Problem

You're an elevator manufacturer.

When button is pressed, the elevator responds with its status:

1) Not here, coming — 0.85
2) Already here — 0.10
3) Broken — 0.03
4) Held at another floor — 0.02.

Option 1: 00 - 1) On average: 2
01 - 2)
10 - 3)
11 - 4)

↖ two bits per option

0 → option 1

10 → option 2

110 → option 3

111 → option 4

Average:

$1 \cdot 0.85 + 2 \cdot 0.10 + 3 \cdot 0.05 = 1.2$ bits on average!

A	123
B	82
C	76
D	40
E	11

Assign a string of bits to each so that none is a prefix of another, and average number of bits is as small as possible.

A	00
B	01
C	10
D	110
E	111

optimal! 2.15 bits on average

A	15	00
B	12	01
C	6	100
D	5	101
E	4	110
F	2	1110
G	1	1111

right again!

A	0.28	00
B	0.27	01
C	0.23	10
D	0.09	110
E	0.04	11100
F	0.04	11101
G	0.03	11110
H	0.02	11111

→ 2.48 on average.

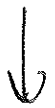
A	B	C	D	E	F	G	H
0.28	0.27	0.23	0.09	0.04	0.04	0.03	0.02

A	B	C	D	E	F	GH
0.28	0.27	0.23	0.09	0.04	0.04	0.05

A	B	C	D	EF	GH
0.28	0.27	0.23	0.09	0.08	0.05

A	B	C	D	EFGH
0.28	0.27	0.23	0.09	0.13

A	B	C	DEFGH
0.28	0.27	0.23	0.22



A B CDEFGH

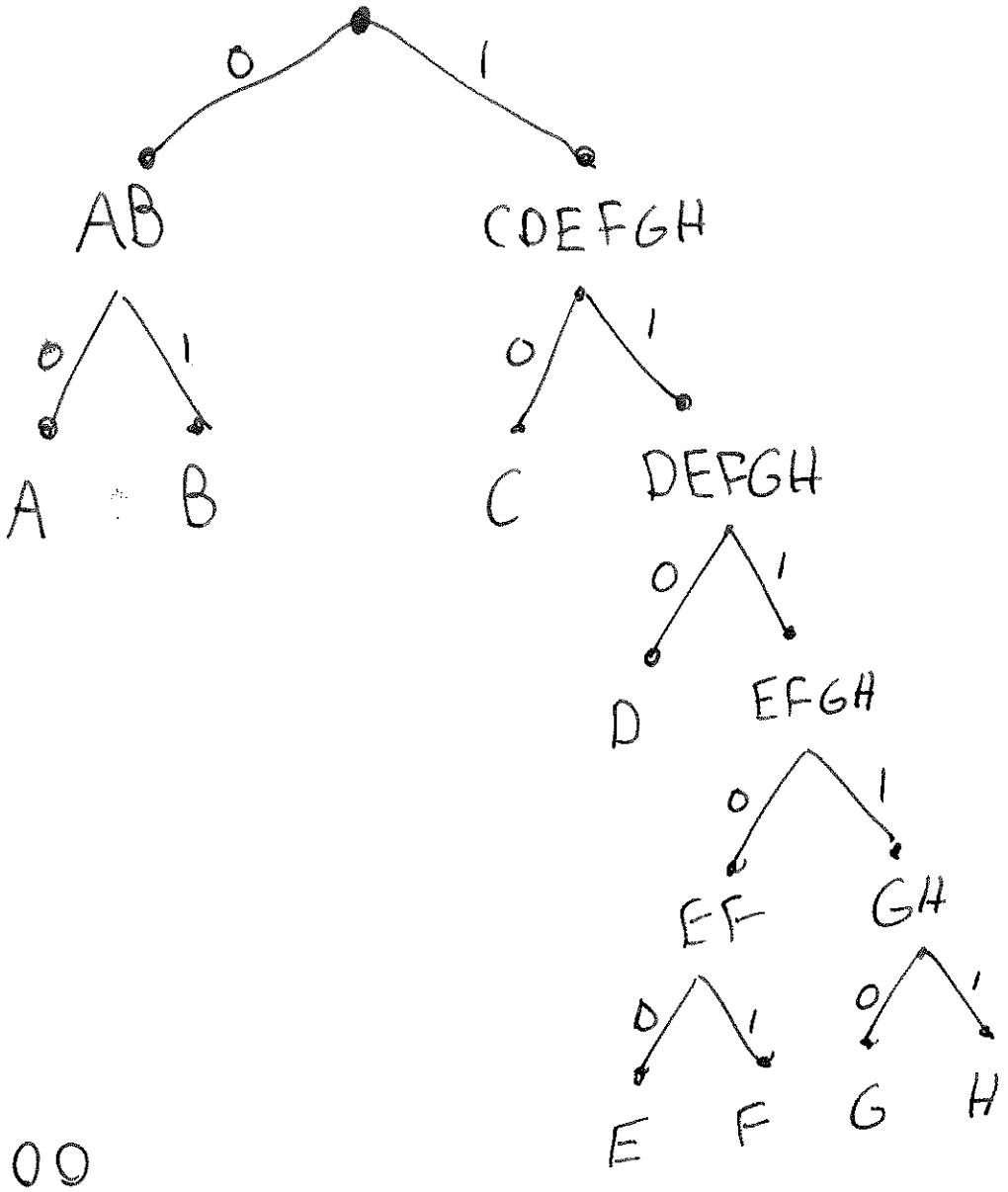
0.28 0.27 0.45

AB CDEFGH

0.55 0.45

ABCDEFGH

1.0



A	00
B	01
C	10
D	110
E	11100
F	11101
G	1111 0
H	1111 1

"Huffman encoding"

Another challenge:

You want to send a binary message over a "noisy" channel: every bit has probability $p \leq 1$ of getting reversed.

How to send message so the recipient can most likely determine original message even if some bits are flipped? (You can send redundant bits)

"Error-correcting code"

Guesses:

- Take your message, and send every bit three times.

01101 → 000111111000111

or quadruple! Quintuple!

high accuracy, but many wasted bits.

- Add a single "check" bit at the end.
(e.g. use binary sum of the bits)

0110100 → 0110100 ≠ checksum

If a single bit is flipped, what would recipient do?

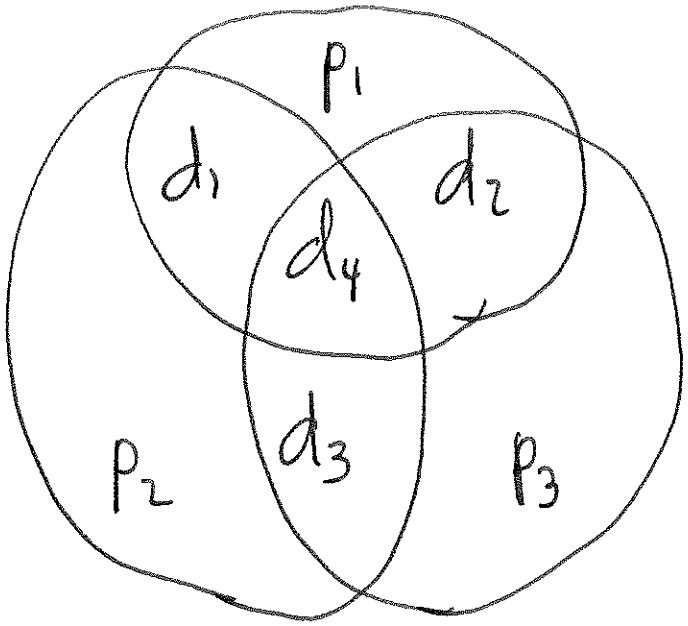
They know it's invalid, and ask for it to be sent again.

Another error-correcting code: Hamming code (7,4)

For every four bit chunck of message,

d_1 d_2 d_3 d_4

Compute three checksum bits p_1 p_2 p_3



Transmit:

p_1 p_2 d_1 p_3 d_2 d_3 d_4

Test: | 0 1 1 0 | 1 1 1 0 |
$d_1\ d_2\ d_3\ d_4$ | $d_1\ d_2\ d_3\ d_4$ |

$p_1 = 1$ $p_1 = 0$

$p_2 = 1$ $p_2 = 0$

$p_3 = 0$ $p_3 = 0$

↓

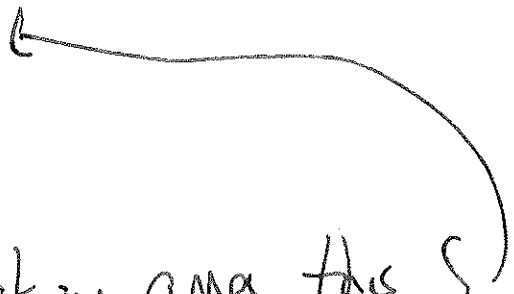
1 1 0 0 1 1 0 | 0 0 1 0 1 1 0

This multiplies the length by 7/4.

But it can still survive any single bit being flipped!

For any 7-bit string, there's at most one 4-bit string whose 7-bit encryption differs from s in at most one place.

1101 $\underline{=}$ 110



no 4-bit string gives this S