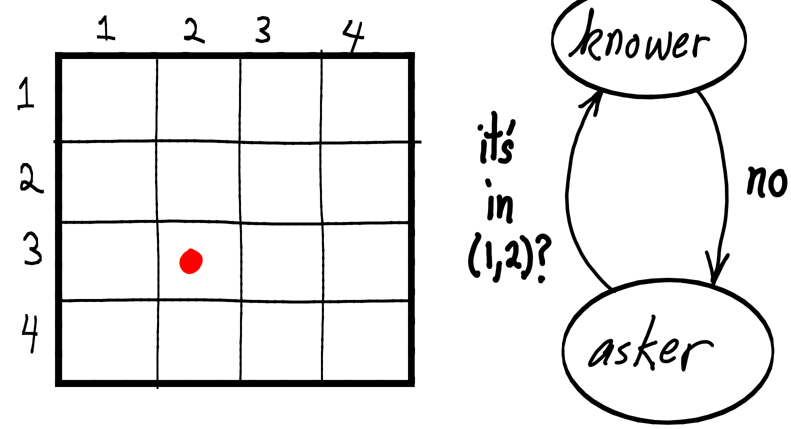
CODING and INFORMATION
We need encodings for data.

Where's the ball?



The Info Game

--- Knower knows where ball is.

--- Asker wants to know where it is.

--- Only ask YES/NO questions.

ARE ALL questions equally informative?

--- What's the MIN number of questions?
  - --- average case?
  - --- worst case?
  - --- best case?

--- Is there a good series of questions?

--- How much information does an answer give?

Suppose equally likely in each box. Ask, "in (x,y)"?
What's expected number of questions?

P(Hit 1st) = 1/16

P(Hit 2nd) = P(Hit 2nd | Miss 1st) P(Miss 1st) = (1/15) (15/16) = 1/16

P(Hit 3rd) = (1/14) * P(Miss 2nd and 1st) = (1/14) (14/15) (15/16) = 1/16

E(n) = 1*(1/16) + 2*(1/16) + ... + 15*(1/16) + 15(1/16) = (1+2+3+...+15+15) / 16 ~ 8 1/2

Does that mean there are ≈ 8 bits of information?

What if we asked questions so that the answer is 50/50 yes/no each time?

P(Hit 1st) = 1/2 (1/2 the boxes eliminated, 8 boxes left)
P(Hit 2nd) = 1/2 (1/2 the remaining boxes eliminated, 4 boxes left)
P(Hit 3rd) = 1/2 (1/2 the remaining boxes eliminated, 2 boxes left)
P(Hit 4th) = 1/2 (1 box left, we know the answer)

Always takes 4 questions. $\longrightarrow$ 4 bits of information?

Prob $= 1/2 \rightarrow$ 1 bit ? amount of information is $-\log_2(p) = 1$ bit? per question

How about sending actual bits?

There are 16 boxes, label each w/ 4 bits.

question = "Is next bit 0?" $\quad \mathrm{Prob}(\text{yes}) = \mathrm{Prob}(\text{no}) = 1/2$

4 bits sent, info is 4 bits.

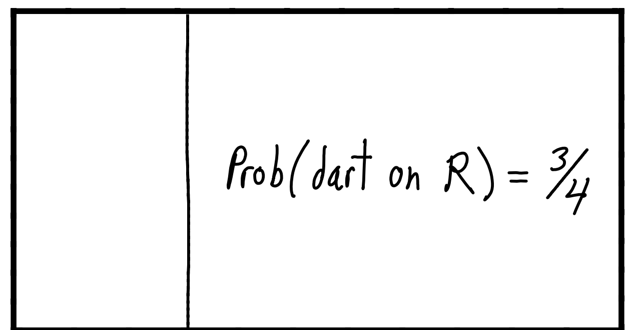
How much context information must we share for this to work?

Ordering of boxes?
Which bit comes first?
...

Game: Darts Info

Throw dart ====> uniform probability
Set divider so that 1/4 of area is on left side.

Ask, "Dart on Left side?"



$$\mathrm{Prob}(\text{yes}) = 1/4 \rightarrow -\log(2^{-2}) = 2 \text{ bits}$$

$$\mathrm{Prob}(\text{no}) = \quad \rightarrow -\log(3/4) \cong 1/2 \text{ bit}$$

question = "is dart left of here?"

Expected number of bits of info per question?

$$E = (2 \text{ bits})\,\mathrm{Prob}(2 \text{ bits}) + (1/2 \text{ bit})\,\mathrm{Prob}(1/2 \text{ bit})$$

$$= (2 \text{ bits})(1/4) + (1/2 \text{ bit})(3/4) = 7/8 \text{ bit}$$

Extreme Case **Let's see what happens if we move divider far to left.**

$$\mathrm{Prob}(\text{yes}) = 1/2^{10} \qquad \mathrm{Prob}(\text{no}) = 1 - 1/2^{10}$$

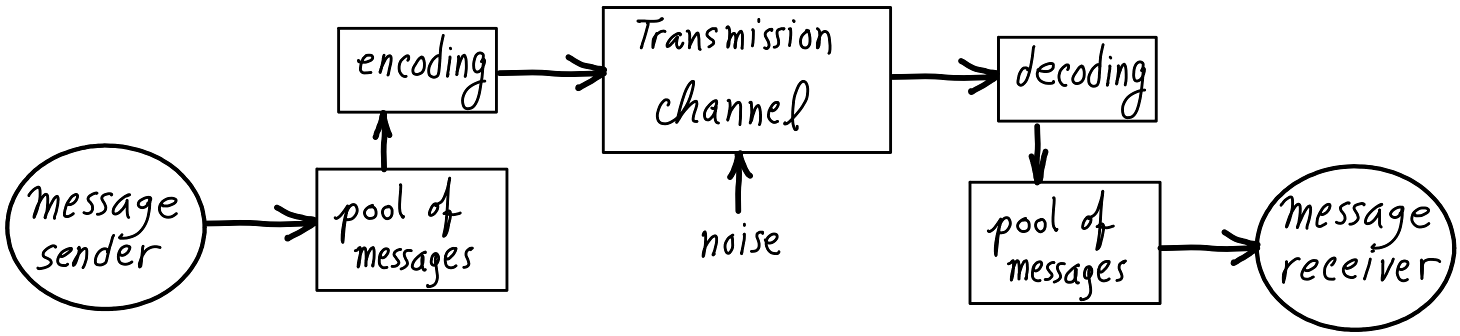
$$E = -\log(1/2^{10})(1/2^{10}) + -\log(1 - 1/2^{10})(1 - 1/2^{10})$$

$$= (10 \text{ bits})(2^{-10}) + (\sim 0 \text{ bits})(\sim 1)$$

$$\cong 1/100 \text{ bits} \quad \text{per question}$$

Thm E is max if $\mathrm{Prob}(\text{yes}) = \mathrm{Prob}(\text{no})$

Shannon information (entropy H) = (Expected (Avg.) bits of information, per message) = $E(-\log \text{Prob})$



Shared Context:

Pool of Messages
Encoding/Decoding Method

The number of bits sent across channel cannot be less than amount of information in stream of messages.

Suppose our pool of messages is { a, b, c, d }.

Suppose the probabilities of sending/receiving are,

Prob(a) == 0.1 Prob(b) == 0.4 Prob(c) == 0.2 Prob(d) == 0.3

Two questions:

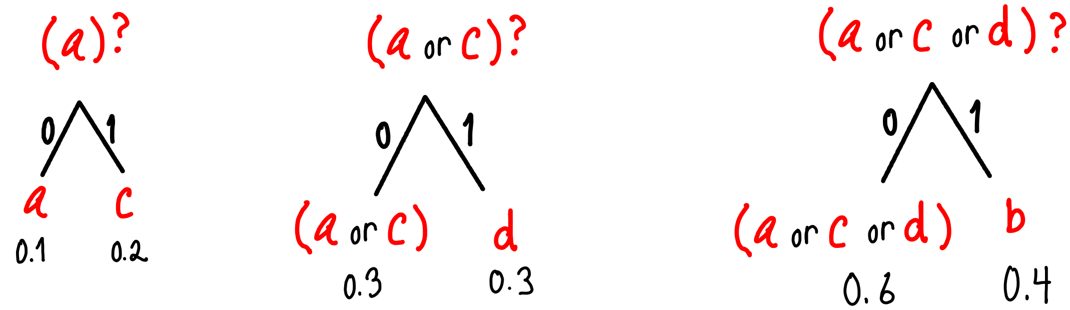
1. What is the average information bit rate?

2. How can we encode messages to approximate that minimum bit rate?

Shannon Information Theory is also called

"Shannon Coding Theory"
"Shannon Minimum Compression Theory"

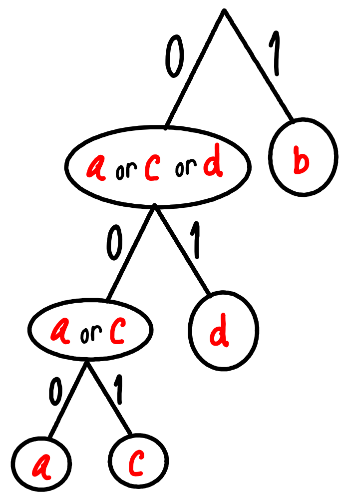
<u>Huffman Coding</u> code by pairing least likely messages to get 50/50 form question = "Is message a or c?" e.g.



Can you decode messages?

What if receive 5 messages?

Where does one message begin and end?



code	message	Prob
000	a	0.1
001	c	0.2
01	b	0.3
1	d	0.4

$$H = -\Big[\underset{a}{0.1\log(0.1)} + \underset{b}{0.4\log(0.4)} + \underset{c}{0.2\log(0.2)} + \underset{d}{0.3\log(0.3)}\Big]$$

$$= \big[0.33 + 0.53 + 0.46 + 0.52\big] = 1.84$$

information rate of sender = 1.84 bits per message

How'd we do?

$$\boxed{\text{Avg. \#bits sent, using our code.}} = (0.1)\overset{000}{3} + (0.4)\overset{1}{1} + (0.2)\overset{001}{3} + (0.3)\overset{01}{2}$$

Pretty good!

$$= 0.3 + 0.4 + 0.6 + 0.6 = 1.9$$

bit rate through channel = 1.9 bits per message

We are sending more bits than information content, but we are very close.

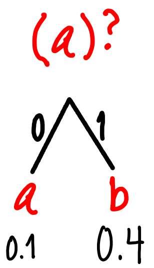
MIN-Length code ==> MAX compression ==> most info bits in least number of communicated bits.

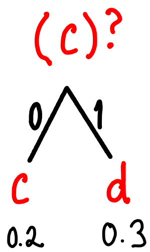
Suppose n different "messages" to send, n = 2^k.
Maximum entropy => equally likely: Prob(message-i) = (1/n) for any message-i.

Expected information per message is,

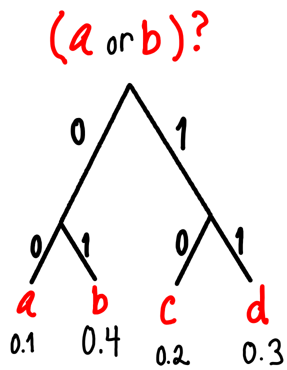
Sum[- (1/n) log[1/n]] = - n (1/n log[1/n]) = -1 log[2^-k] = -1 (-k) = k bits per message. If we use a k-bit code for our messages, we will be 100% compressed. (k-bit integers? Are they equally likely?)

Is that the only code that works?





Change code to have fixed number of bits?



code	message	Prob
00	a	0.1
01	b	0.4
10	c	0.2
11	d	0.3

How'd we do?

$$\boxed{\text{Avg. \#bits sent, using this code.}} = (0.1)\overset{00}{2} + (0.4)\overset{01}{2} + (0.2)\overset{10}{2} + (0.3)\overset{11}{2}$$

Pretty good!

$$= 0.2 + 0.8 + 0.4 + 0.5 = 1.9$$

bit rate through channel = 1.9 bits per message

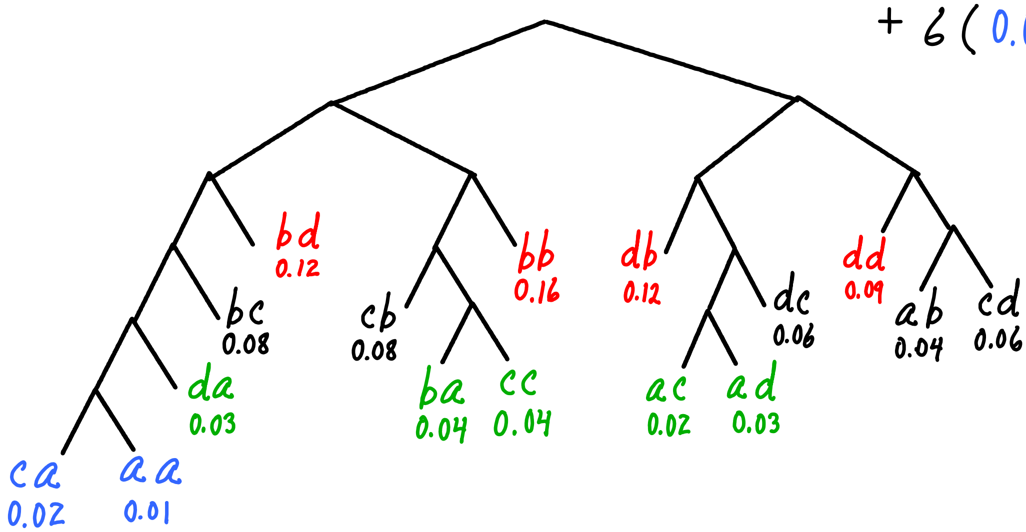
Huffman Algorithm Code: guaranteed to minimize bit rate.

Are there other codes? YES. Are they more compressed? NO.

Is there anything else we can try? Pairs of messages?

$$\text{bit rate per 2 chars} = 3(0.12 + 0.16 + 0.12 + 0.09) + 4(0.08 + 0.08 + 0.06 + 0.04 + 0.06) + 5(0.03 + 0.04 + 0.04 + 0.02 + 0.03) + 6(0.02 + 0.01) = 3.73$$

$\rightarrow$ 1.865 per char



What happens as the number of chars goes up?

Run Length Encoding

File is series of alternating runs of 0s and 1s.

Keep only length of each run.

file of 51 bits

0000000111111100
00010100000000111
11111111100000000

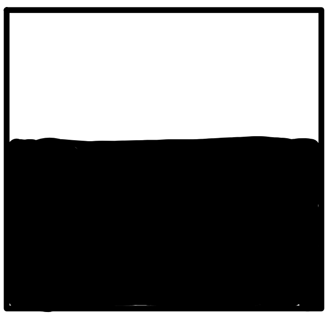
$\rightarrow$ 7, 7, 6, 1, 1, 1, 8, 13, 8

9 integers.
10 digits.
9 commas?

How many bits per digit? Lets say 4.
How many bits per comma? 4 also?

====> 76 bits Hmmmm.

100 x 100 bit-map image file



10,000 bits

1 = black 0 = white

$\rightarrow$ 5000, 5000

9 characters @ 4 bits ===> 36 bits, Wow! Lossless compression.

Hey, hold on there. What's Shannon's H?

$\text{Prob}(0) = \text{Prob}(1) = 1/2 \qquad -\mathbf{H} = (1/2)\log(1/2) + (1/2)\log(1/2)$

$\mathbf{H} = 1$ bit per image bit?

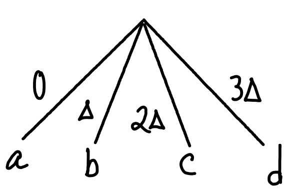
$\text{Prob}(\{0\}^{5,000}) = \text{Prob}(\{1\}^{5,000}) = 1/2$

$\mathbf{H} = 1$ bit per 1/2 entire image?

H assumes independence between messages. Here, lots of dependence. Also, not enough messages.

Δ = Smallest detectable voltage difference

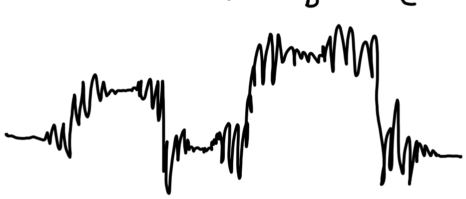
$\rightarrow$



$\{a, b, c, d\} \xrightarrow{\text{coded}} \{0, \Delta, 2\Delta, 3\Delta\}$

How many bits per message? 1 bit?

How fast can you send changes in voltage?



switching creates noise.

Error Detection / Correction

"message" could be a bit, a string of bits, a character, a page of characters, ...

"MESSAGE" → Communication channel → "MASSAGE"   Pink Panther

message coded in 1 bit

"1" → encode → 1 → channel → 0 → decode → "0"

1-bit Error

not detectable

2-bit encoding

"1" → encode → 11 → channel → 10 → decode → "Error"

Code words: 00 and 11 --- codes for "0" and "1"
Code words: 10 and 01 --- signals a 1-bit error (odd parity)
k-bit messages w/ 1 parity bit --- detects 1-bit errors

What if 2-bit error?

Hamming Distance == number of hypercube edges

message	code
"0"	000
"1"	111

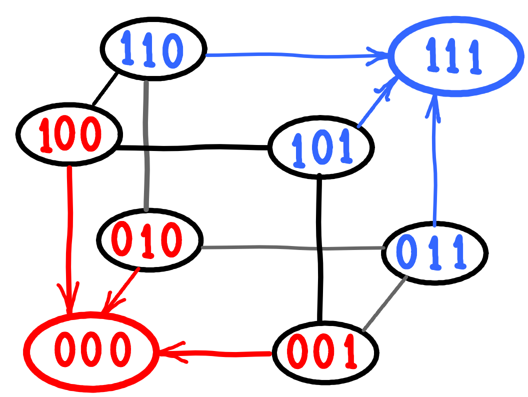
Distance between code words == 3

1-bit error ===> distance == 1

1-bit error CORRECTED :-)

2-bit error not detected :-(

Select code words at distance > 3?



1-bit Correction, 2-bit Detection

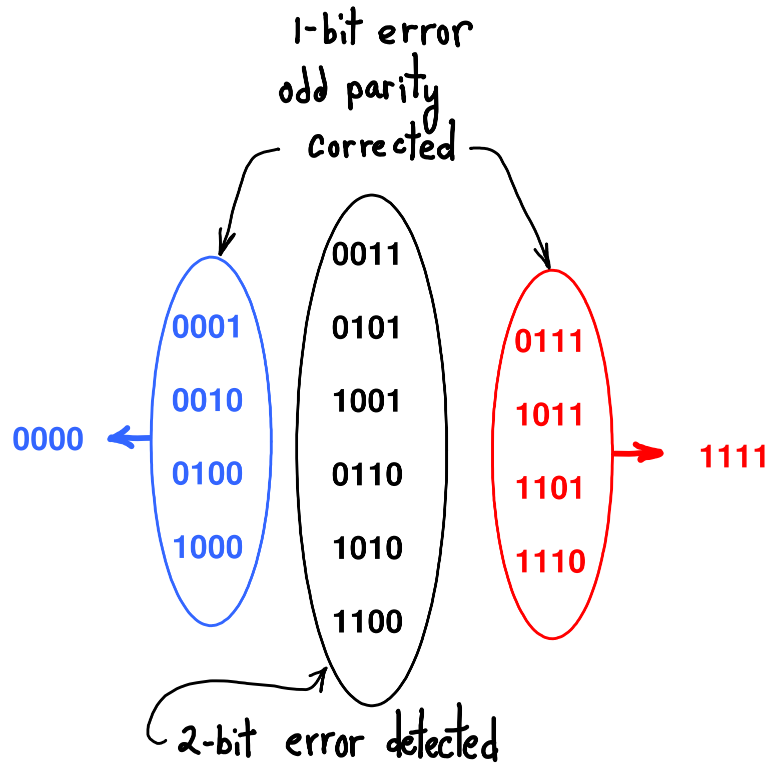
Code Words:
"0" ===> 0000
"1" ===> 1111

no error, or #errors > 2

How many bits are needed?
Depends on noise:
Shannon Noisy Coding Theorem.

Can you think of a scheme like the parity-bit scheme that uses as few bits as possible? (See Reed-Solomon codes, for instance.)

More bits, higher error probability?



Min. Hamming Distance between code words
→ How many bit errors we can handle.

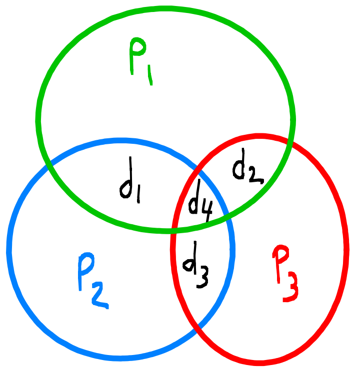
What's the probability of more than 2 bit errors?

Hamming (7, 4) Code **(Single-Error Detection, Single-Error Correction)**

7 bits per code word:

4 data bits
3 parity bit

code word $= d_1\, d_2\, d_3\, d_4\ p_1\ p_2\ p_3$



$p_1 = \text{parity}(d_1\ d_2\ d_4)$

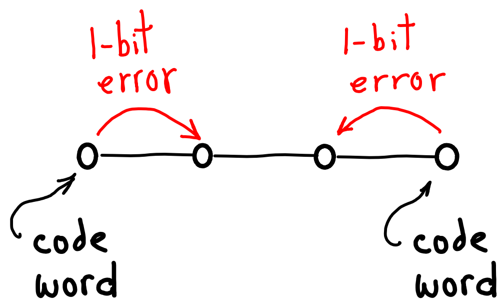
$p_2 = \text{parity}(d_1\ d_3\ d_4)$

$p_3 = \text{parity}(d_2\ d_3\ d_4)$

Guaranteed min. distance between code words is 3.

1-bit error: can detect and correct

2-bit error: cannot detect



What can we do about 2-bit errors? **Add another parity bit.**

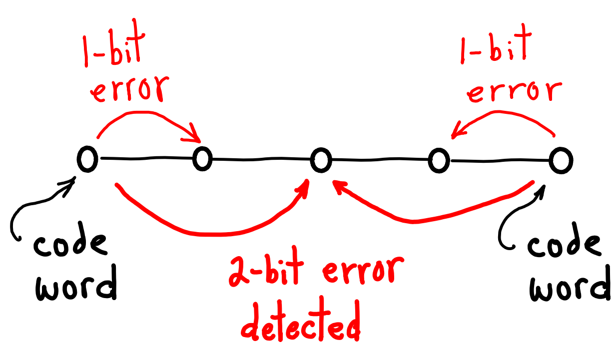
$p_4 = \text{parity}(d_1\, d_2\, d_3\ d_4\ p_1\ p_2\ p_3)$

Code word $= d_1\, d_2\ d_3\ d_4\ p_1\ p_2\ p_3\ p_4$

$\longrightarrow$ 4 steps min to next code word

1-bit error: detect + correct

2-bit error: detect



Hamming 7,4 code:
Find distances to all other code words from 0000000.
GREEN-PARITY: Bits[3, 2, 0]
BLUE-PARITY: Bits[3, 1, 0]
RED-PARITY: Bits[2, 1, 0]

0 0 0 0 0 0 0

0 0 0 1 1 1 1

(1-bit flip causes 3 other flips distance = 4)

distance = 3

0 0 1 0 0 1 1

0 1 0 0 1 0 1

1 0 0 0 1 1 0

1 1 1 0 0 0 0

0 0 1 1 1 0 0

0 1 0 1 0 1 0

1 0 0 1 0 0 1

distance = 4

0 1 1 0 1 1 0

0 1 1 1 0 0 1

1 0 1 1 0 1 0

1 1 0 1 1 0 0

1 0 1 0 1 0 1

1 1 0 0 0 1 1

distance = 7

1 1 1 1 1 1 1

Other encodings

Who's on first?

ASCII (See back cover of PP)

HEX CODE	MEANING	Printable?
00	NUL	no
01	SOH	no
...	...	
20	space	yes
...	...	
30	"0"	yes
31	"1"	yes
...	...	
41	"A"	yes
42	"B"	yes
...	...	
61	"a"	yes
62	"b"	yes
...	...	
7A	"z"	yes
...	...	

(other stuff, non-standard)

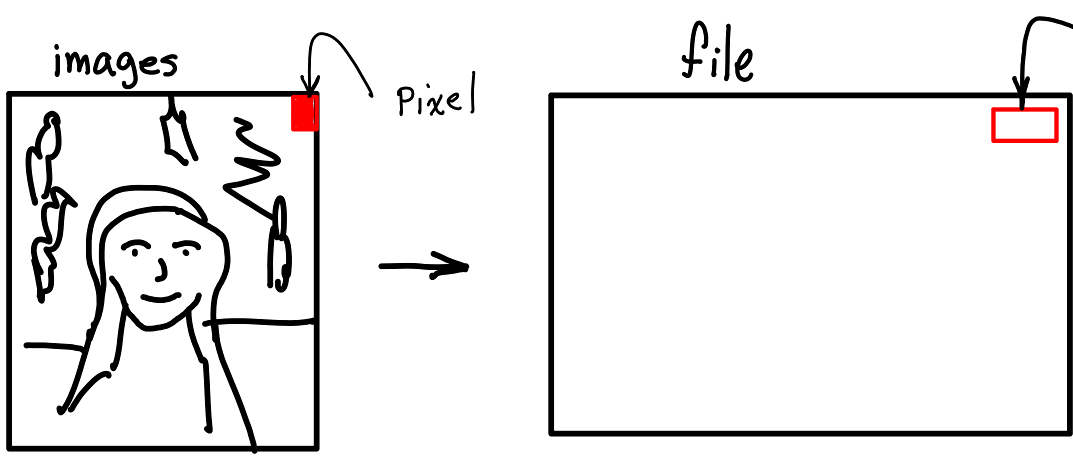
ASCII		Character
h32	===>	'2'
h2F	===>	'/'
h41	===>	'A'
h6D	===>	'm'

8-bit addr	Byte Addressable Memory bits		
00000000	00110010 (b_0)	→ h32	'2'
00000001	00101111	→ h2F	'/'
00000010	01000001	→ h41	'A'
00000011	01101101 (b_{63})	→ h6D	'm'

Print order ↓

What to Print	Starting Memory Address	What is displayed (left-to-right)
4-byte number (in hex notation)	0	6D412F32
two 2-byte numbers (in hex)	0	2F32 6D41
four 1-byte numbers (in hex)	0	32 2F 41 6D
one 4-byte string	0	2 / A m

(see "od" in unix)



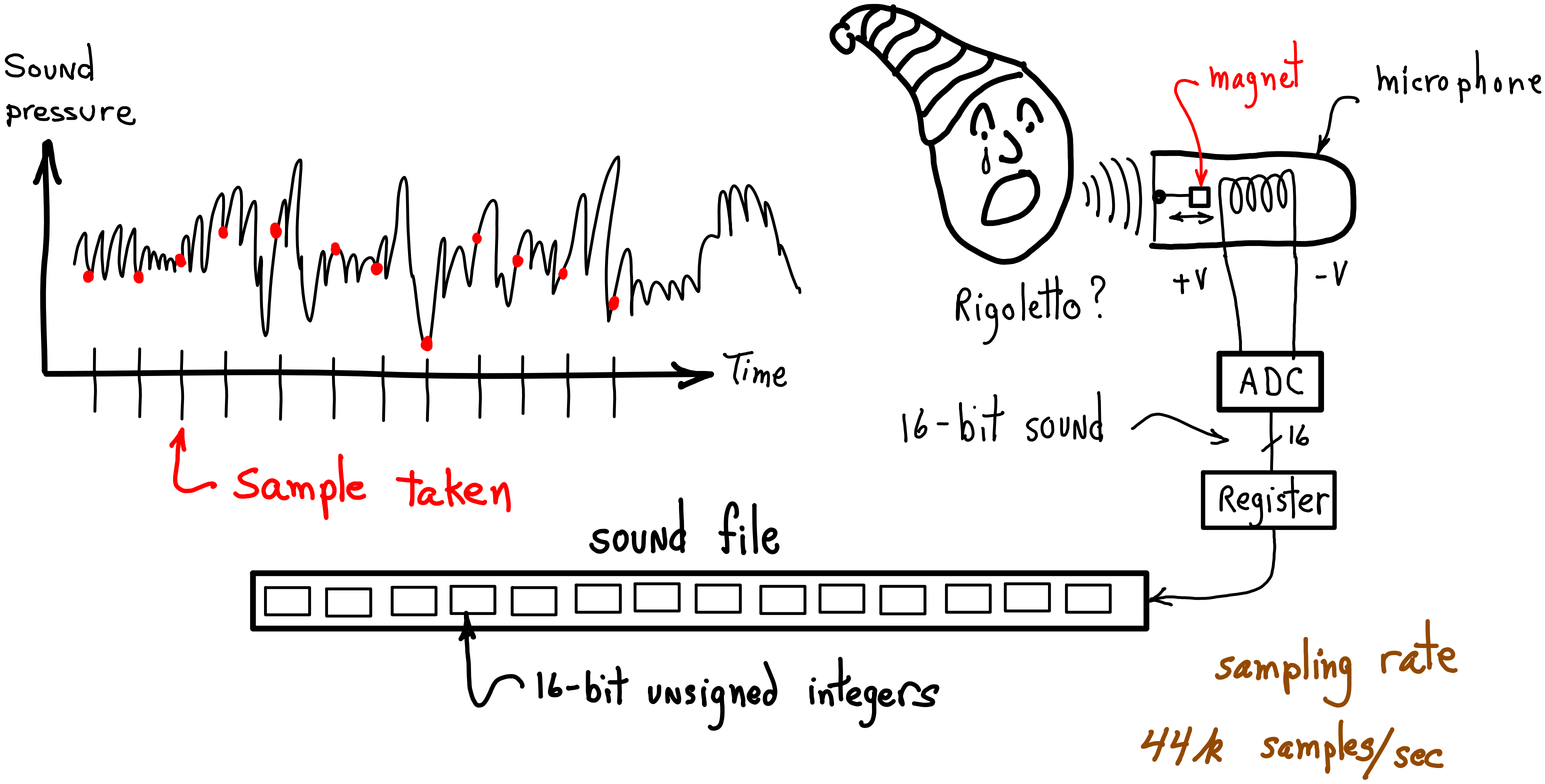
red green blue
(255, 0, 0)

three 8-bit numbers

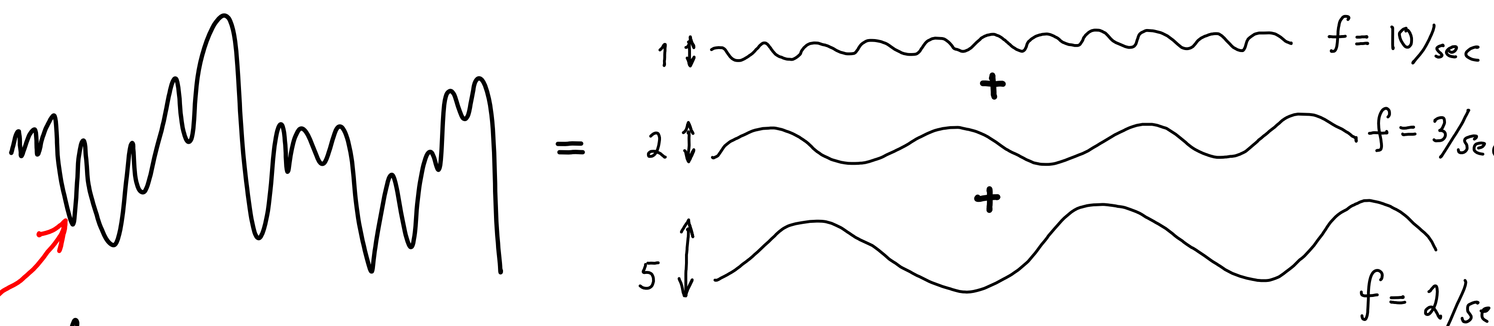
24-bits per pixel

1024 X 1024 pixels

===> 1 M x 3B
== 3 MB file



Lossy Compression



$$f(x) = 1 \cdot \cos(10t) + 2 \cdot \sin(3t) + 5 \cdot \sin(2t)$$

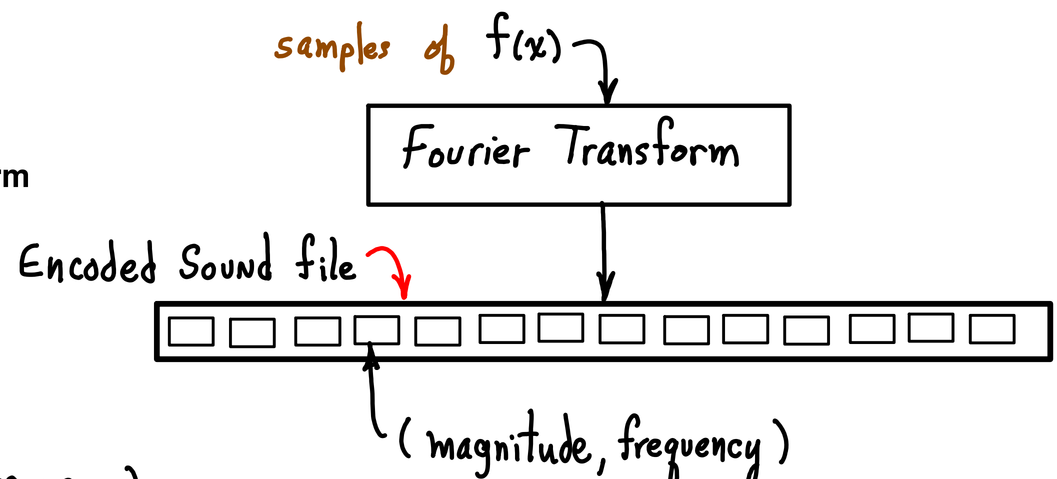
Compression

1. Filtering:

Eliminate cos(10 t) term

1. Coding

[(2, 3) ; (5, 2)]



JPEG
MPEG } CODEC

CODEC coder/decoder

Convert from/to sound/picture samples