image compression:
- lossless - lossy

lossless compression is important for
- medical diagnosis - legal reasons
- satellite imagery - archival
- high quality imaging

* general file compression

to measure performance, we can use the compression ratio: $\frac{\text{\# of bits before}}{\text{\# of bits after}}$

for real images, lossless compression ratios in the range 2-10

recall the image histogram, thinking about it as a PMF(probability mass function)

$$p_i = P(I(x,y)=i) = \frac{\#\ I(x,y)=i}{\#\ \text{pixels}}$$

the average bits per pixel:

$$L_{av} = \sum_{i=1}^{N} p_i \cdot b_i = E(b)$$

uniform coding, $b_i = b$ (eg, 8 bits)

idea: use fewer bits to describe more frequent symbols (eg. Morse code)

consider an 8-level image

level(i)	p	unif code	variable-length code
0	0.19	000	00
1	0.25	001	11
2	0.21	010	01
3	0.16	011	101
4	.08	100	1001
5	.06	101	10001
6	.03	110	100001
7	.02	111	100000
		$L_{av}=3$	$L_{av}=2.7$

compaction ratio = 3/2.7 = 1.11

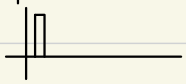
what's the best we can do: related to entropy of the PMF

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

entropy measure how "uncertain" a random variable is

worst case: uniform

best case: delta function



uniform distribution:

$$p_i = \frac{1}{N} \qquad H = -\sum_{i=1}^{N} p_i \log_2 p_i = -\sum_{i=1}^{N} p_i \log_2 p_i$$

$$= N \cdot (-1/N \log_2 1/N) = \log_2 N$$

if N = 8, H =3, the worst case

Delta function: $p_1 = 1,\ p_2 = \cdots = p_N = 0$

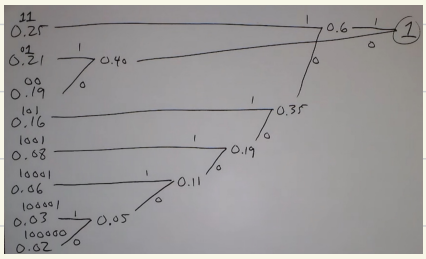
$$H = -1 \log_2 1 = 0$$

the entropy is the lowest number of average bits per symbol that can be used to code the distribution

How to design a code that approaches this value?

Huffman coding: optimal(minimum Lav) for independent coding N symbols

1) arrange symbols in decreasing p_i (think of as nodes/leaves of a tree)
2) merge the 2 nodes w/ lowest probability
3) assign 0/1 to top/bottom branch
4) read code from root-to-leaf



Bonus: the prefix code is a prefix code
no codeword is a prefix of any other codeword

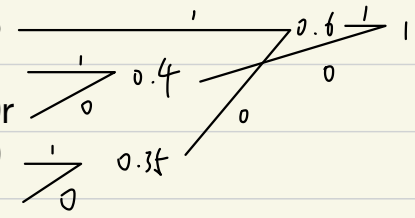
code A (not prefix)		code B (prefix)	
1	10	1	0
2	00	2	10
3	11	3	110
4	110	4	111

1234321:
A: 1000111110110010
B: 010110111110100
prefix code can be decoded instantaneously

can generally do better by considering pairs of symbol (or more)
truncated Huffman coding:
to avoid extremely long code words for infrequent symbols,
- Huffman code the most probable k symbols in the source
- replace the rest with a prefix + a fixed length code

Ex:

0.25 0.25
0.21 => 0.21
0.19 0.19r
0.16 0.19
0.19 the rest: 0.16



0.25 11 the rest 0.08 00 00
0.21 01 0.06 00 01
0.19 101 0.03 00 10
0.16 100 0.02 00 11
Lav = 2.75, true Huffman Lav = 2.65

Lempel-Ziv coding: basis for gif, tiff, png, zip
- parse source string sequentially into “phrases”: string that haven’t appeared before
- code each phrase by giving location of the prefix (fixed-length code) plus last bit

- no knowledge of PMF for symbols
- exploits how symbol occur together
- really pays off for very long strings

run-length encoding (RLE) fax
simple: code the number of 0’s between successive 1’s
or: code the length of continuous black/ white runs