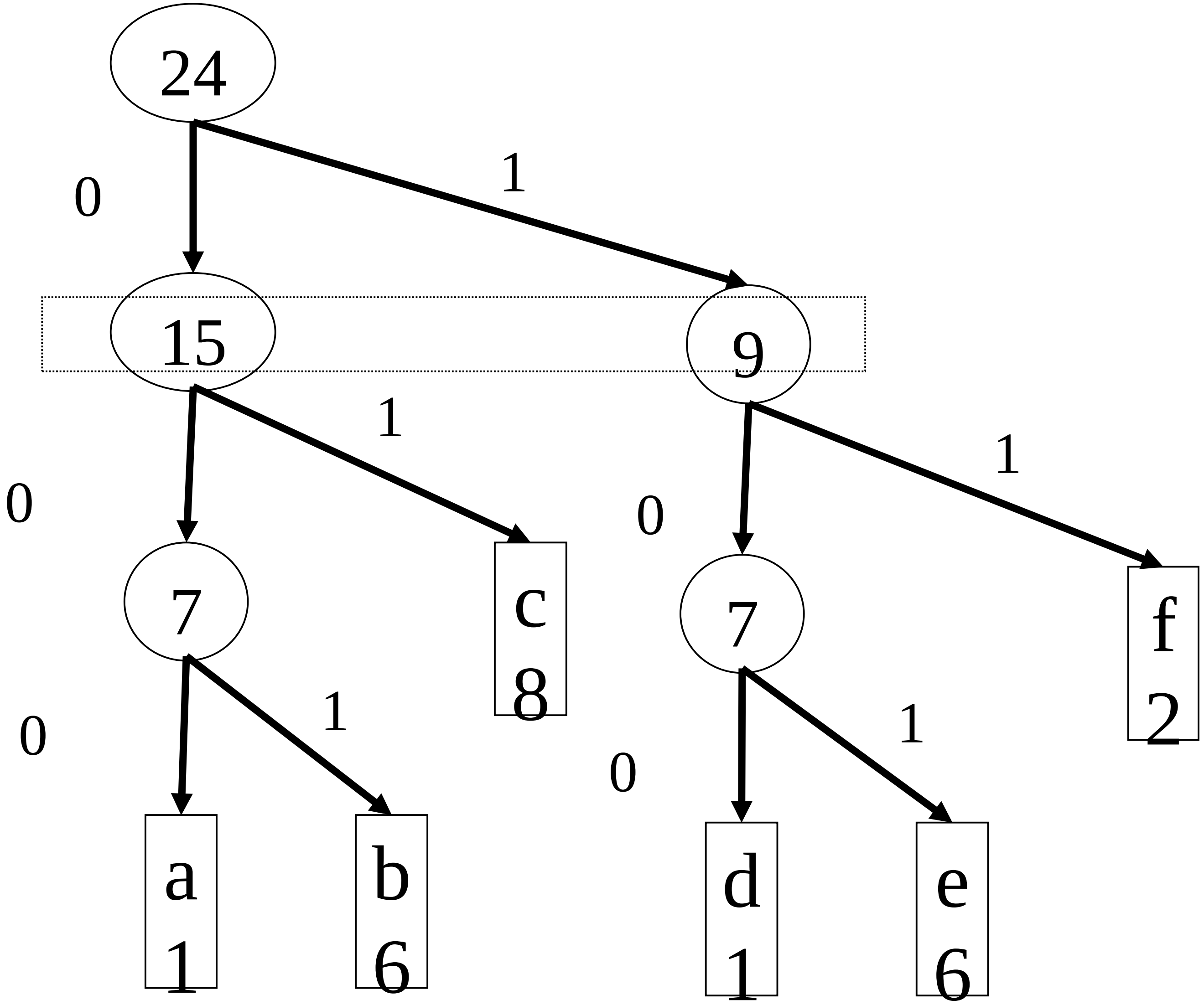
Hu – Tucker Algorithm



Hu – Tucker Algorithm



Hu – Tucker Algorithm



a	000
b	001
c	01
d	100
e	101
f	11

Modeling

- The algorithms discussed so far are coding algorithms. Before do the coding we must know the frequency of occurrences of symbols.

- Parsing and estimating frequencies is the job of modeling algorithms.

Ziv – Lempel model

- Consider a message as a window of text.
- Represent the current text as a pointer back into the window
- Need to know how far back to point
- How many characters to copy
- Used as the model in gzip
- Huffman code used for coding in gzip

Example

- Suppose we have the message:

ABRACADABRA

How we are going to compress them using this method?

ABRACADABRA

	Distance Back	Num to Copy or ASCII code
A	0	65
AB	0	66
ABR	0	82
ABRA	3	1
ABRAC	0	67
ABRACA	2	1
ABRACAD	0	68
ABRACADABRA	7	4

Ziv – Lempel model - Gzip

- So ABRACADABRA gets transformed into the symbol stream
  (0,65) (0,66) (0,82) (3,1) (0,67) (2,1) (0,68) (7,4)

- Two Huffman codes are then built on the frequency counts of
  - distances (including the 0s which signal a new character)
  - copy lengths

Ziv – Lempel model - Gzip eg (cont.)

- So in our example we have

Distance	Freq	Codeword
0	5	0
2	1	10
3	1	110
7	1	111

Length	Freq	Codeword
1	2	0
4	1	1

- A = 65
- B = 66
- C = 67
- D = 68
- E = 69
- F = 70
- G = 71
- H = 72
- I = 73
- J = 74
- K = 75
- L = 76

M = 77
N = 78
O = 79
P = 80
Q = 81
R = 82
S = 83
T = 84
U = 85
V = 86
W = 87
X = 88
Y = 89
Z = 90

a = 97 b = 98 c = 99 d = 100 e = 101 f = 102 g = 103 h = 104 i = 105 j = 106 k = 107 l = 108 m = 109 n = 110 o = 111 p = 112 q = 113 r = 114 s = 115 t = 116 u = 117 v = 118 w = 119 x = 120 y = 121 z = 122

Summery:

- What is data compression?
- How to compress data
- Compression Types
- Entropy
  - Shanon Fano Algorithm
  - Huffman Algorithm
  - Hu Tucker Algorithm
  - Modeling and Coding.


SEE YOU
next
week!