CSc 461/561
Multimedia Systems
Lossless Compression

Jianping Pan

Spring 2015

First things first

- A1 posted on connex
  - due on Friday, Jan 23
  - any questions?
- Project ideas?
  - anything related to multimedia systems
  - survey (and evaluate) it for 461 (561)
  - group by 3 (or 2) or individual for 461 (561)
  - email me by Monday, Jan 26
    - [csc461] or [csc561] on subject line

Compression

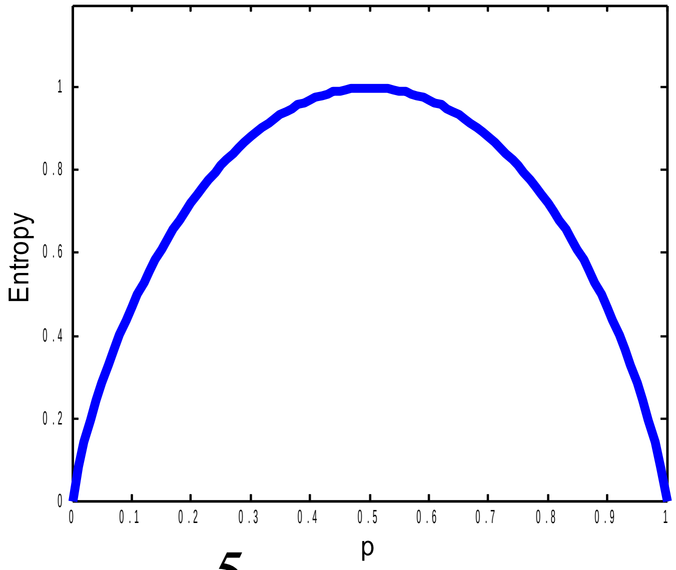
- Why compression?
  - there is (a lot of) redundancy!
- How to compress?
  - remove data *and* information redundancy
- Lossless compression
  - without information loss
- Lossy compression

Compressibility

- Compression ratio
  - B_0: # of bits to represent before compression
  - B_1: # of bits to represent after compression
  - compression ratio = B_0/B_1
- Entropy: a measure of uncertainty; min bits
  - alphabet set $\{s_1, s_2, \ldots, s_n\}$
  - probability $\{p_1, p_2, \ldots, p_n\}$
  - entropy: $- p_1 \log_2 p_1 - p_2 \log_2 p_2 - \ldots - p_n \log_2 p_n$

Entropy examples

- Alphabet set {0, 1}
- Probability: {p, 1-p}
- Entropy: $H = - p \log_2 p - (1-p) \log_2 (1-p)$
  - when p=0, H=0
  - when p=1, H=0
  - when p=1/2, $H_{max}=1$
    - 1 bit is enough!



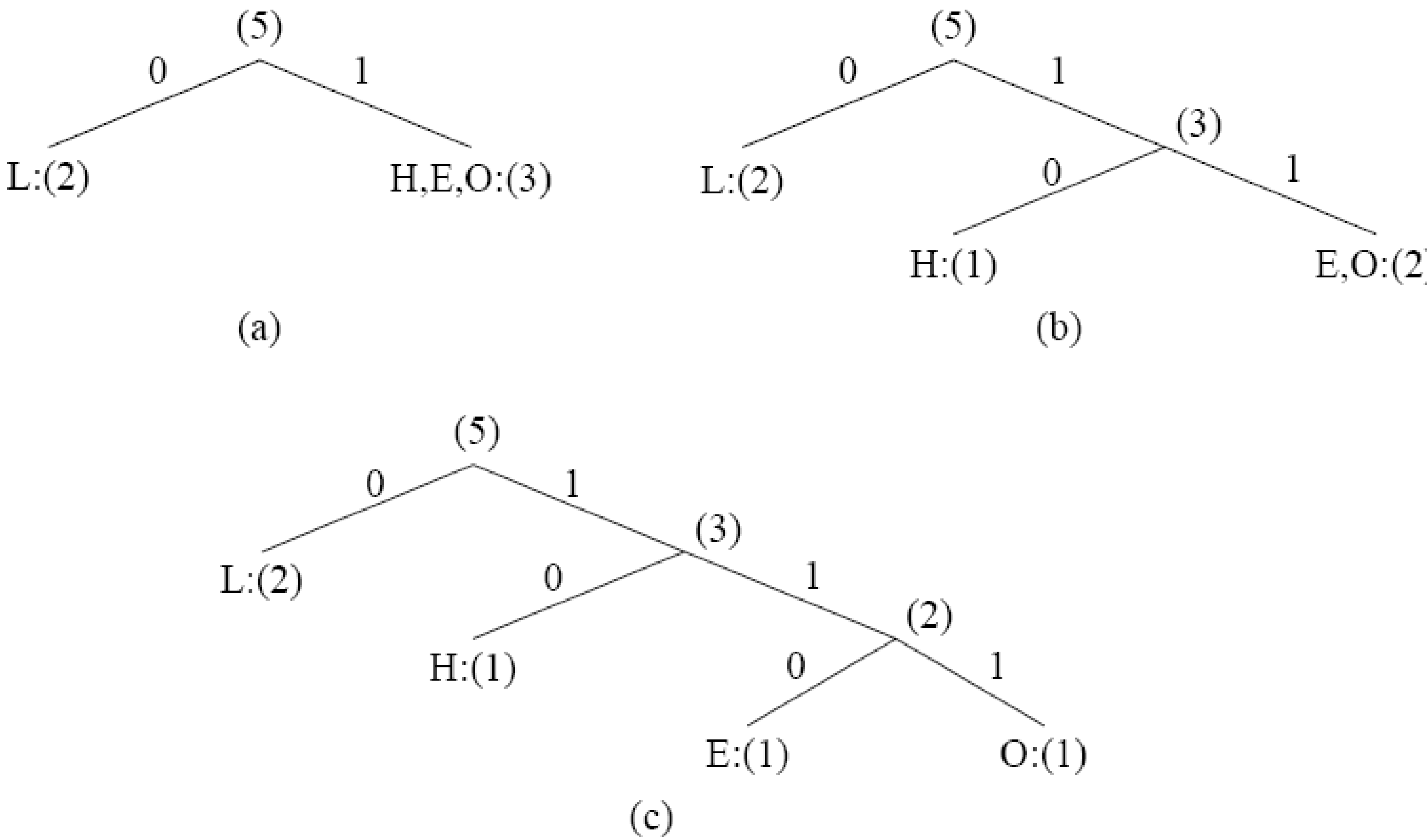
Shannon-Fano algorithm

- Fewer bits for symbols appear more often
- "divide-and-conquer"
  - also known as "top-down" approach
  - split alphabet set into subsets of (roughly) equal probabilities; do it recursively
  - similar to building a binary tree

Symbol	H	E	L	O
Count	1	1	2	1

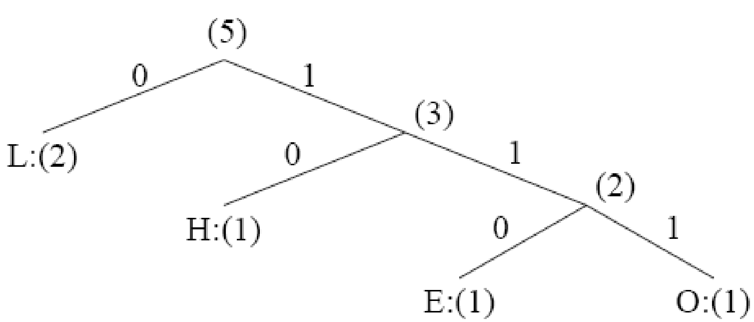
Frequency count of the symbols in "HELLO"

Shannon-Fano: examples



Shannon-Fano: results

- Prefix-free code
  - no code is a prefix of other codes
  - easy to decode

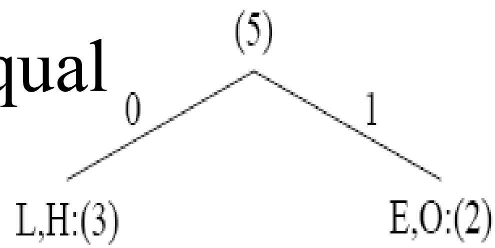


Symbol	Count	$\log_2 \frac{1}{p_i}$	Code	# of bits used
L	2	1.32	0	2
H	1	2.32	10	2
E	1	2.32	110	3
O	1	2.32	111	3
TOTAL number of bits:				10

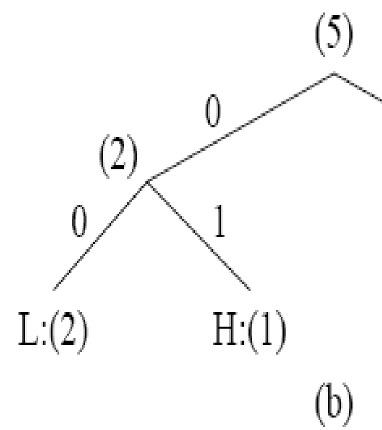
* what if {0.4, 0.3, 0.2, 0.1}

Shannon-Fano: more results

- Encoding is not unique
  - *roughly* equal

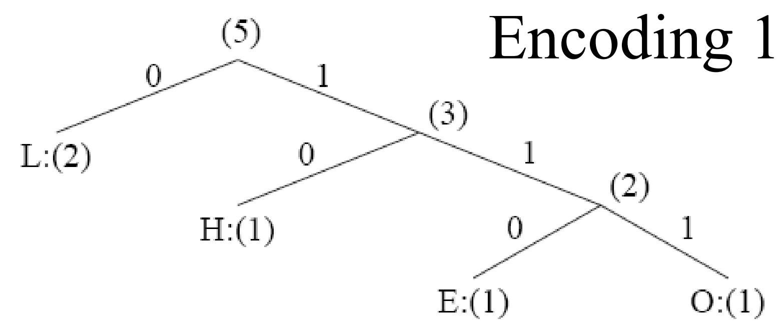


Encoding 2



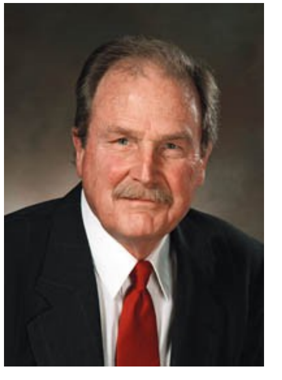
Symbol	Count	$\log_2 \frac{1}{p_i}$	Code	# of bits used
L	2	1.32	00	4
H	1	2.32	01	2
E	1	2.32	10	2
O	1	2.32	11	2
TOTAL number of bits:				10

Encoding 1



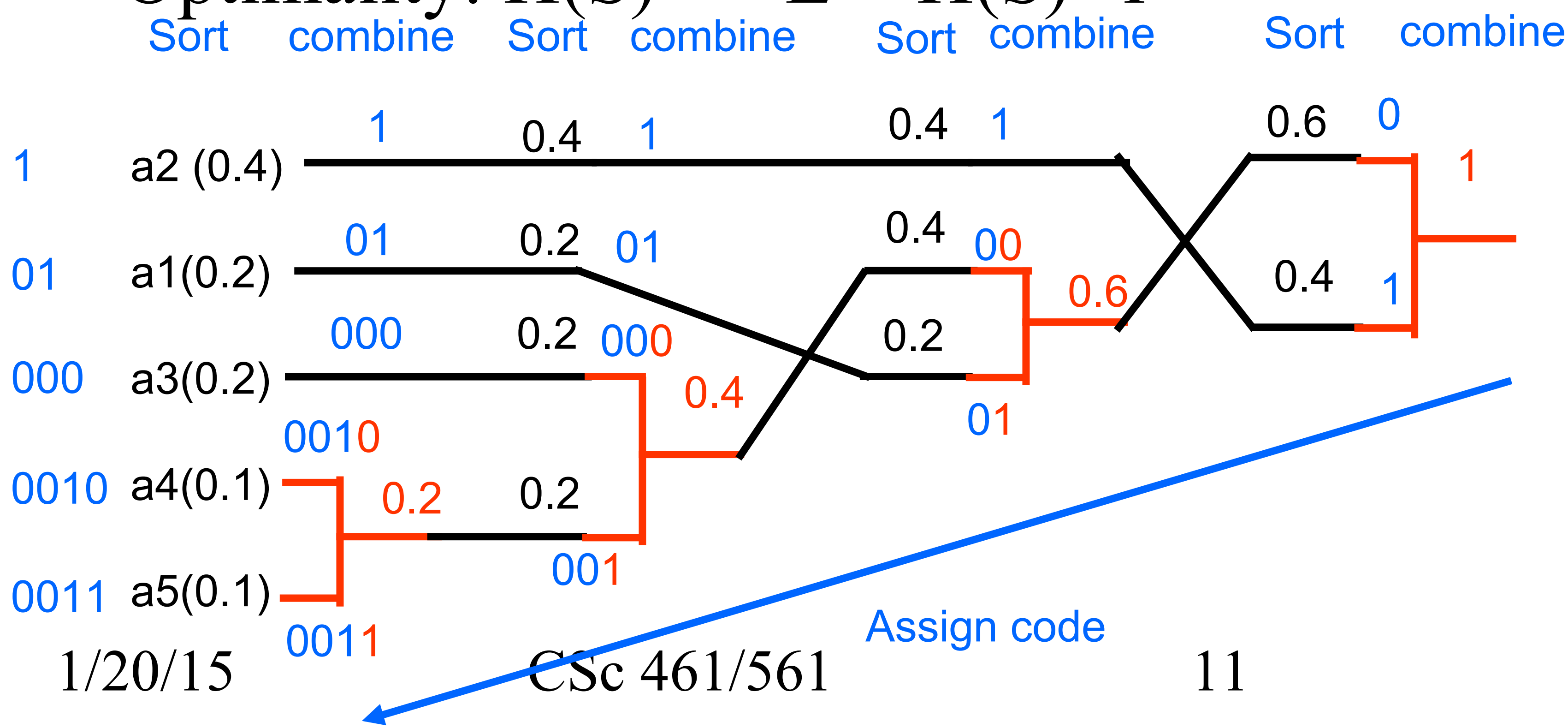
Huffman coding

- "Bottom-up" approach
  - also build a binary tree
    - and know alphabet probability!
  - start with two symbols of the least probability
    - s_1: p_1
    - s_2: p_2
    - s_1 or s_2: p_1+p_2
  - do it recursively



Huffman coding: examples

- Encoding not unique; prefix-free code
- Optimality: H(S) <= L < H(S)+1



Run-length coding

- Run: a string of the same symbol
- Example
  - input: AAABBCCCCCCCCCAA
  - output: A3B2C9A2
  - compression ratio = 16/8 = 2
- Good for some inputs (with long runs)
  - bad for others: ABCABC
  - how about to treat ABC as an *alphabet*?

LZW compression

- Lempel-Ziv-Welch (LZ77, W84)
  - Dictionary-based compression
  - no a priori knowledge on alphabet probability
  - build the *dictionary* on-the-fly
  - used widely: e.g., Unix compress
- LZW coding
  - if a word does not appear in the dictionary, add it
  - refer to the dictionary when the word appears again

LZW examples

- Input
  - ABABBABCABABBA
- Output
  - 1 2 4 5 2 3 4 6 1

code	string
1	A
2	B
3	C

s	c	output	code	string
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	A	4	10	ABA
A	B			
AB	B			
ABB	A	6	11	ABBA
A	EOF	1		

This lecture

- Lossless compression
  - entropy
  - Shannon-Fano algorithm
  - Huffman coding
  - LZW compression
- Explore further
  - decoding: Shannon-Fano, Huffman, LZW
  - arithmetic coding [Ref: Li&Drew 7.6]

Next lecture

- Multimedia manipulation
  - lossy compression [Ref: Li&Drew Chap 8]
    - rate vs distortion [8.2-3]
    - quantization: uniform vs non-uniform [8.4.1-2]
    - discrete cosine transform [8.5.1]