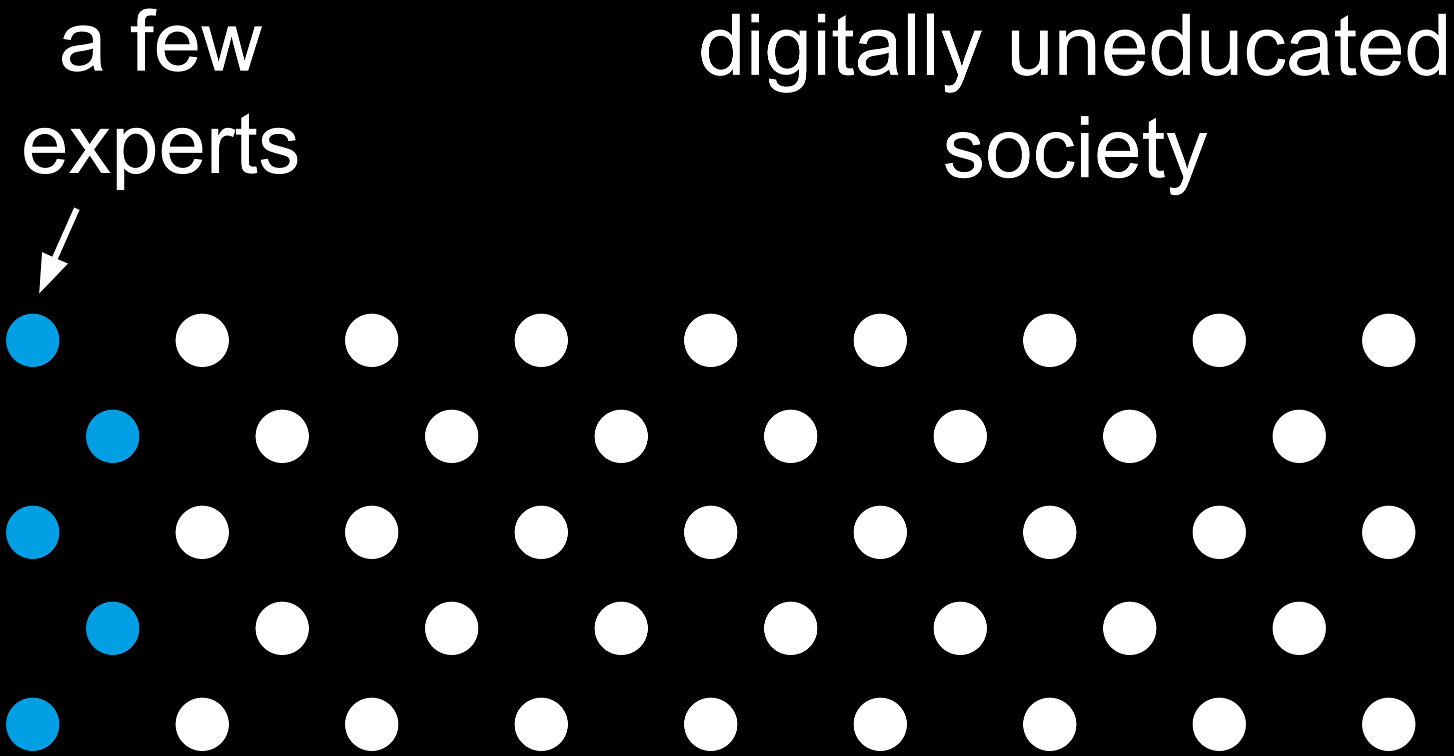
0. MOTIVATION
1. PROBLEM SOLVING
2. INFORMATION
3. COUNTING
4. BITS
5. CODES
6. ALGORITHMS
7. COMPUTERS
8. ARITHMETIC
9. MEMORY
10. ANALOG VS. DIGITAL

The slides are meant as visual support for the lecture.
They are not a documentation nor a script.

Please do not print the slides.

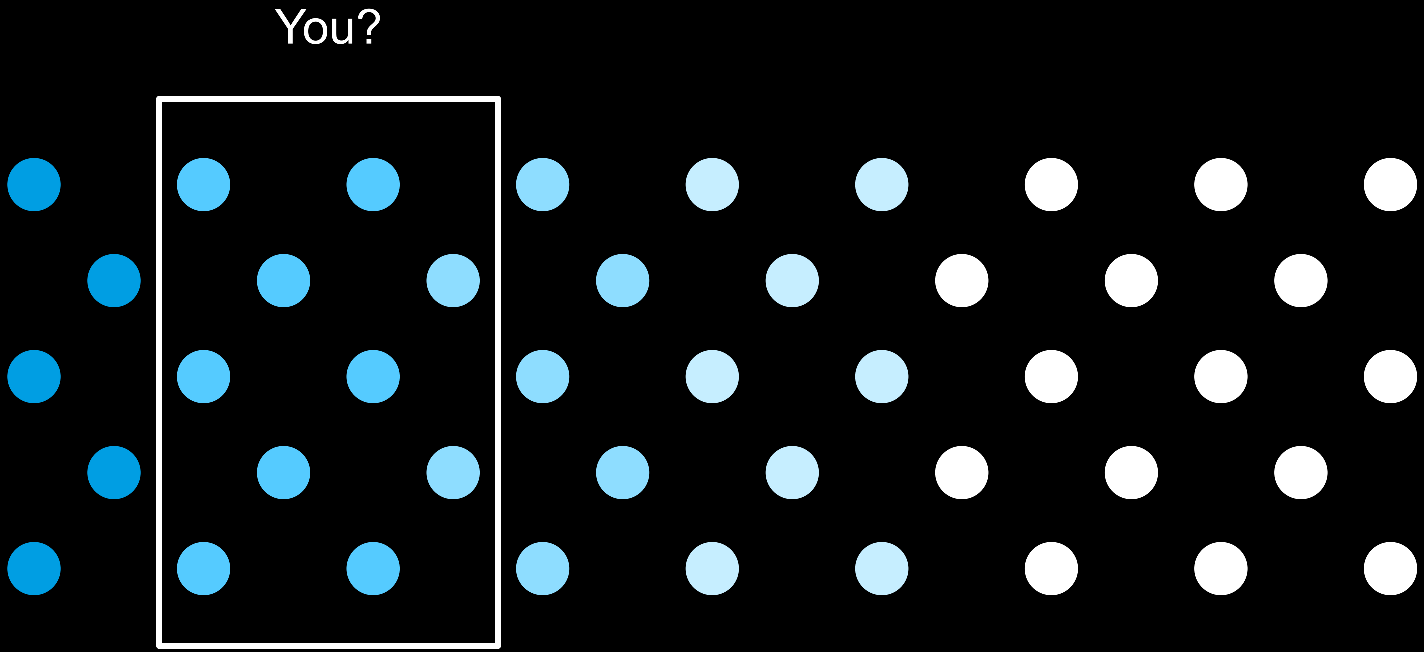
Comments and feedback at n.meseth@hs-osnabrueck.de

MOTIVATION



digitally illiterate society with a few experts

collective understanding



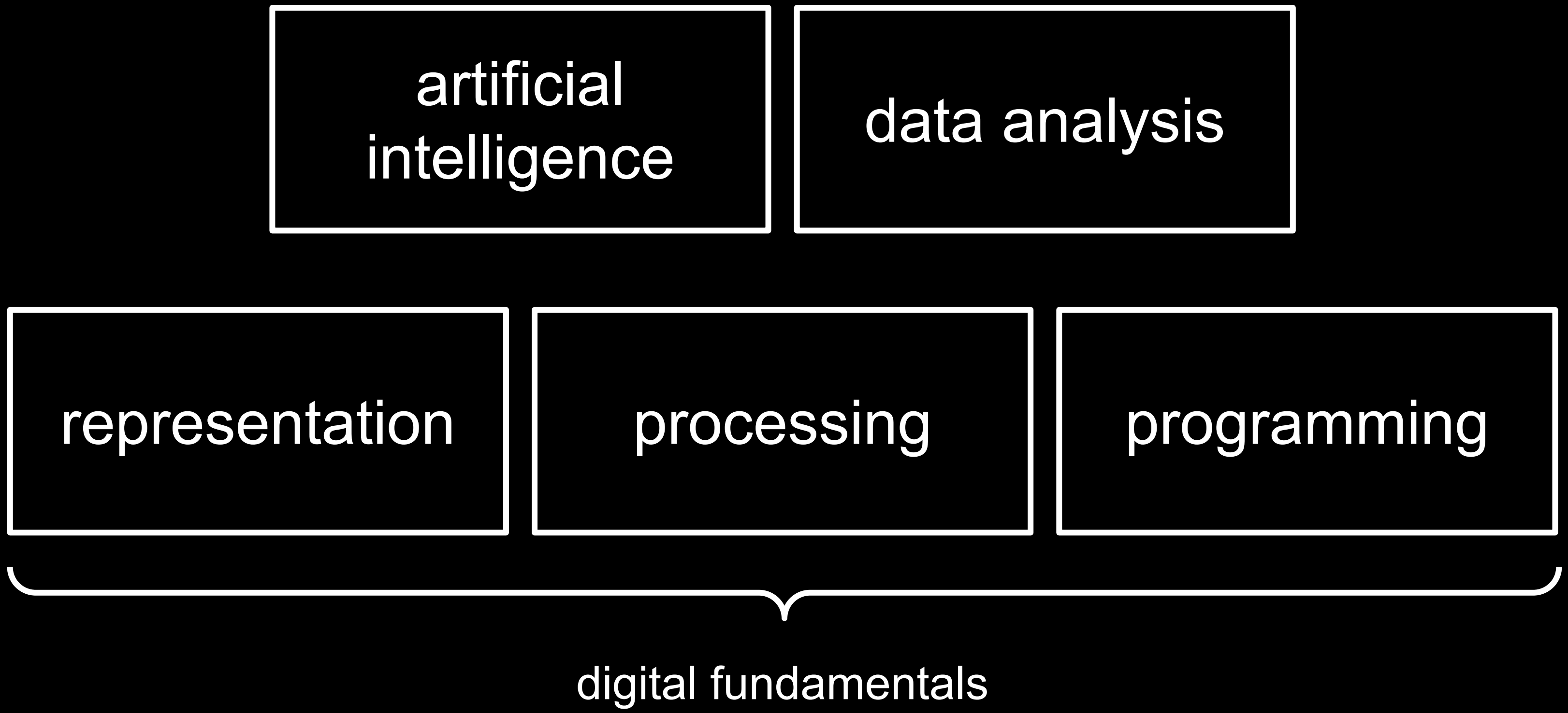
society with a distributed and high degree of digital education artificial Intelligence

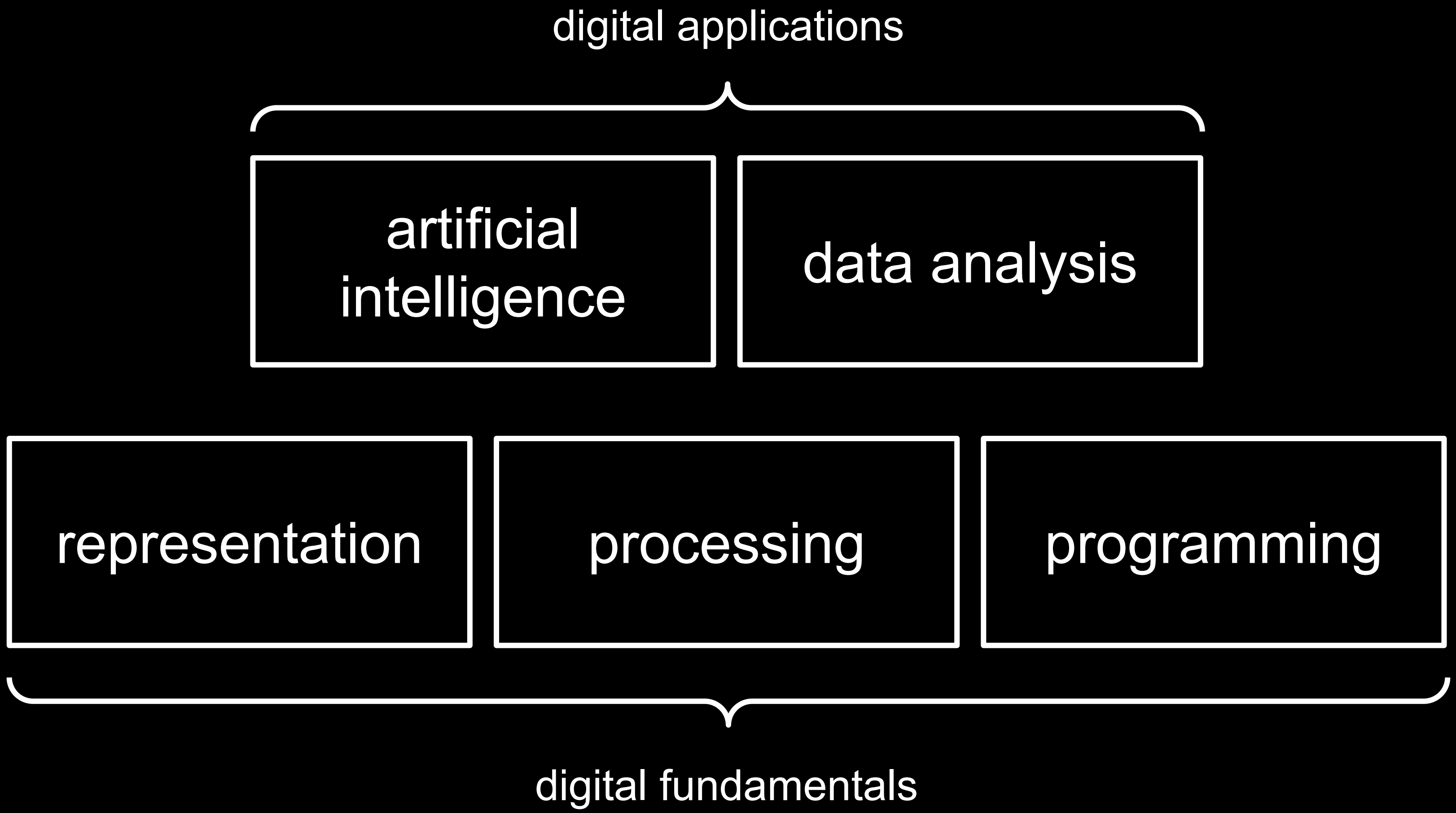
data analysis

representation

processing

programming


artificial intelligence
data analysis
representation
processing
programming
digital fundamentals


digital applications
artificial intelligence
data analysis
representation
processing
programming
digital fundamentals

digital applications

artificial intelligence

data analysis

representation

processing

programming

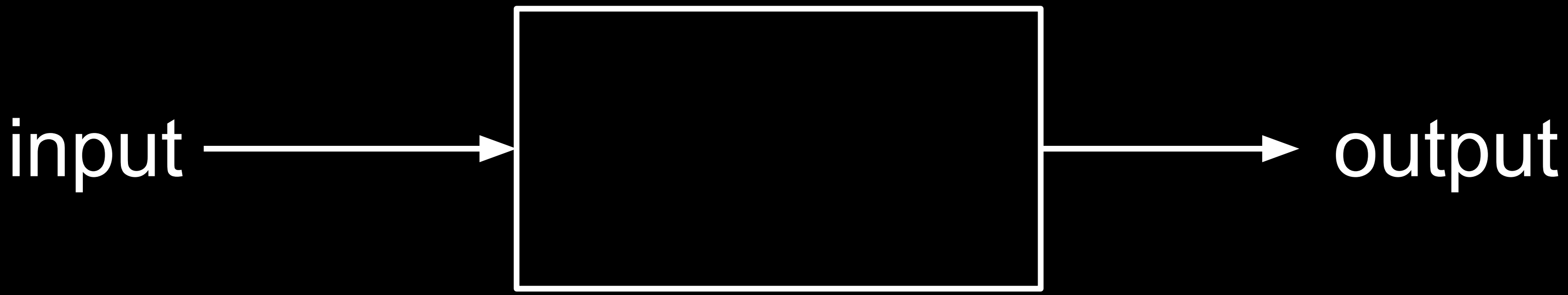
digital fundamentals

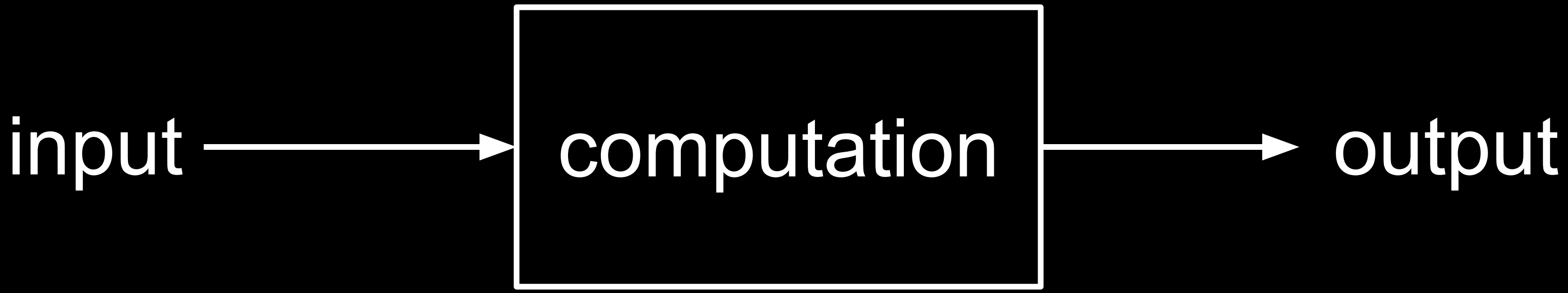
PROBLEM SOLVING

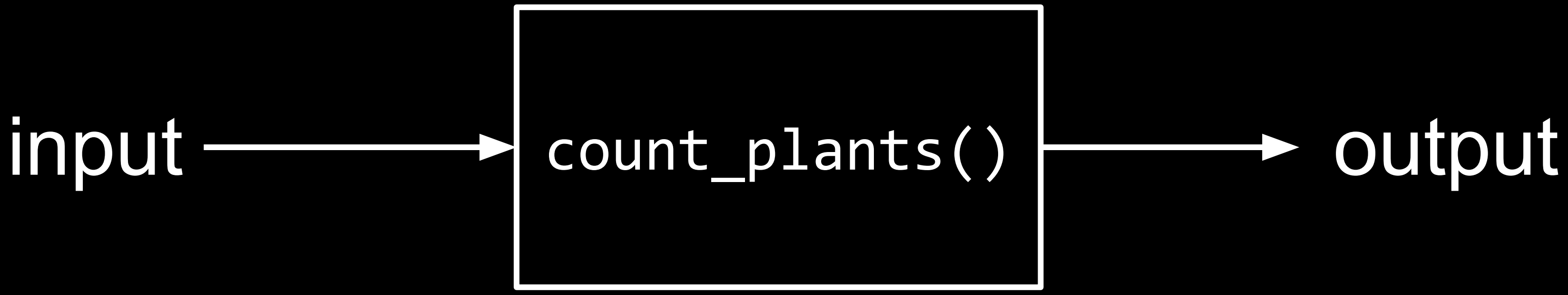
a model for solving problems

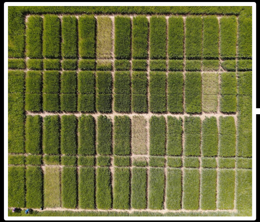
a model for solving problems

$$\{a, b\} \longrightarrow \boxed{a + b} \longrightarrow s$$


input
output

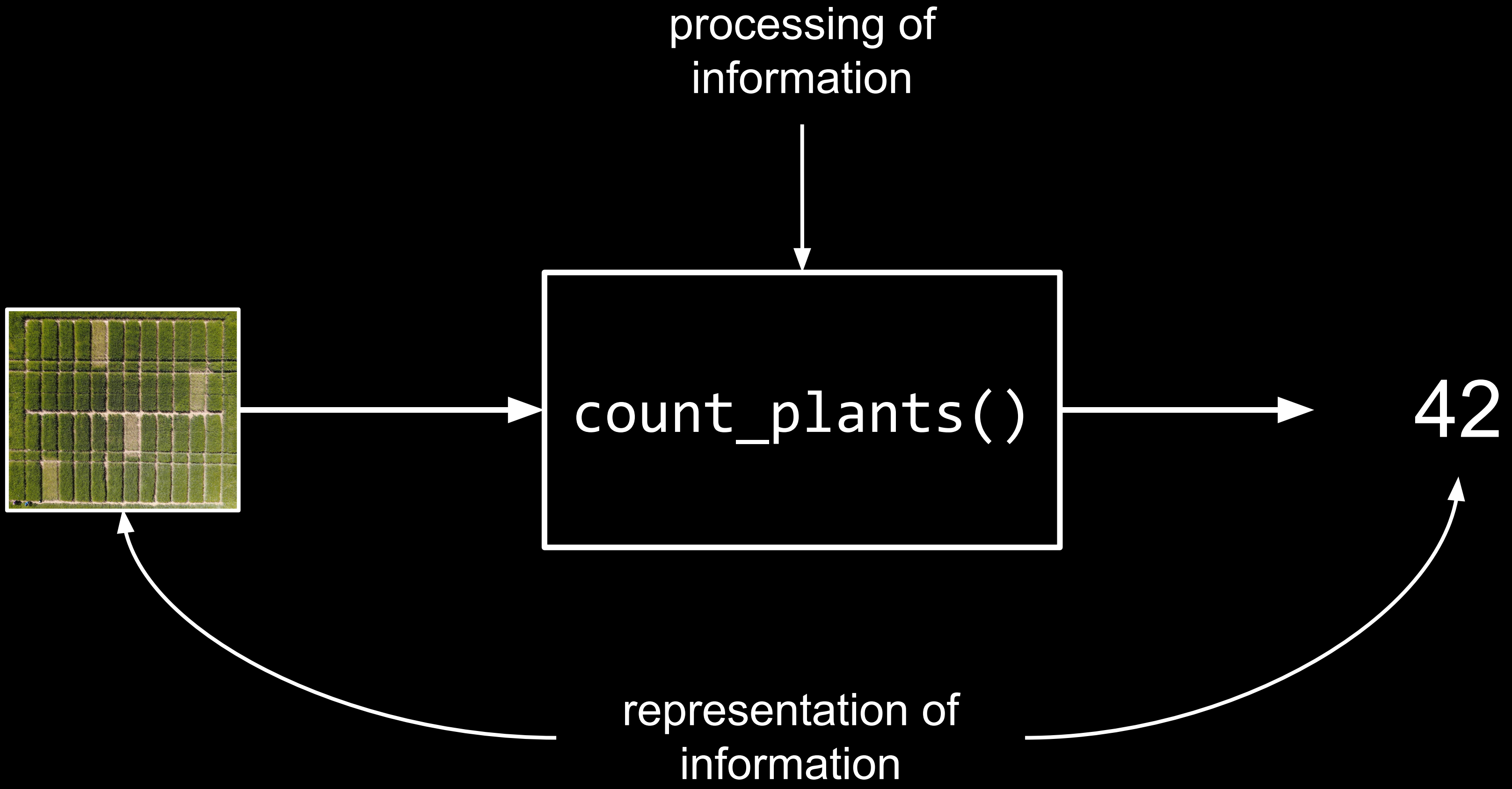

input
computation
output

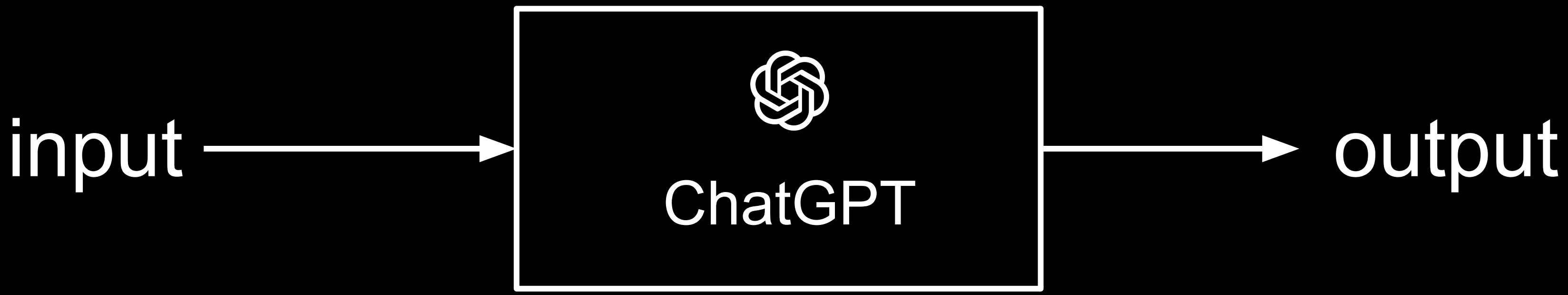

input
count_plants()
output

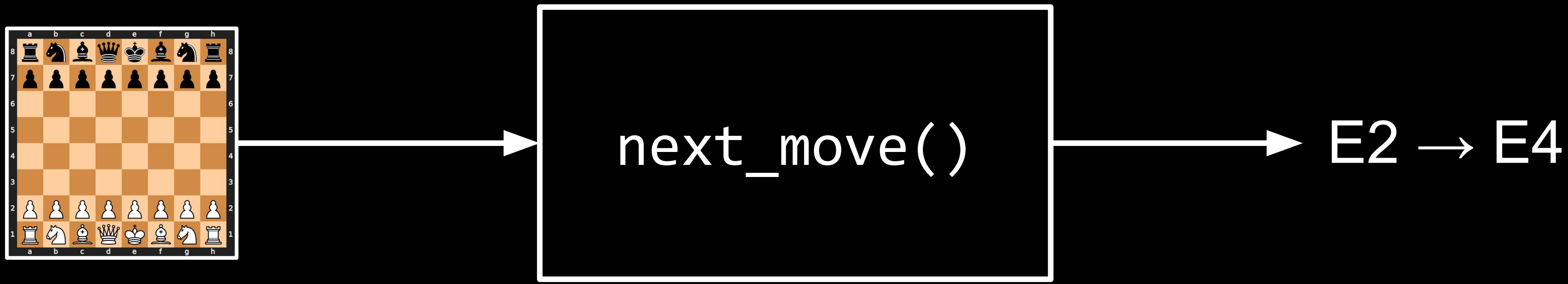


`count_plants()`

42


processing of information
count_plants()
42
representation of information


input
ChatGPT
output

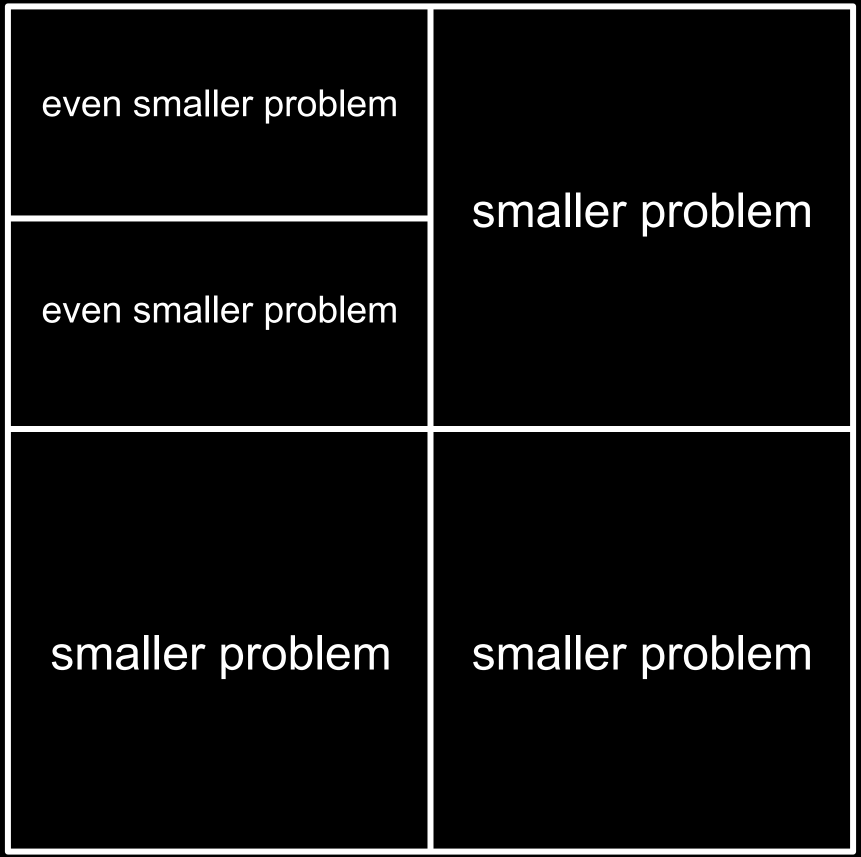

next_move()
E2 → E4

problem solving strategies

divide and conquer

large and complex problem

smaller problem	smaller problem
smaller problem	smaller problem


even smaller problem
even smaller problem
smaller problem
smaller problem
smaller problem


sorted list +
element
search()
yes / no

is 67 a prime number?

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41,
43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

linear search

```
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41,
43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97
```

linear search

~~2~~, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41,
43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

linear search

~~2~~, ~~3~~, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41,
43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

linear search

~~2~~, ~~3~~, ~~5~~, ~~7~~, ~~11~~, ~~13~~, ~~17~~, ~~19~~, ~~23~~, ~~29~~, ~~31~~, ~~37~~, ~~41~~,
~~43~~, ~~47~~, ~~53~~, ~~59~~, ~~61~~, 67, 71, 73, 79, 83, 89, 97

19 steps… can’t we do better?

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41,
43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

⬆

binary search

67 != 41

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, ~~41~~,
43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

binary search

67 > 41

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, ~~41~~,
43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

binary search

67 > 41

~~2~~, ~~3~~, ~~5~~, ~~7~~, ~~11~~, ~~13~~, ~~17~~, ~~19~~, ~~23~~, ~~29~~, ~~31~~, ~~37~~, ~~41~~,
43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

binary search

~~2~~, ~~3~~, ~~5~~, ~~7~~, ~~11~~, ~~13~~, ~~17~~, ~~19~~, ~~23~~, ~~29~~, ~~31~~, ~~37~~, ~~41~~,
43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

⬆

67 != 71

binary search

~~2~~, ~~3~~, ~~5~~, ~~7~~, ~~11~~, ~~13~~, ~~17~~, ~~19~~, ~~23~~, ~~29~~, ~~31~~, ~~37~~, ~~41~~,
43, 47, 53, 59, 61, 67, ~~71~~, 73, 79, 83, 89, 97

⬆

67 != 71

binary search

~~2~~, ~~3~~, ~~5~~, ~~7~~, ~~11~~, ~~13~~, ~~17~~, ~~19~~, ~~23~~, ~~29~~, ~~31~~, ~~37~~, ~~41~~,
43, 47, 53, 59, 61, 67, ~~71~~, ~~73~~, ~~79~~, ~~83~~, ~~89~~, ~~97~~

⬆

$67 < 71$

binary search

~~2~~, ~~3~~, ~~5~~, ~~7~~, ~~11~~, ~~13~~, ~~17~~, ~~19~~, ~~23~~, ~~29~~, ~~31~~, ~~37~~, ~~41~~,
43, 47, 53, ~~59~~, 61, 67, ~~71~~, ~~73~~, ~~79~~, ~~83~~, ~~89~~, ~~97~~

⬆

67 != 59

binary search

~~2~~, ~~3~~, ~~5~~, ~~7~~, ~~11~~, ~~13~~, ~~17~~, ~~19~~, ~~23~~, ~~29~~, ~~31~~, ~~37~~, ~~41~~,
~~43~~, ~~47~~, ~~53~~, ~~59~~, 61, 67, ~~71~~, ~~73~~, ~~79~~, ~~83~~, ~~89~~, ~~97~~

⬆

67 > 59

binary search

~~2~~, ~~3~~, ~~5~~, ~~7~~, ~~11~~, ~~13~~, ~~17~~, ~~19~~, ~~23~~, ~~29~~, ~~31~~, ~~37~~, ~~41~~,
~~43~~, ~~47~~, ~~53~~, ~~59~~, 61, 67, ~~71~~, ~~73~~, ~~79~~, ~~83~~, ~~89~~, ~~97~~

⬆

67 = 67

binary search

~~2~~, ~~3~~, ~~5~~, ~~7~~, ~~11~~, ~~13~~, ~~17~~, ~~19~~, ~~23~~, ~~29~~, ~~31~~, ~~37~~, ~~41~~,
~~43~~, ~~47~~, ~~53~~, ~~59~~, ~~61,~~ 67, ~~71~~, ~~73~~, ~~79~~, ~~83~~, ~~89~~, ~~97~~

⬆

67 = 67

3 splits → much better

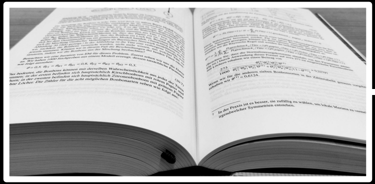
~~2~~, ~~3~~, ~~5~~, ~~7~~, ~~11~~, ~~13~~, ~~17~~, ~~19~~, ~~23~~, ~~29~~, ~~31~~, ~~37~~, ~~41~~,
~~43~~, ~~47~~, ~~53~~, ~~59~~, ~~61,~~ 67, ~~71~~, ~~73~~, ~~79~~, ~~83~~, ~~89~~, ~~97~~

⬆

67 = 67

🤔

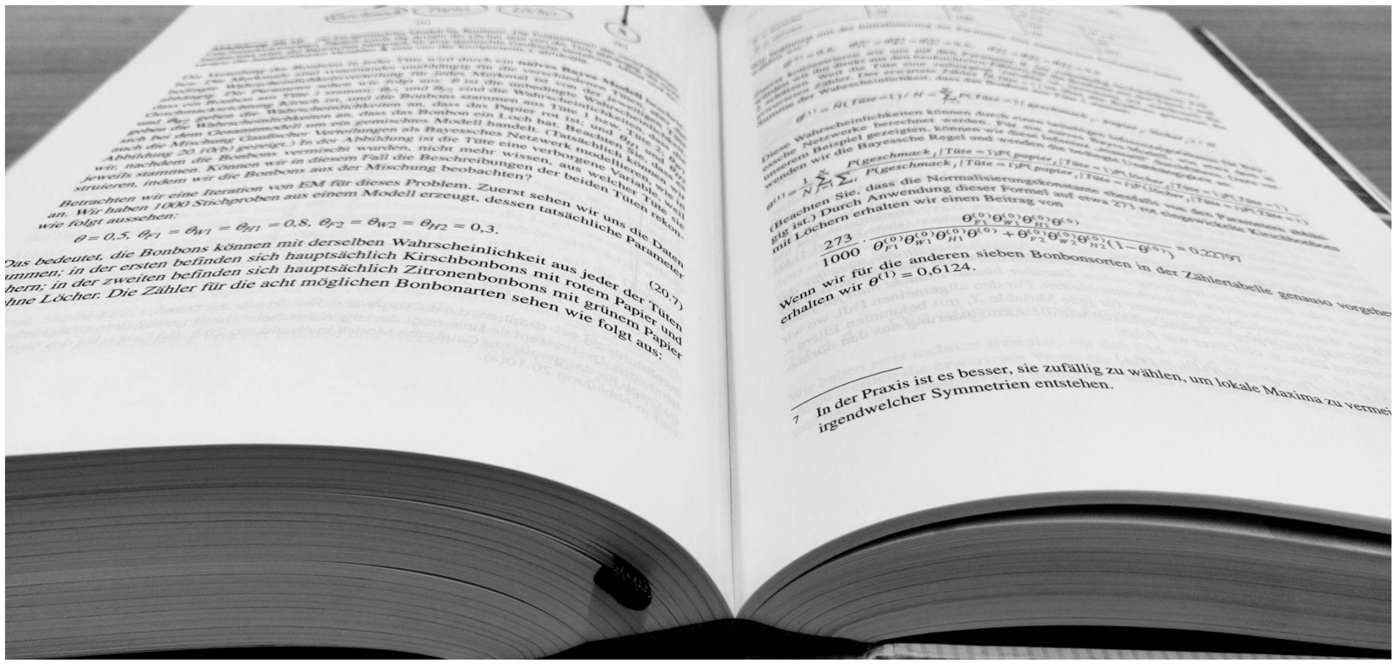
how efficient are linear and binary search in general?



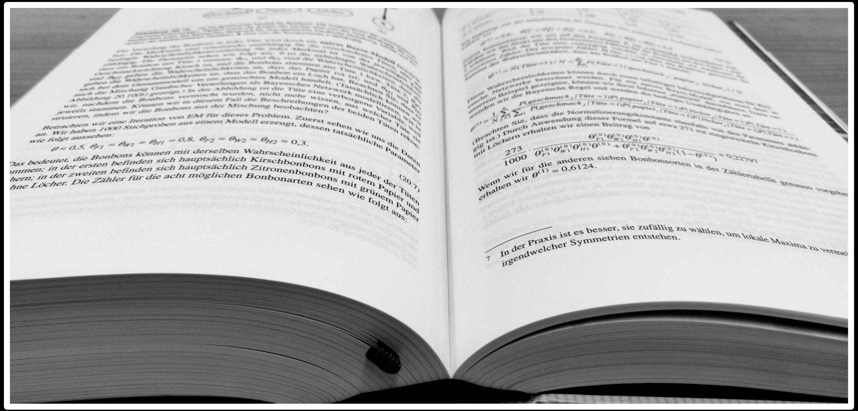
count_words()

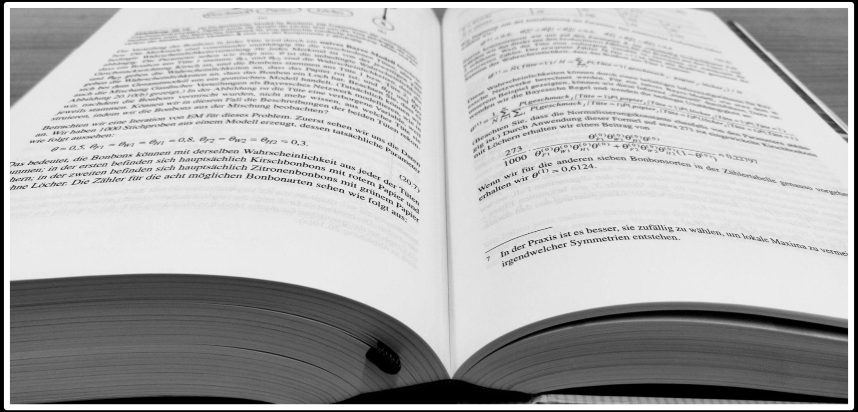
word count

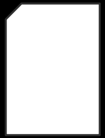
how many words are in the book?



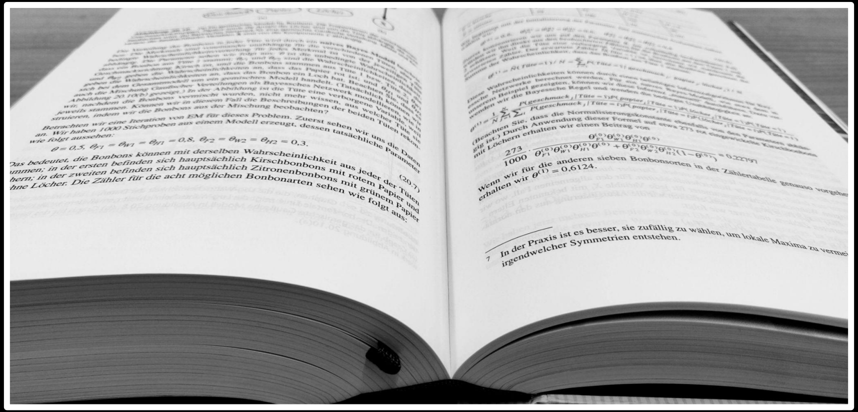
strategies, anyone?


Betrachten wir eine Iteration von EM für dieses Problem. Zuerst sehen wir uns die Daten
an. Wir haben 1000 Stichproben aus einem Modell erzeugt, dessen tatsächliche Parameter
wie folgt aussehen:
Wenn wir für die anderen sieben Bonbonsorten in der Zählertabelle genauso
erhalten wir $\theta^{(1)} = 0{,}6124$.
In der Praxis ist es besser, sie zufällig zu wählen, um lokale Maxima zu
irgendwelcher Symmetrien entstehen.

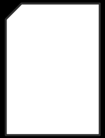




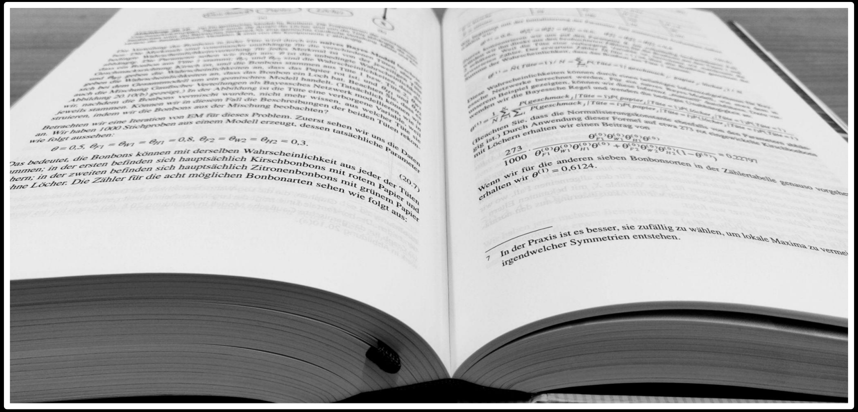
page 1



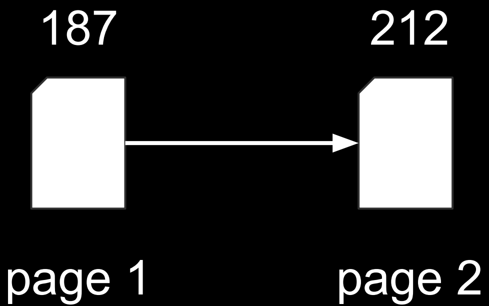
187



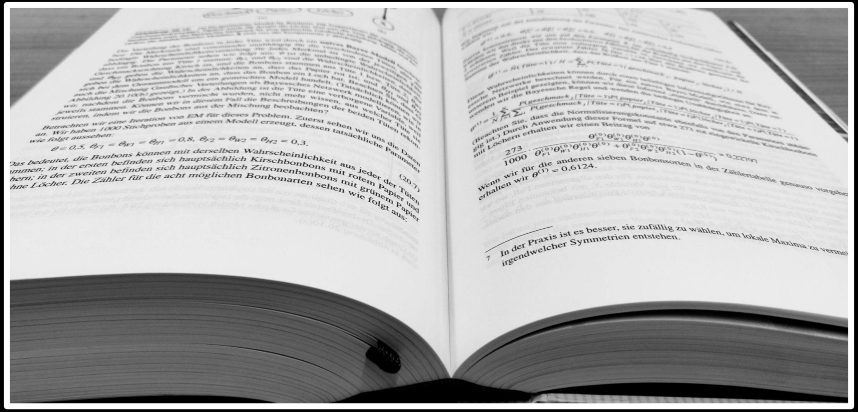
page 1

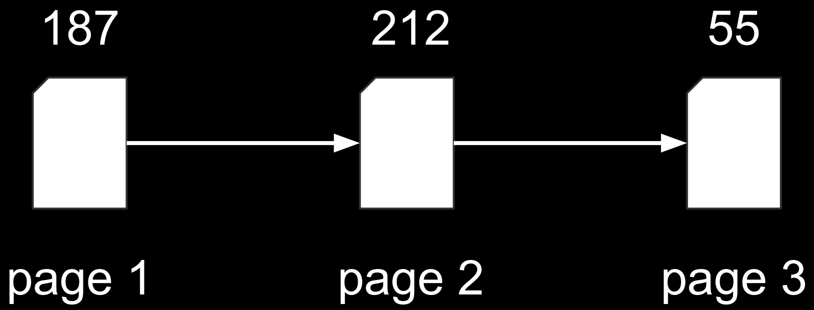


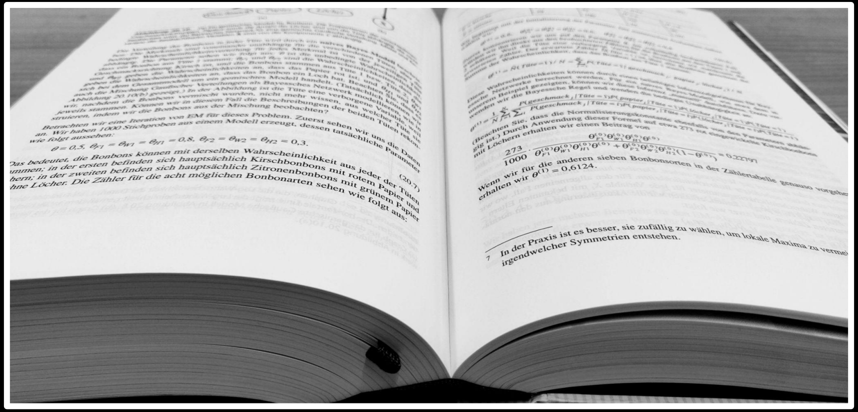
Betrachten wir eine Iteration von EM für dieses Problem.
In der Praxis ist es besser, sie zufällig zu wählen
erhalten wir $\theta^{(1)} = 0,6124$.

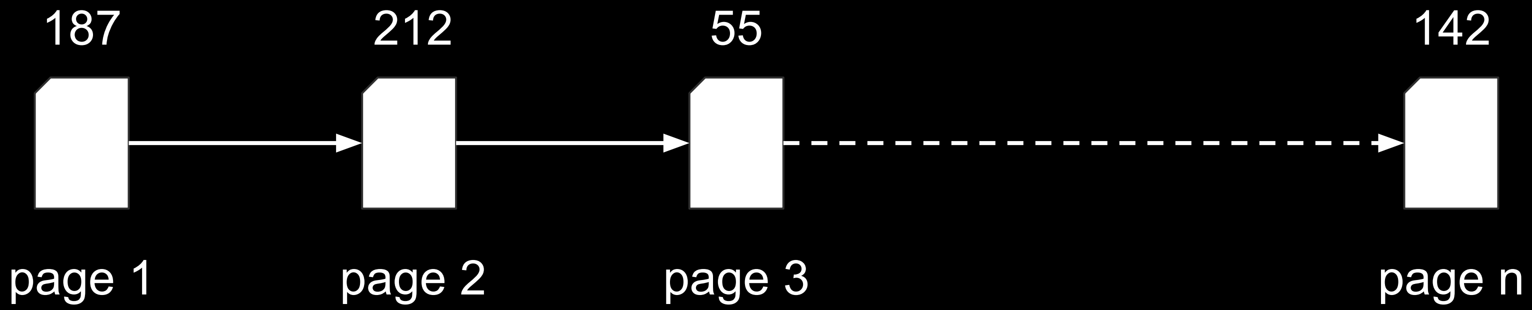


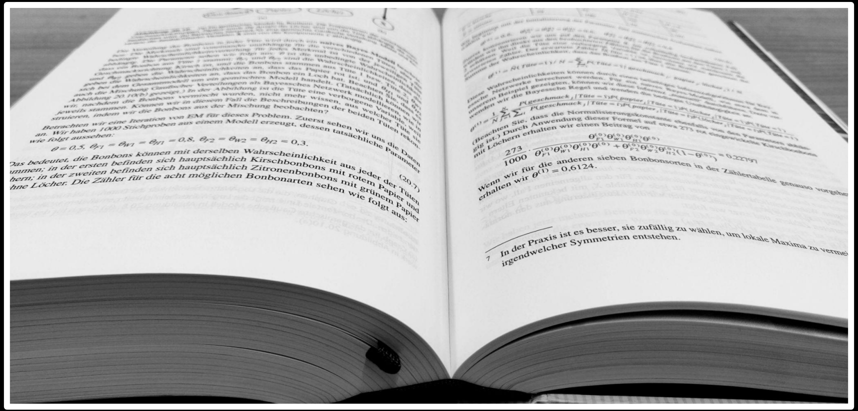
187
212
page 1
page 2




187
212
55
page 1
page 2
page 3




187
212
55
142
page 1
page 2
page 3
page n



n = 1327 pages

Ø 2:23 minutes per page

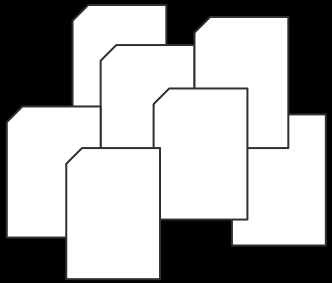
~ 52.34 hours

divide and conquer

+

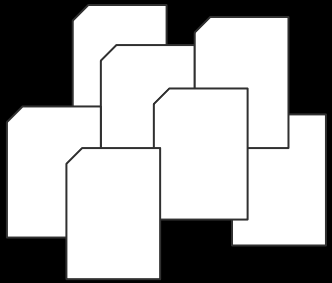
?

pages 1 - 700



student 1

pages 701 - 1327



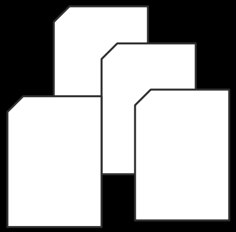
student 2

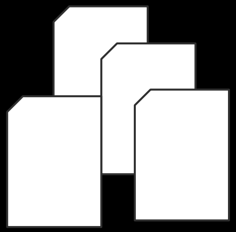
pages 1 - 350

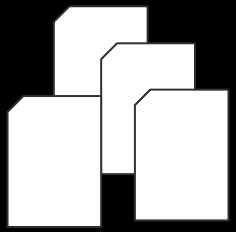
pages 351 - 700

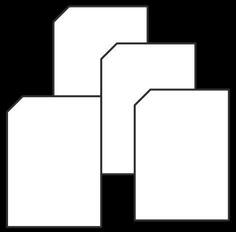
pages 701 - 1050

pages 1051- 1327









student 1

student 2

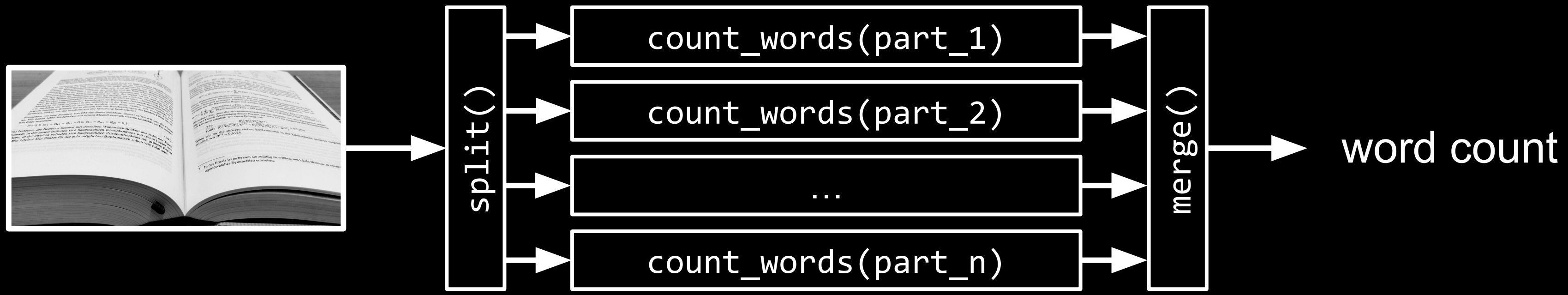
student 3

student 4

divide and conquer

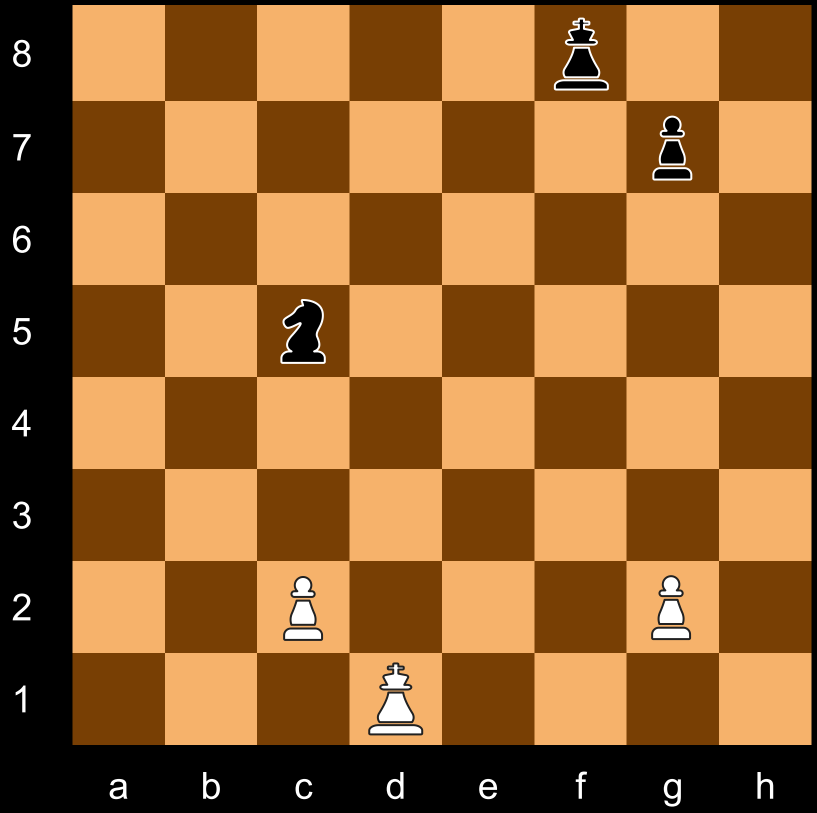
+

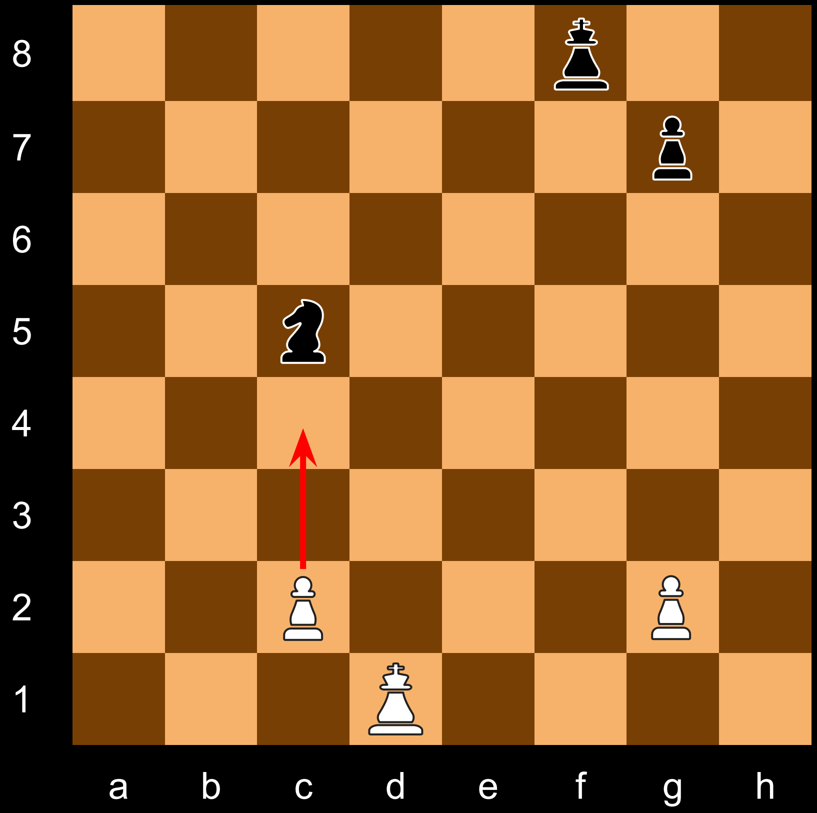
distribution and parallelization

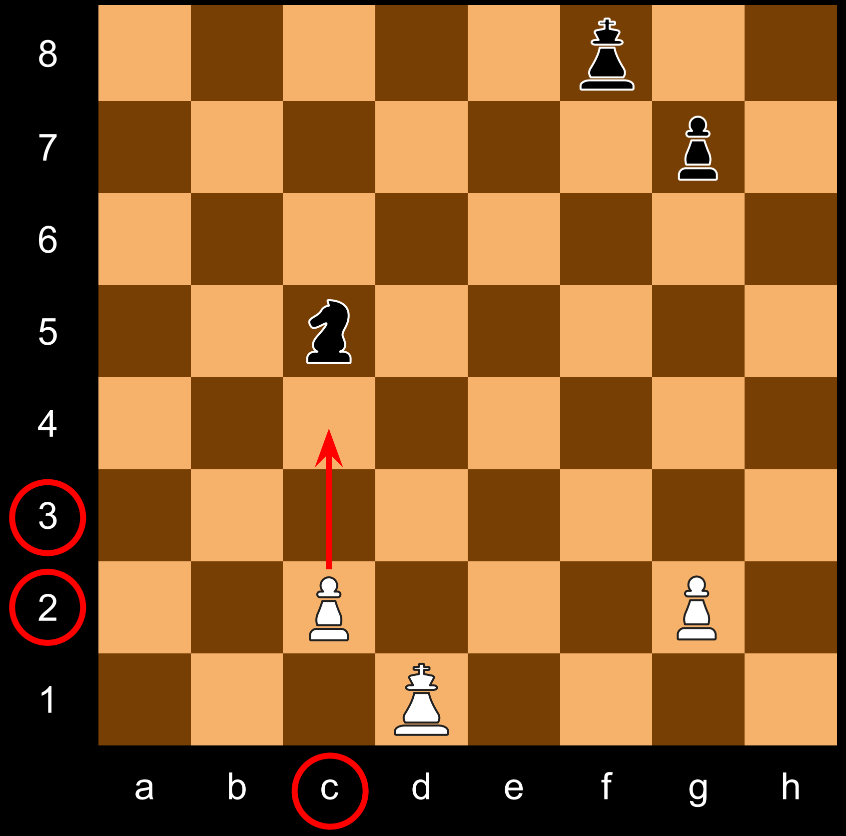

split()
count_words(part_1)
count_words(part_2)
...
count_words(part_n)
merge()
word count

INFORMATION

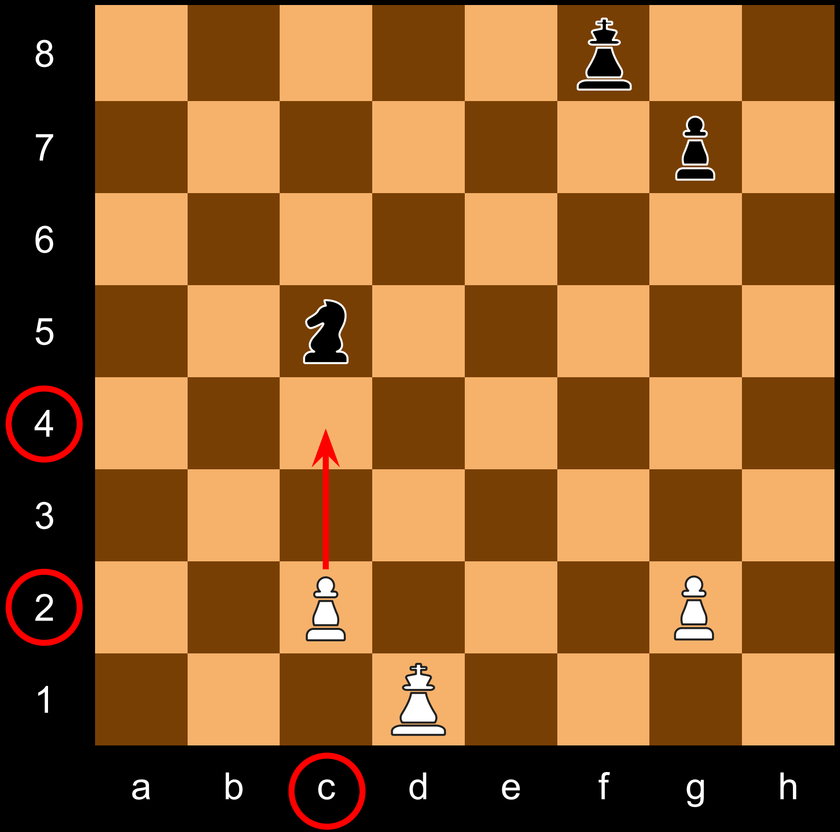

Image created with DALLE-3


8
7
6
5
4
3
2
1
a b c d e f g h

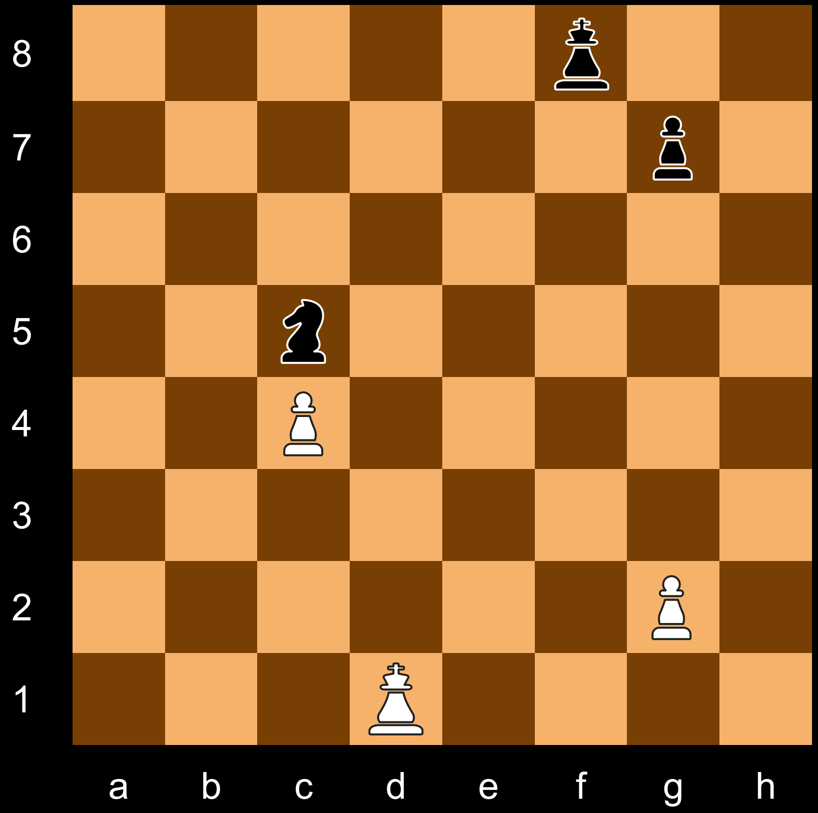

8
7
6
5
4
3
2
1
a b c d e f g h


8
7
6
5
4
3
2
1
a
b
c
d
e
f
g
h

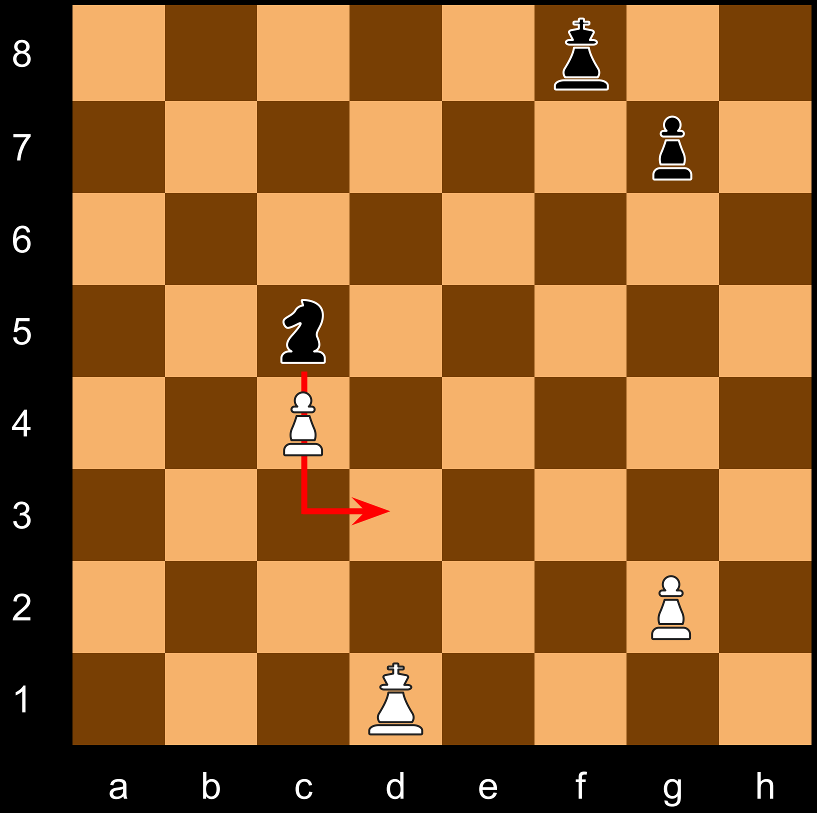
$c2 \to c4$



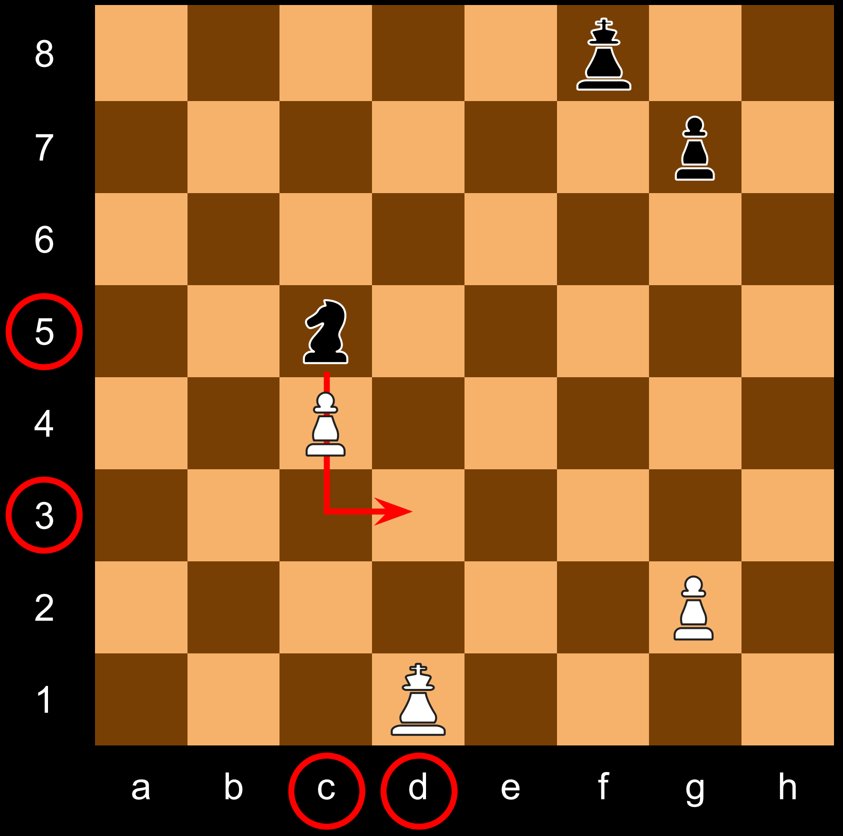
c2 → c4



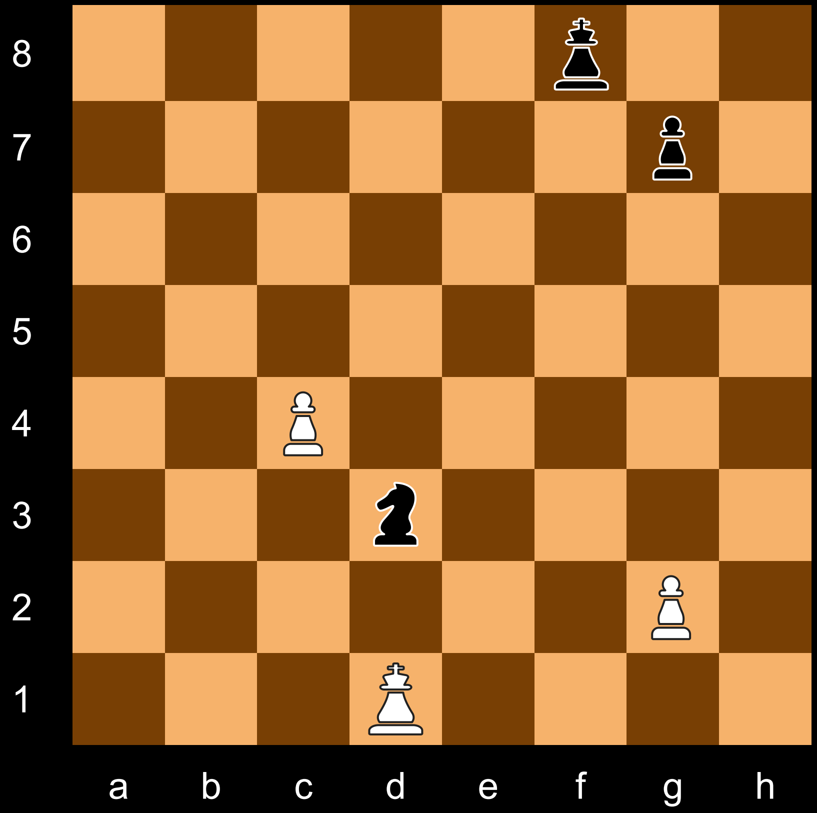
c2 → c4



c2 → c4
c5 → d3



c2 → c4
c5 → d3
…



{A}

_ _

{A}

A A

{A, B}

_ _

{A, B}

A A

{A, B}

A B

{A, B}

B A

{A, B}

B B

{A, B}

AA, AB, BA, BB

{A, B, C}

_ _

{A, B, C}

AA, AB, BA, BB,
AC, BC, CA, CB, CC

{A, B, C, D}

— —

{A, B, C, D}

AA, AB, BA, BB, AC, BC, CA, CB, CC, AD, DA, BD, DB, CD, DC, DD

{A, B, C, D, E}

_ _

{A, B, C, D, E}

AA, AB, BA, BB, AC, BC, CA, CB, CC,
AD, DA, BD, DB, CD, DC, DD, AE, EA,
BE, EB, CE, EC, DE, ED, EE

with length n = 2

# symbols	# messages
1	1
2	4
3	9
4	16
5	25

with length n = 2

# symbols		# messages
1		1
2		4
3	$\xrightarrow{f(x)}$	9
4		16
5		25

{A, B}

_ _ _

COUNTING

1 2 3

1	2	3
10^2	10^1	10^0

1 2 3

10^2 10^1 10^0

$$= 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$$

$$= 1 \times 100 + 2 \times 10 + 3 \times 1$$

$$= 123$$

4	1	2	3
?	10^2	10^1	10^0

4	1	2	3
?	10^2	10^1	10^0

$$= 4 \times 10^3 + 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$$

$$\begin{array}{cccc} 4 & 1 & 2 & 3 \\ \hline ? & 10^2 & 10^1 & 10^0 \end{array}$$

$$= 4 \times 10^3 + 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$$

$$= 4 \times 1000 + 1 \times 100 + 2 \times 10 + 3 \times 1$$

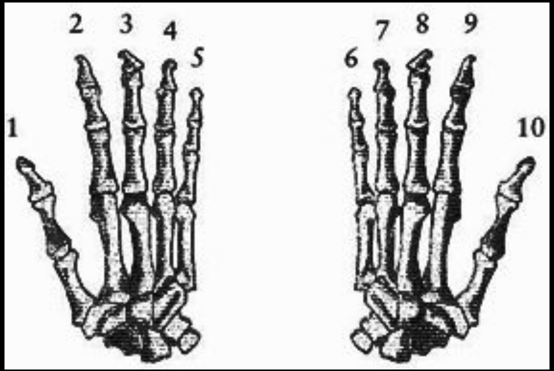
$$4 \quad 1 \quad 2 \quad 3$$

$$? \quad 10^2 \quad 10^1 \quad 10^0$$

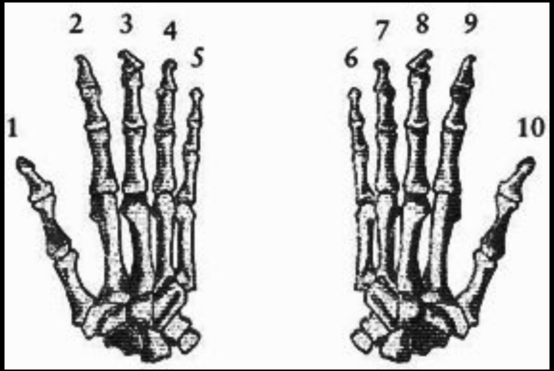
$$= 4 \times 10^3 + 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$$

$$= 4 \times 1000 + 1 \times 100 + 2 \times 10 + 3 \times 1$$

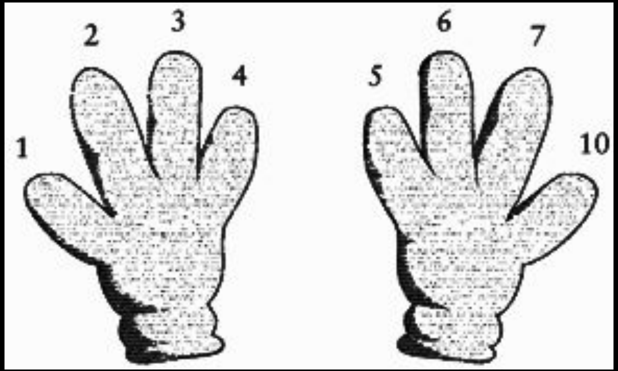
$$= 4123$$



Human Hand



Human Hand



Cartoon Character’s Hand

1 2 3 (octal)

1 2 3 (octal)

8^2 8^1 8^0

$$1 \quad 2 \quad 3 \quad \text{(octal)}$$

$$8^2 \quad 8^1 \quad 8^0$$

$$= 1 \times 8^2 + 2 \times 8^1 + 3 \times 8^0$$

$$1 \quad 2 \quad 3 \quad \text{(octal)}$$

$$8^2 \quad 8^1 \quad 8^0$$

$$= 1 \times 8^2 + 2 \times 8^1 + 3 \times 8^0$$

$$= 1 \times 64 + 2 \times 8 + 3 \times 1$$

$$1 \quad 2 \quad 3$$ (octal)

$8^2 \quad 8^1 \quad 8^0$

$$= 1 \times 8^2 + 2 \times 8^1 + 3 \times 8^0$$

$$= 1 \times 64 + 2 \times 8 + 3 \times 1$$

$= 83$ (decimal)

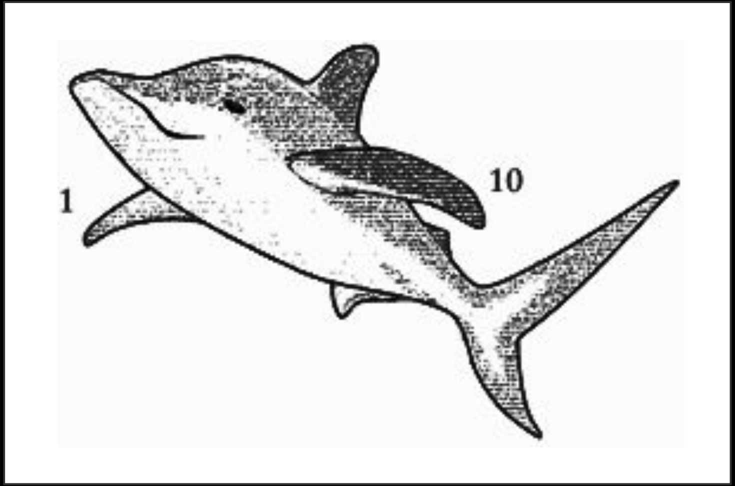
decimal		octal
8	$\longrightarrow$	?

decimal octal

? ← 7

decimal		octal
16	$\longrightarrow$	?

decimal		octal
?	←	100



What now?

0, 1, …

0, 1, 10, …

0, 1, 10, 11, …

0, 1, 10, 11, 100, …

0, 1, 10, 11, 100, 101, …

0, 1, 10, 11, 100, 101, 110

1 1 0 (binary)

1	1	0	(binary)
2^2	2^1	2^0	

$$1 \quad 1 \quad 0$$ (binary)

$$2^2 \quad 2^1 \quad 2^0$$

$$= 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

$$\begin{array}{ccc} 1 & 1 & 0 \\ \hline 2^2 & 2^1 & 2^0 \end{array} \quad \text{(binary)}$$

$$= 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

$$= 1 \times 4 + 1 \times 2 + 0 \times 1$$

1 1 0 (binary)

2^2 2^1 2^0

$= 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= 1 \times 4 + 1 \times 2 + 0 \times 1$

$=$ **6** (decimal)

2 3 4 5 6

0, 1, 10, 11, 100, 101, 110

Place Value Systems

$$N = d_n * R^{n-1} + \ldots + d_1 * R^1 + d_0 * R^0$$

$d \in \{ 0, 1, \ldots R-1 \}$

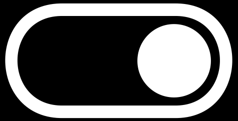
n = Number of digits

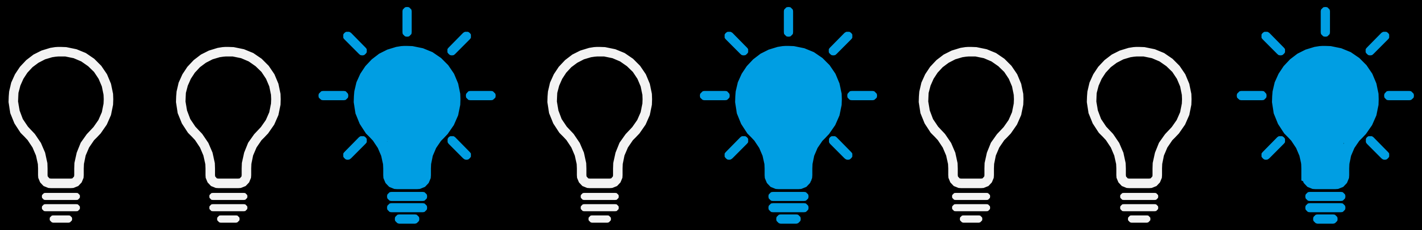
Place Value Systems

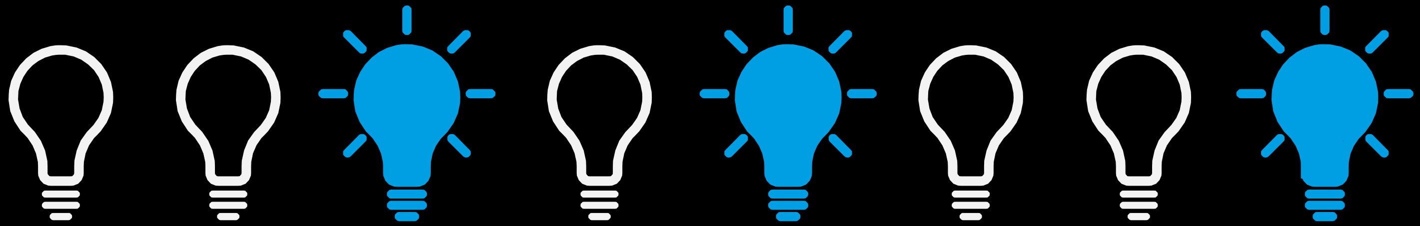
$$R \geq 2$$

BITS

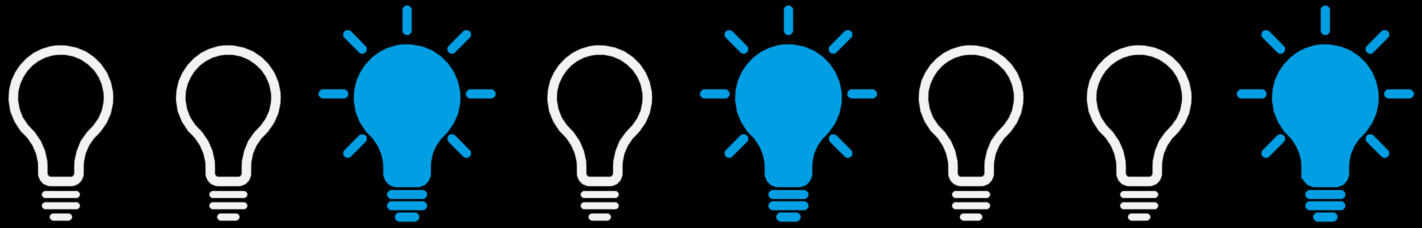
Why do computers think binary?





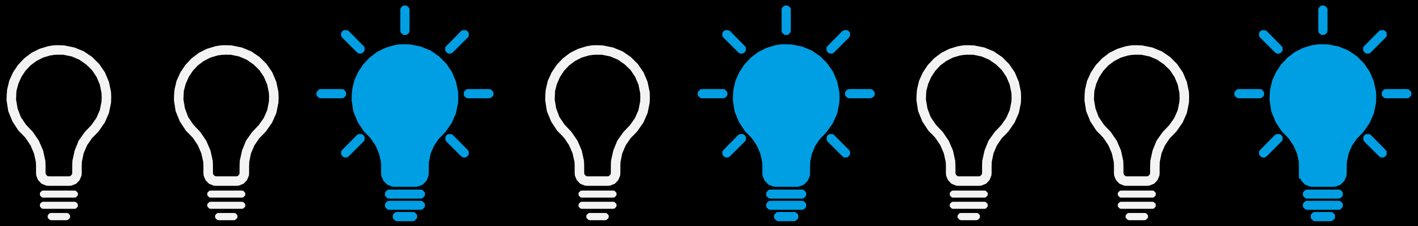


0 0 1 0 1 0 0 1



0 0 1 0 1 0 0 1

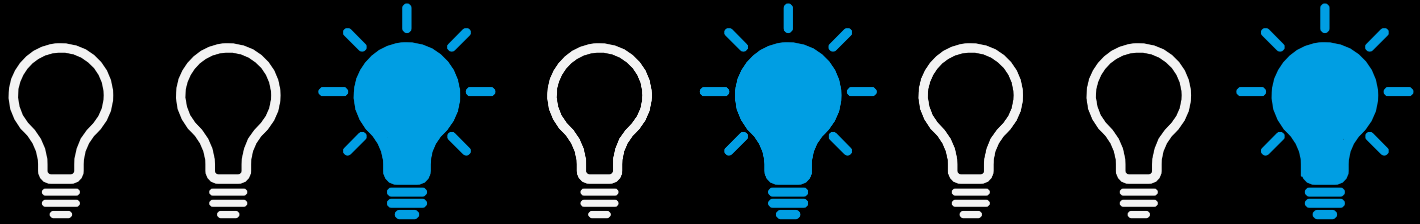
A **Bit** (binary digit)



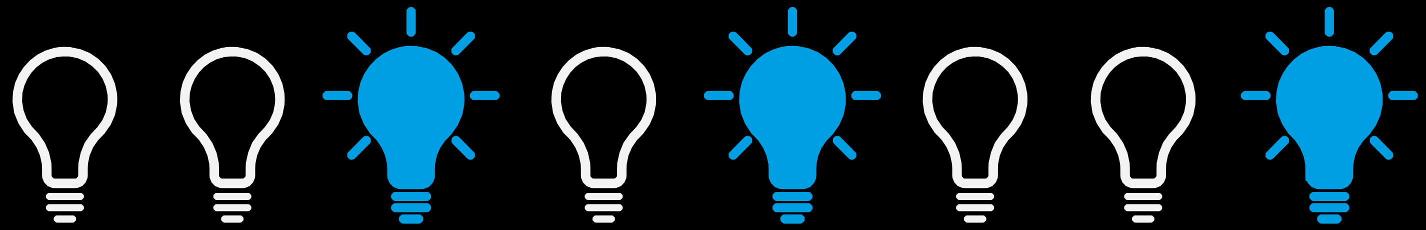
0 0 1 0 1 0 0 1

A **Bit** (binary digit)

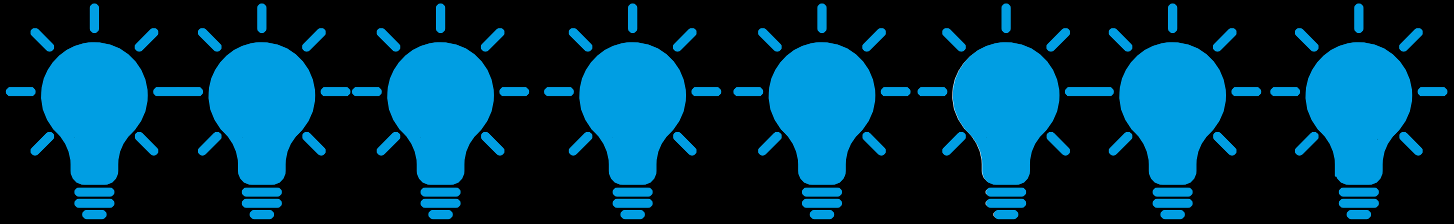
A **byte** (8 bits)



0	0	1	0	1	0	0	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

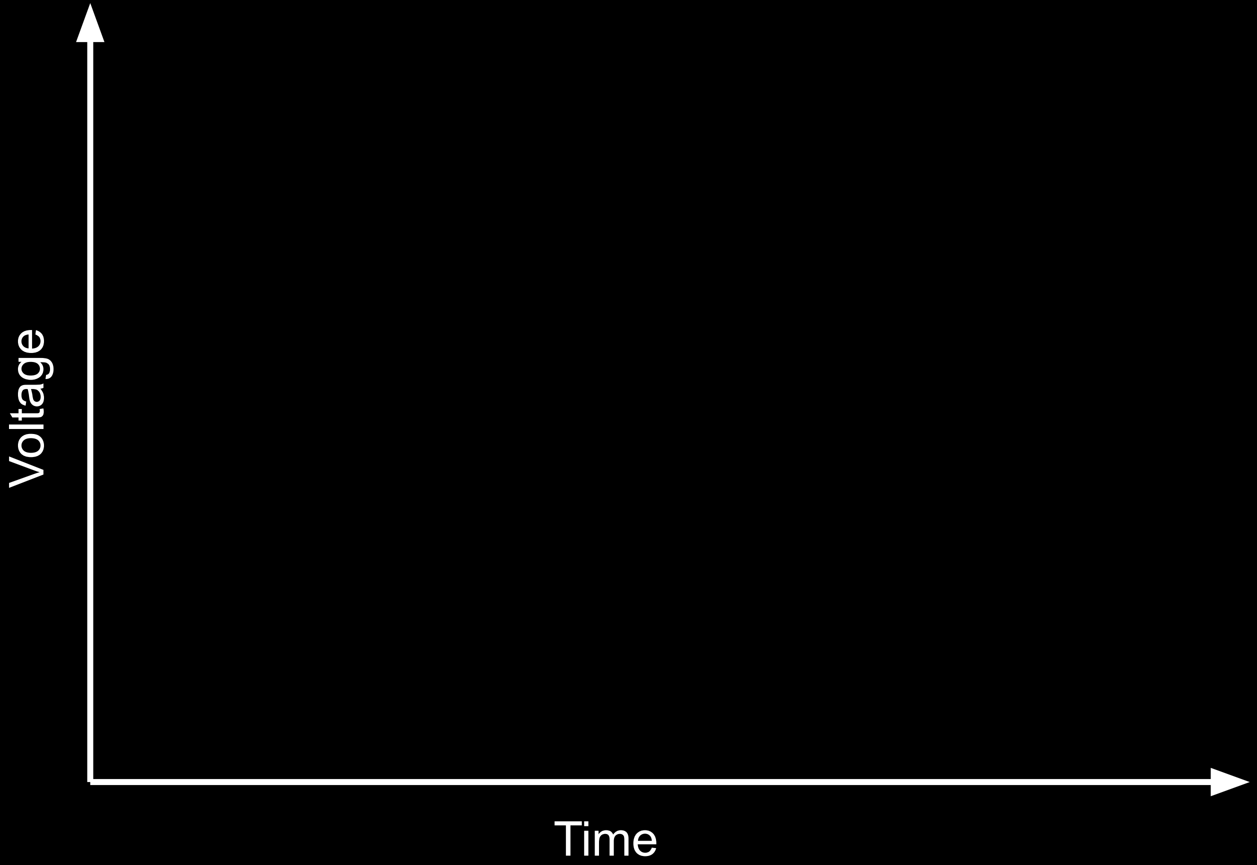


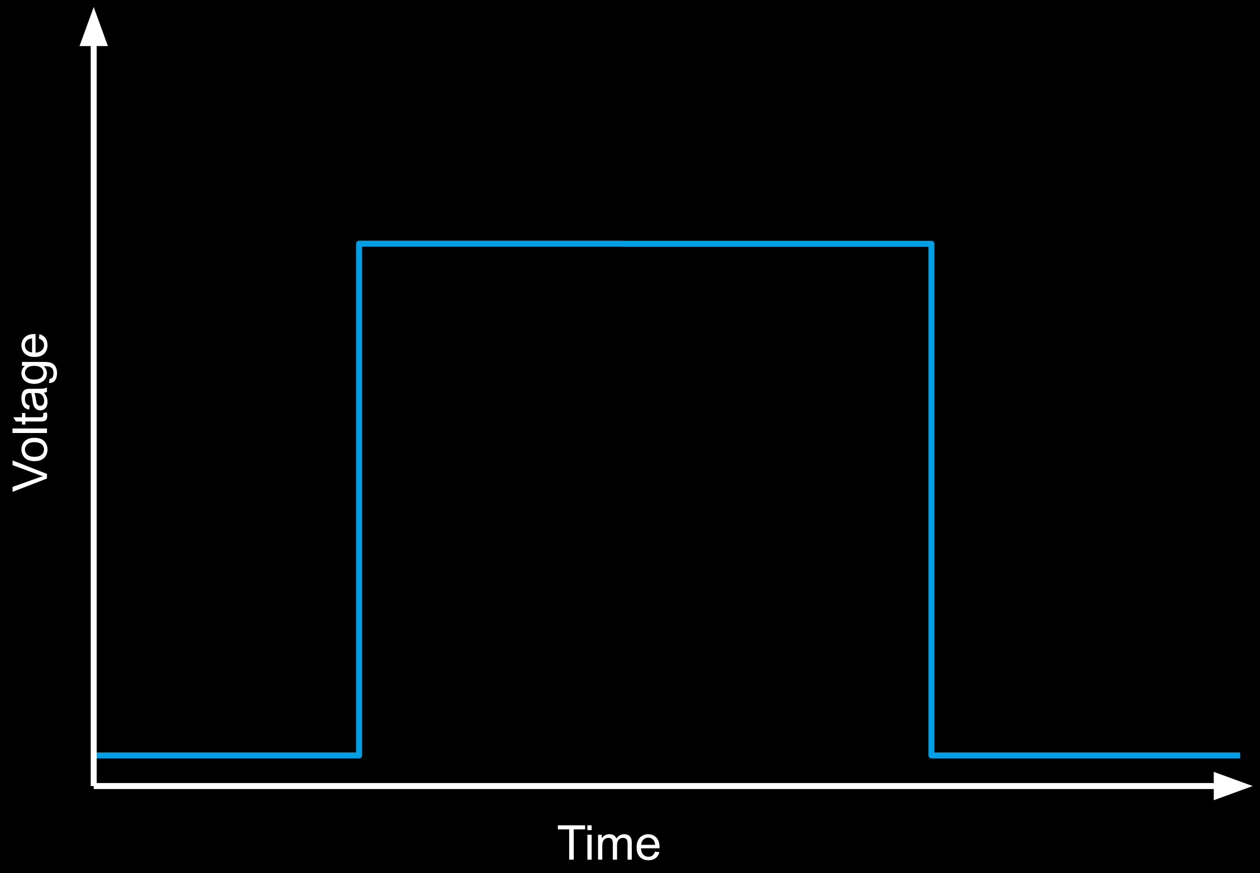
0	0	1	0	1	0	0	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

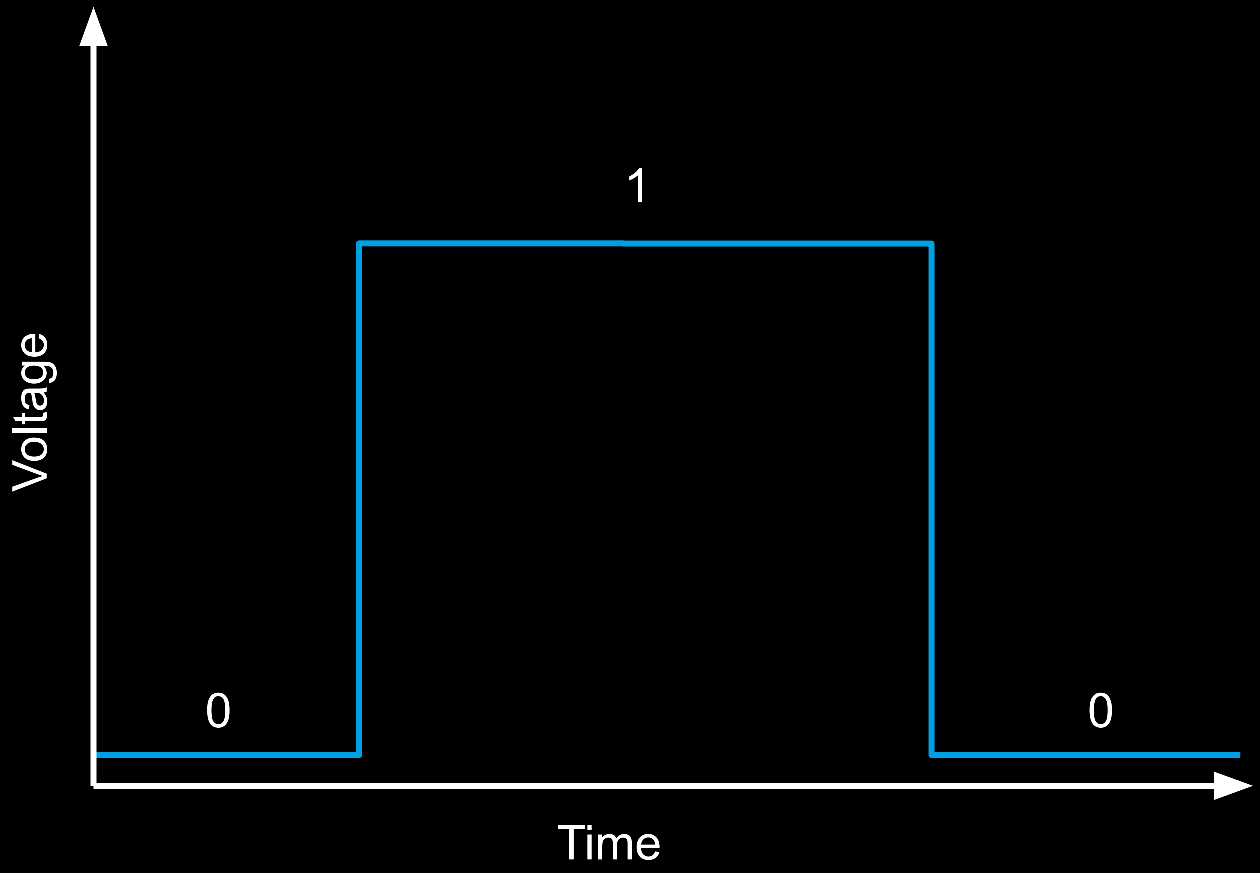


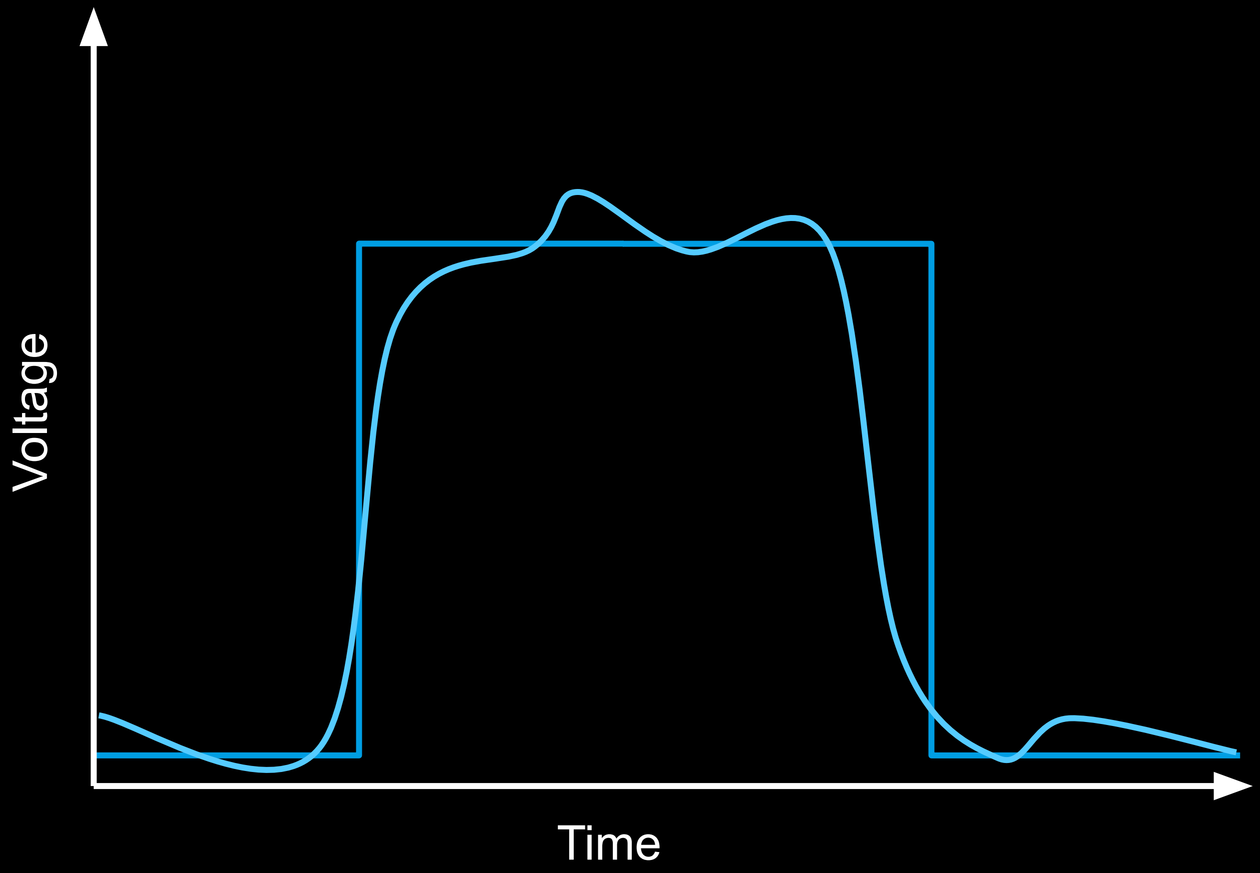
What can we store in one byte?

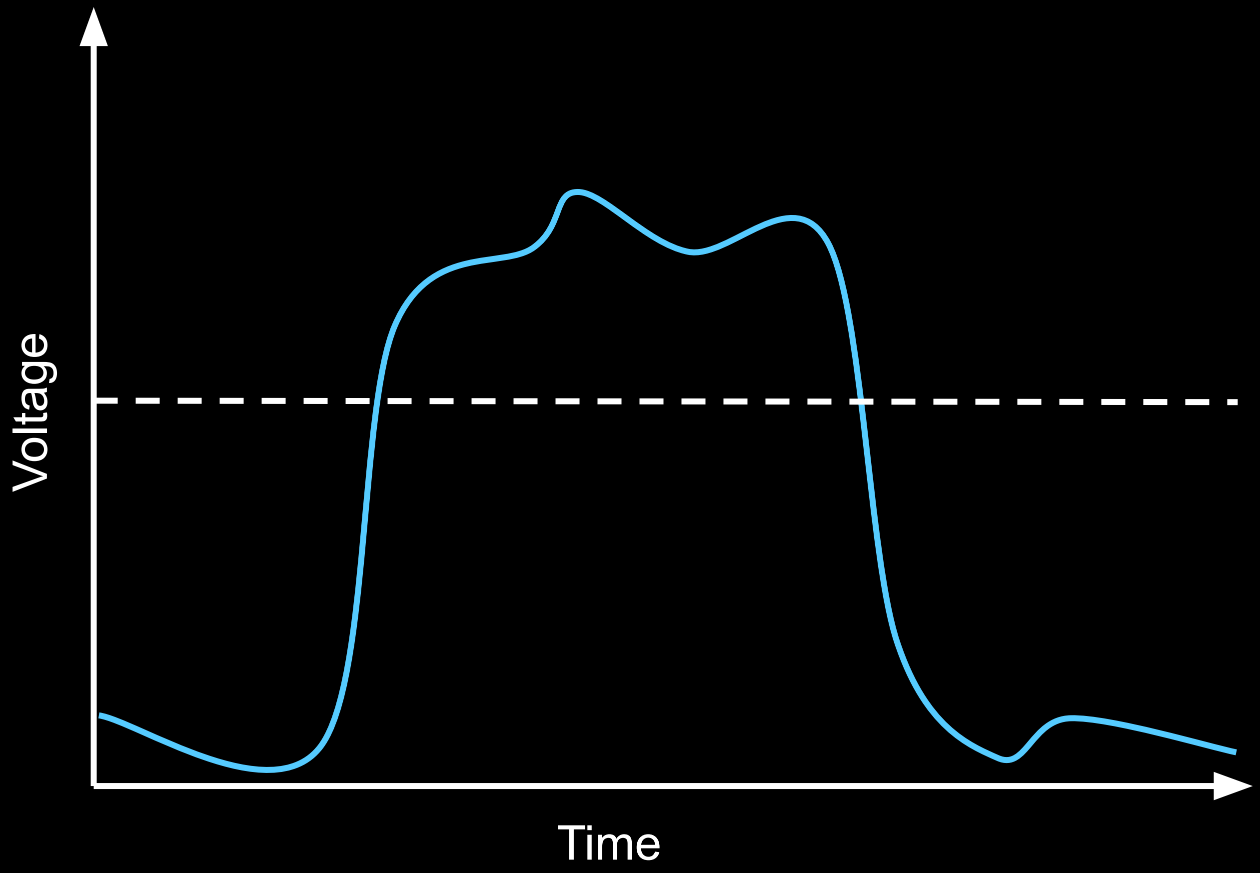
Are we stuck with binary?

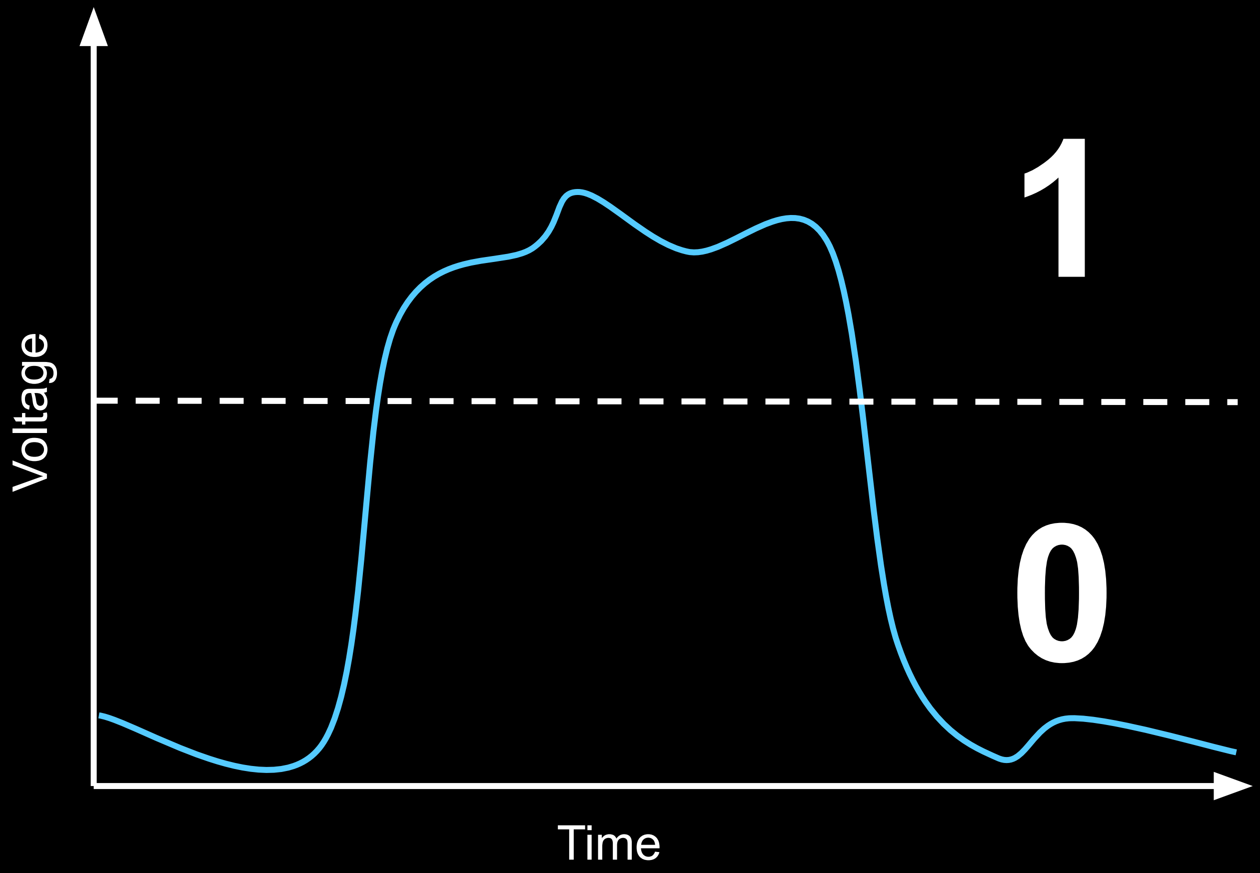

Voltage
Time

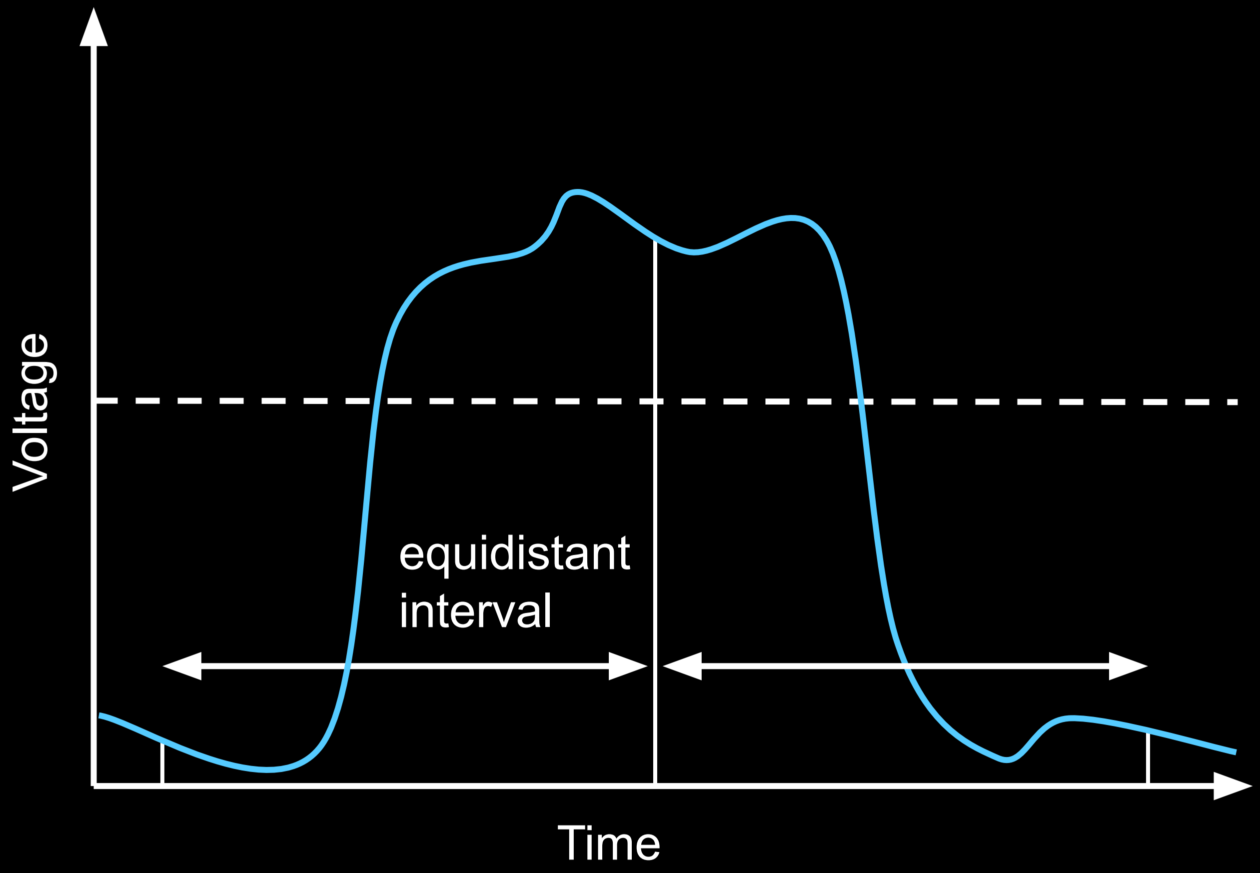

Voltage
Time

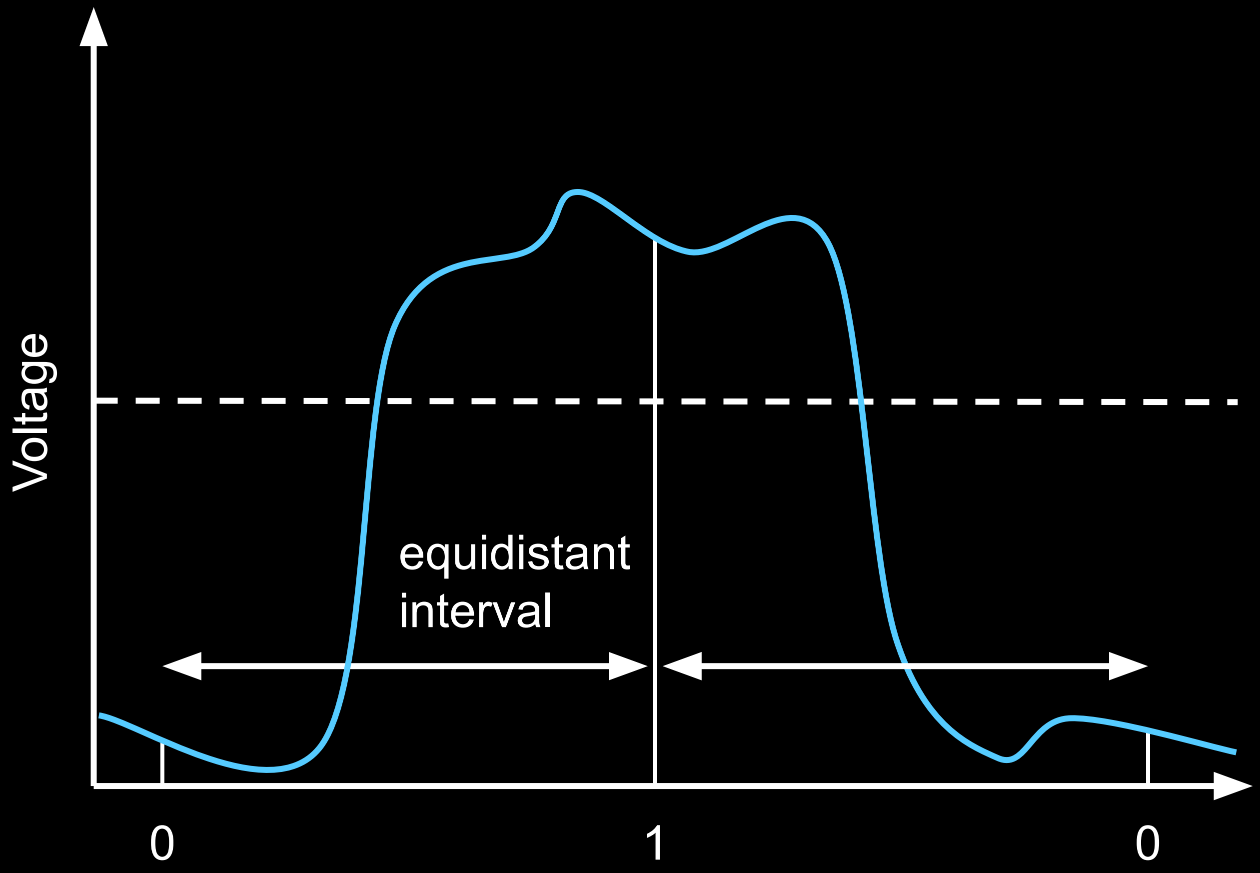

1
Voltage
0
0
Time

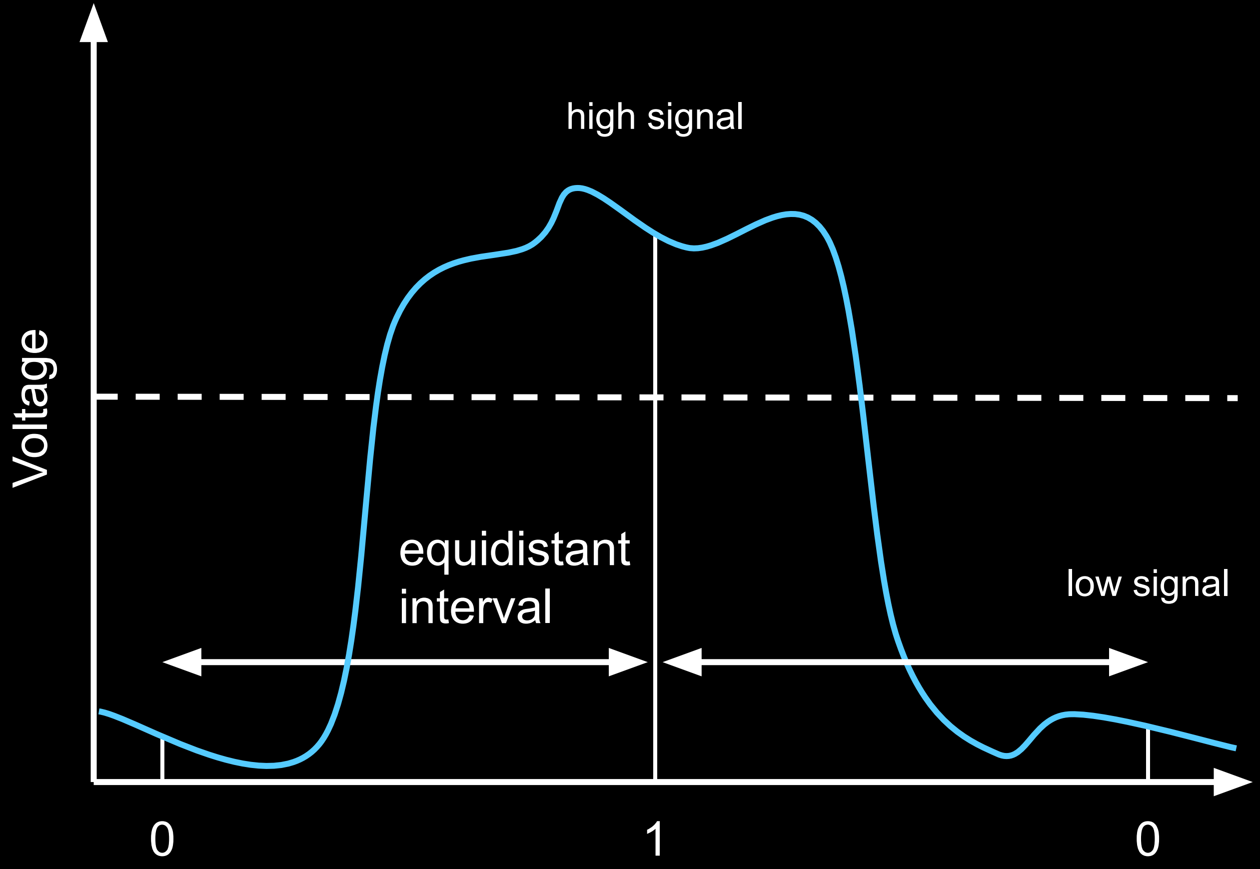

Voltage
Time


Voltage
Time


1
0
Voltage
Time

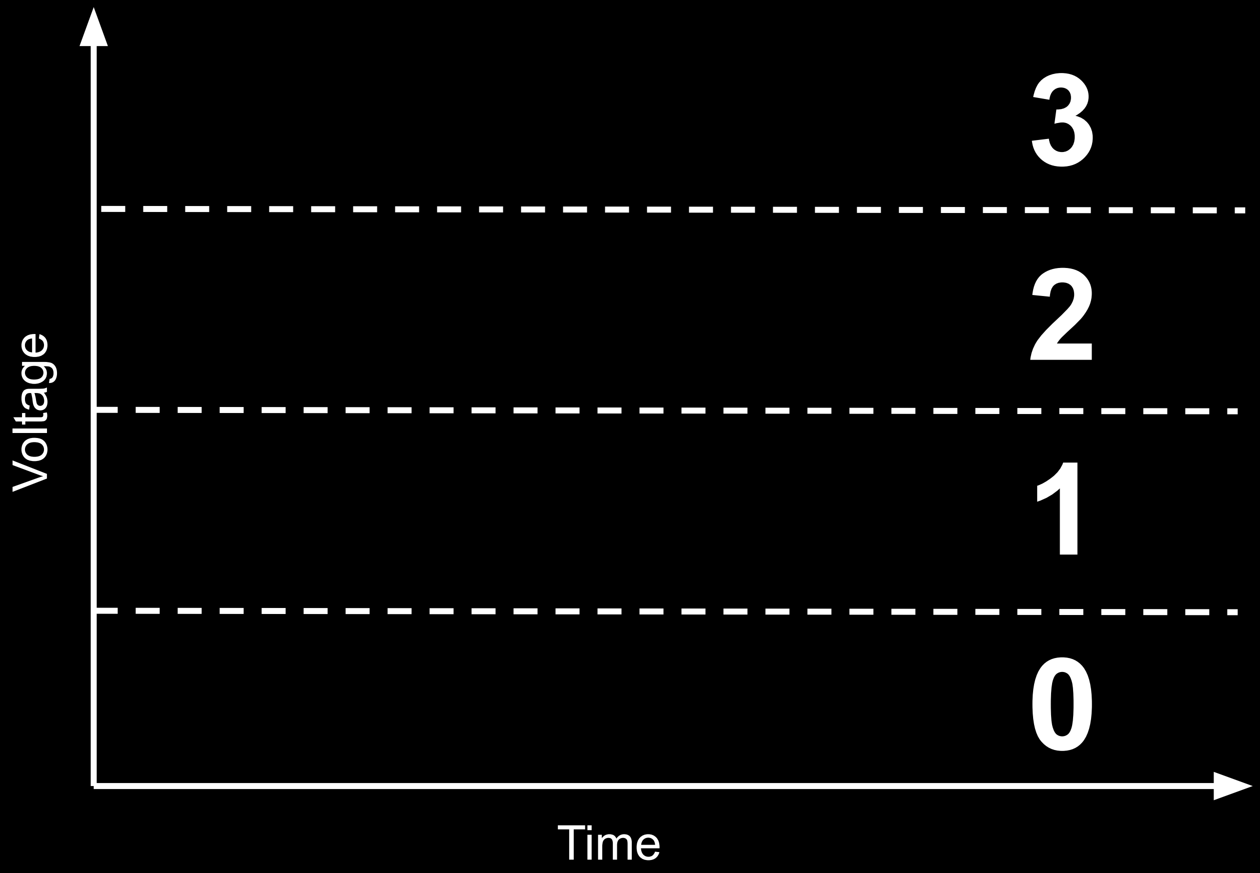

Voltage
equidistant
interval
Time

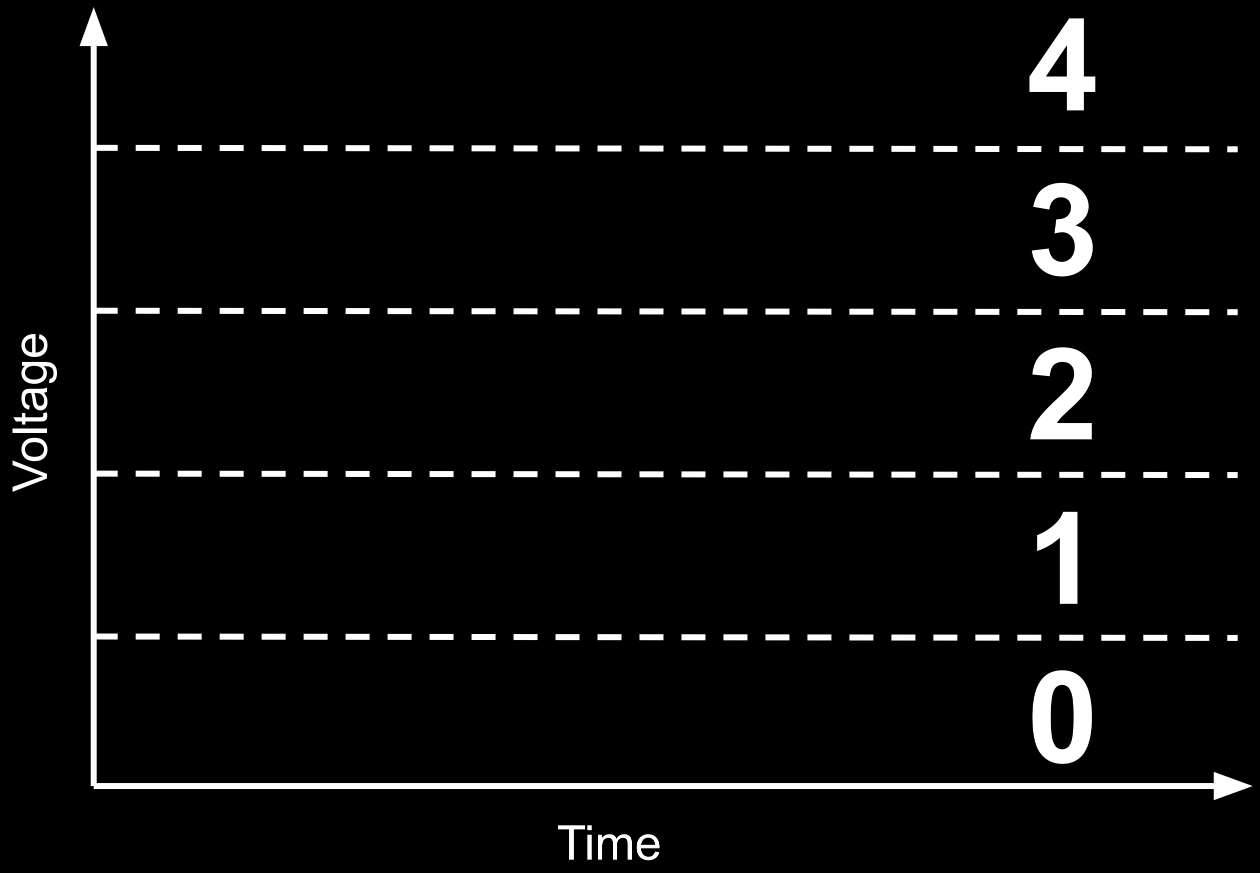

Voltage
equidistant
interval
0
1
0

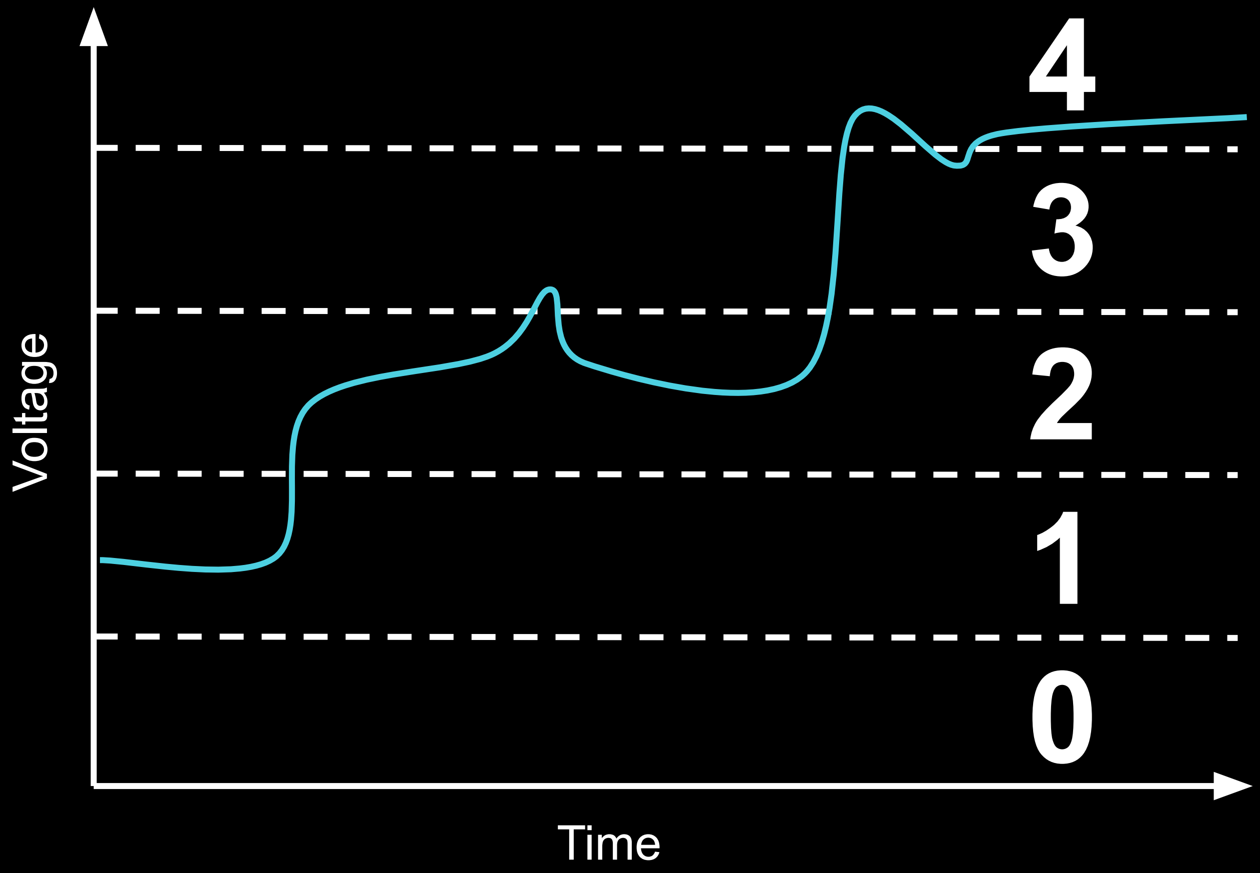

high signal
Voltage
equidistant
interval
low signal
0
1
0

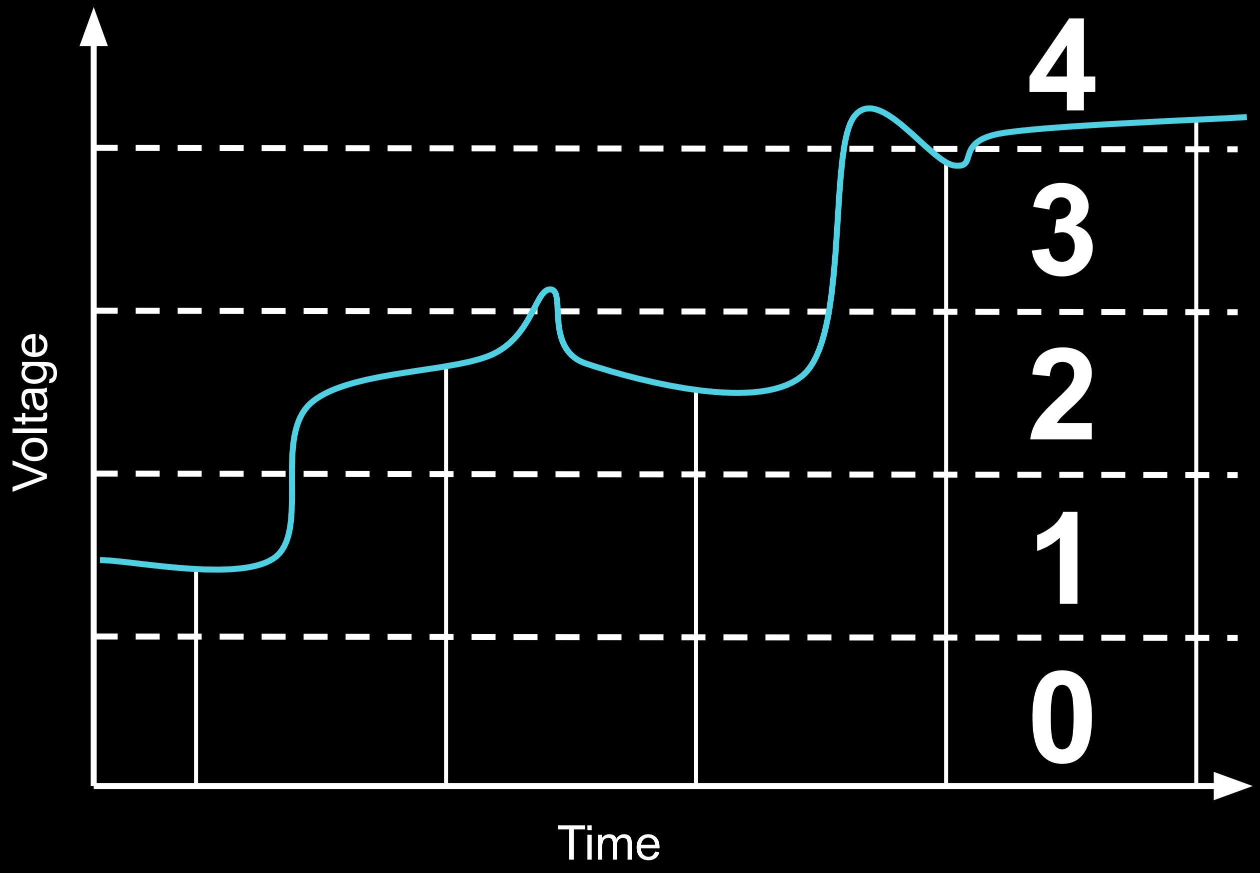
What about ternary?

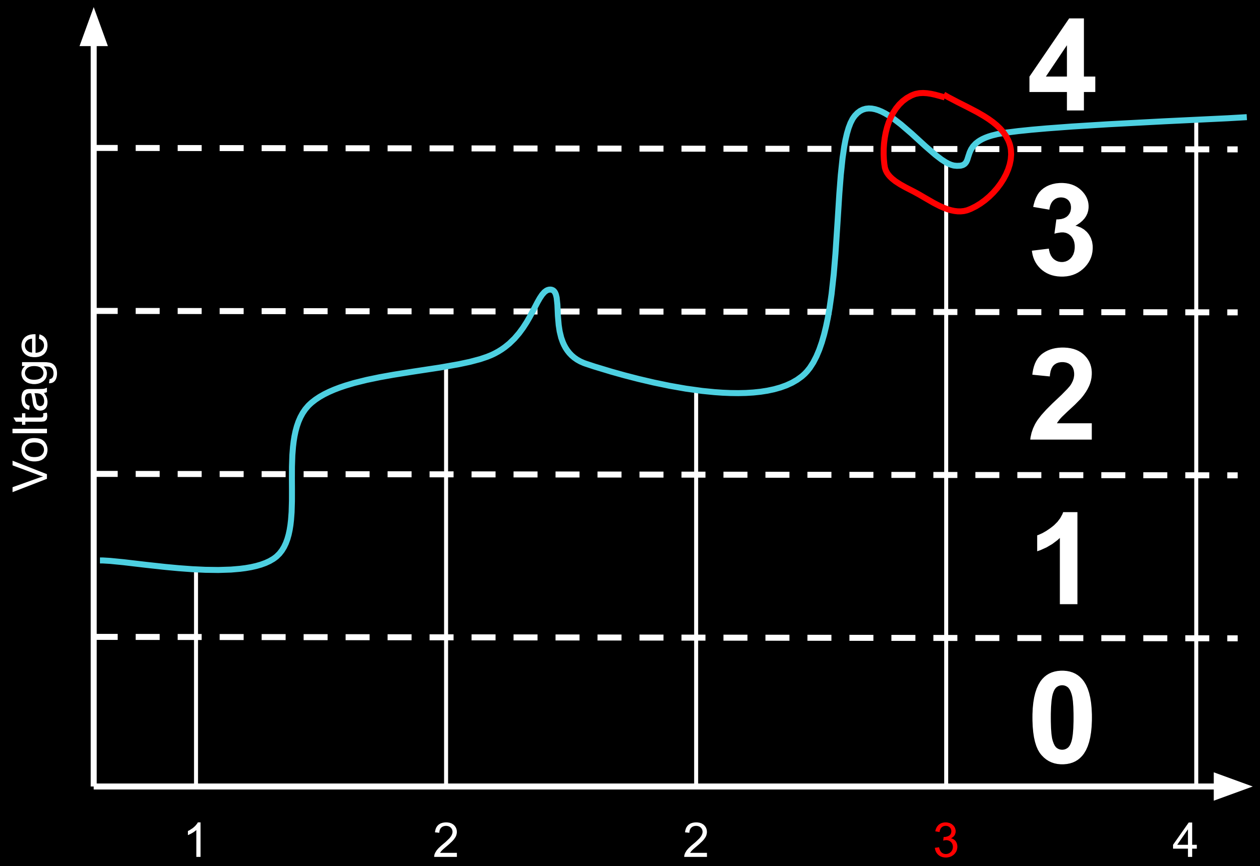

2
1
0
Voltage
Time


3
2
1
0
Voltage
Time


4
3
2
1
0
Voltage
Time

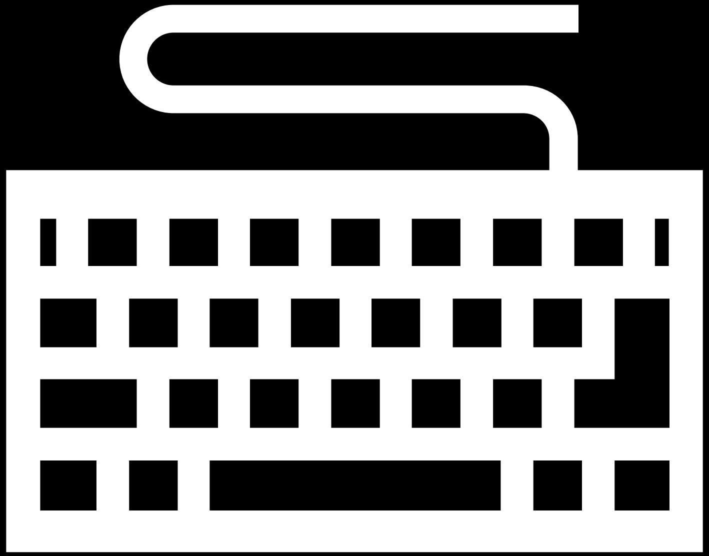

Voltage
Time
4
3
2
1
0


4
3
2
1
0
Voltage
Time


Voltage
4
3
2
1
0
1
2
2
3
4

CODES

• • • ▬ ▬ ▬ • • •



A	B	C	D	...	a	b	c	d
65	66	67	68		97	98	99	100

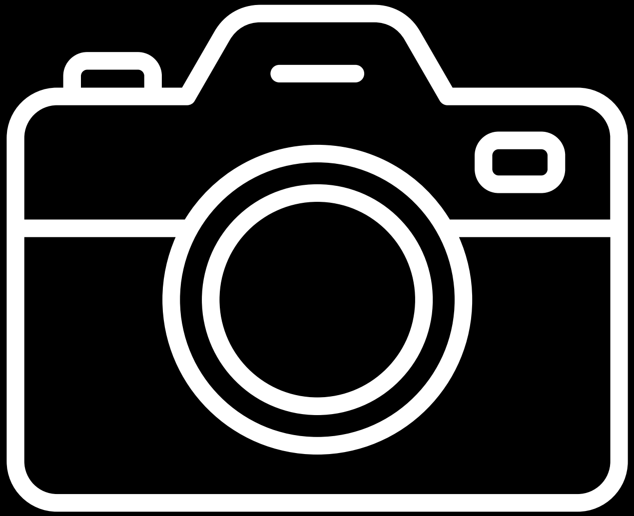
ASCII Code

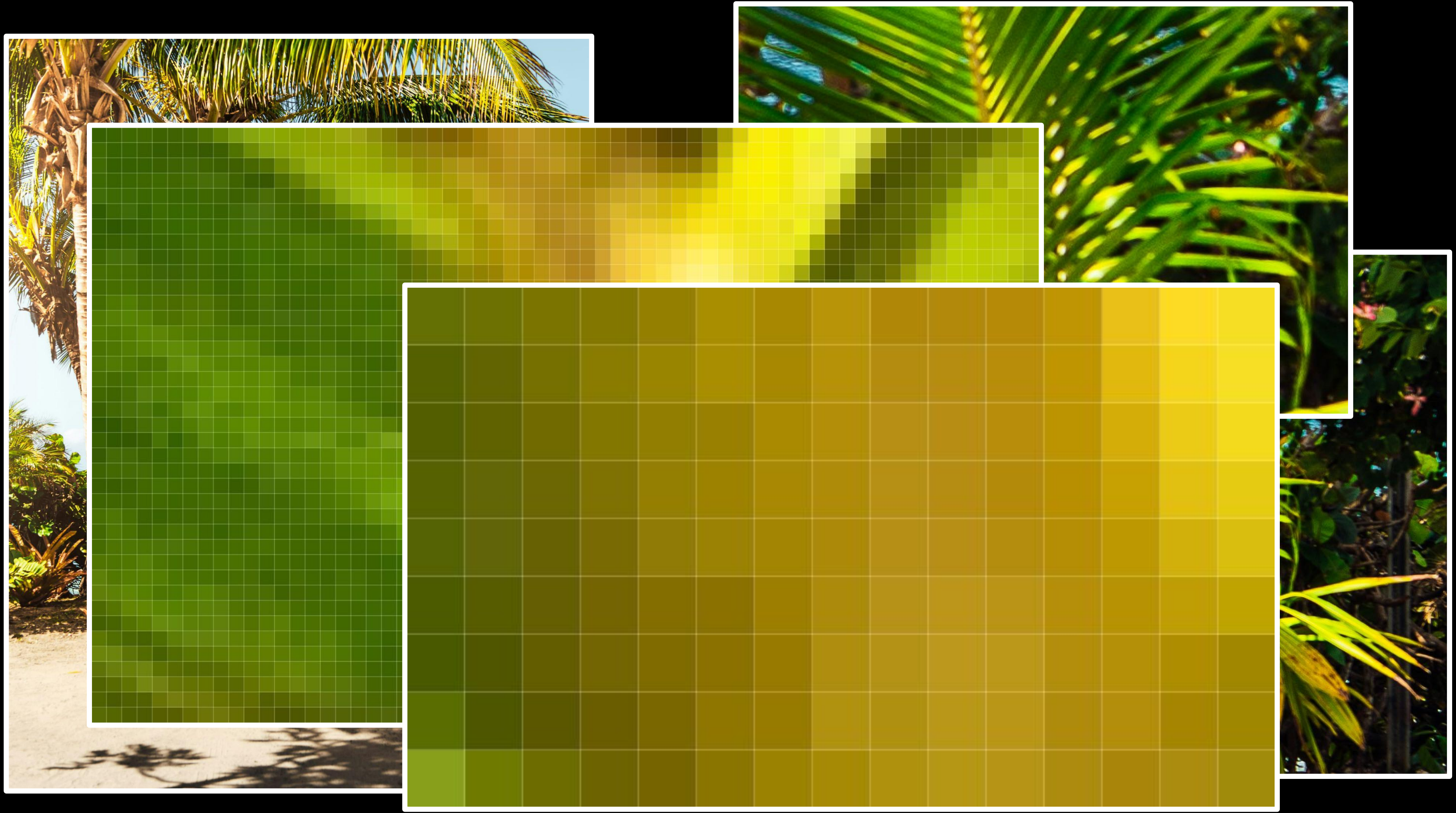
A	B	C	D	...	a	b	c	d
65	66	67	68		97	98	99	100

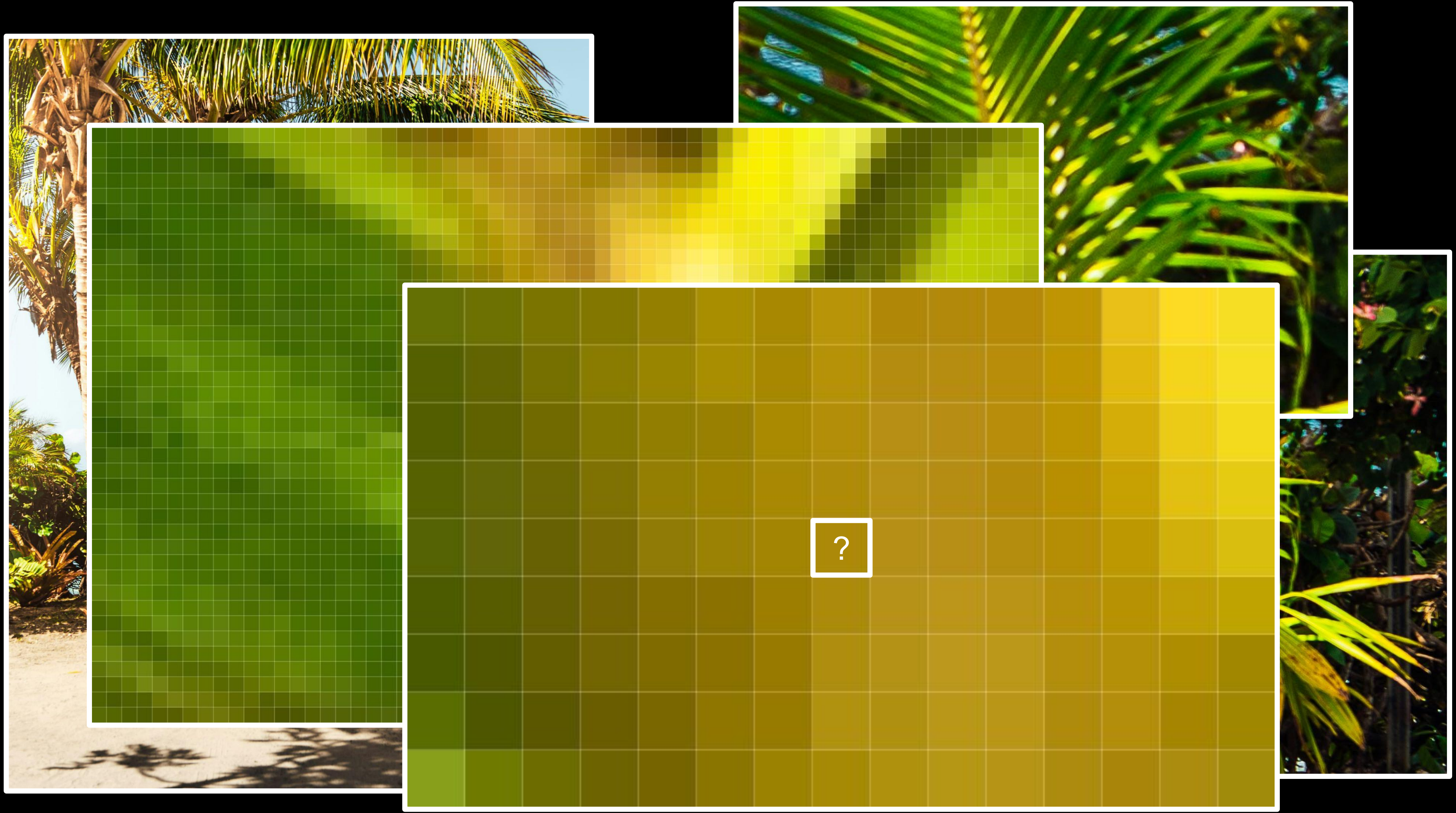
😀	😁	😂	😃	...	🙈	🙉	🙊	🙋
1F600	1F601	1F602	1F603		1F648	1F649	1F64A	1F64B

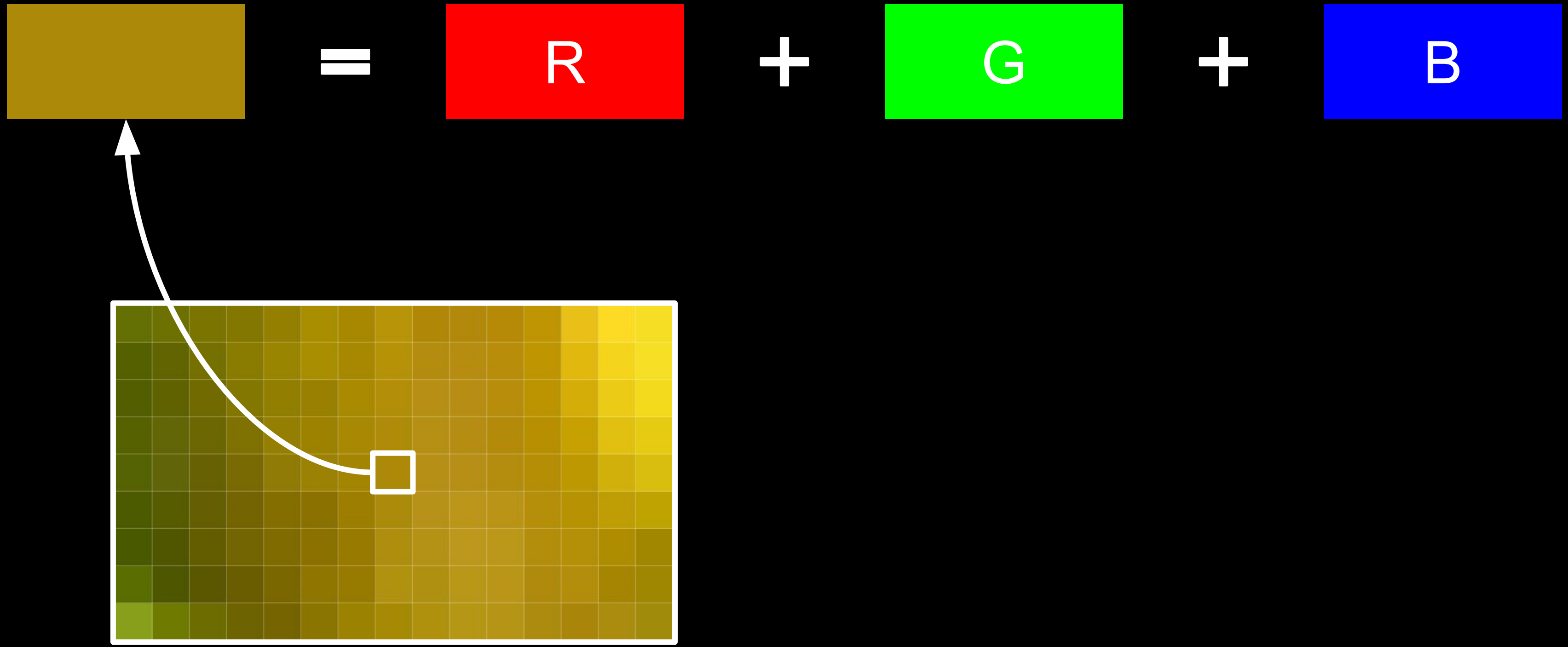
Unicode

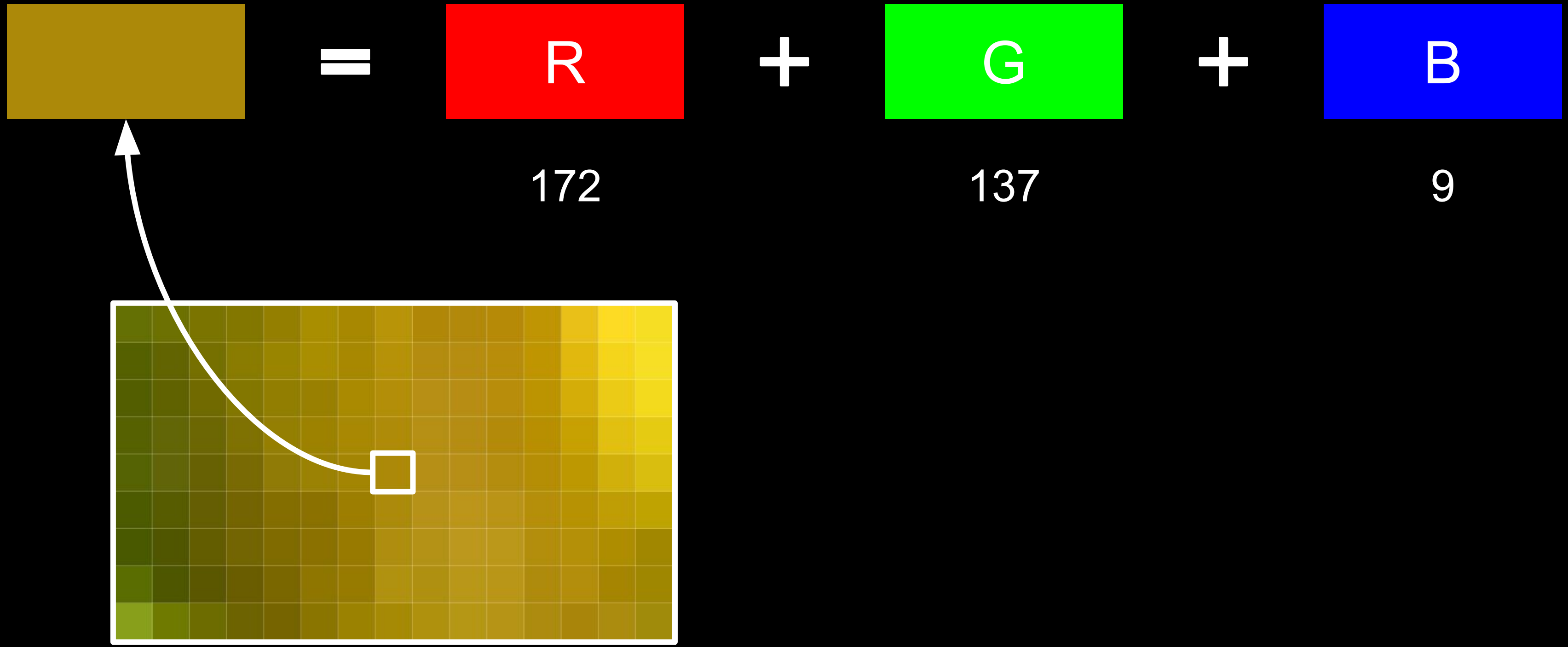
😀	😁	😂	😃	…	🙈	🙉	🙊	🙋
1F600	1F601	1F602	1F603		1F648	1F649	1F64A	1F64B

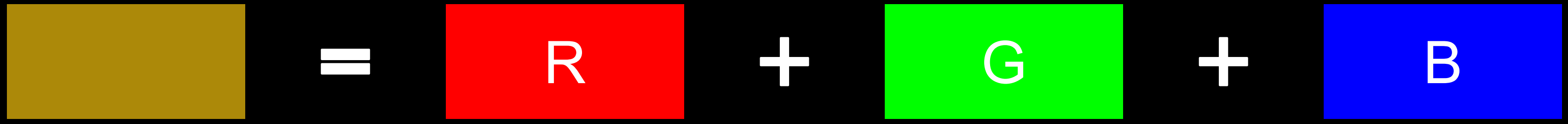






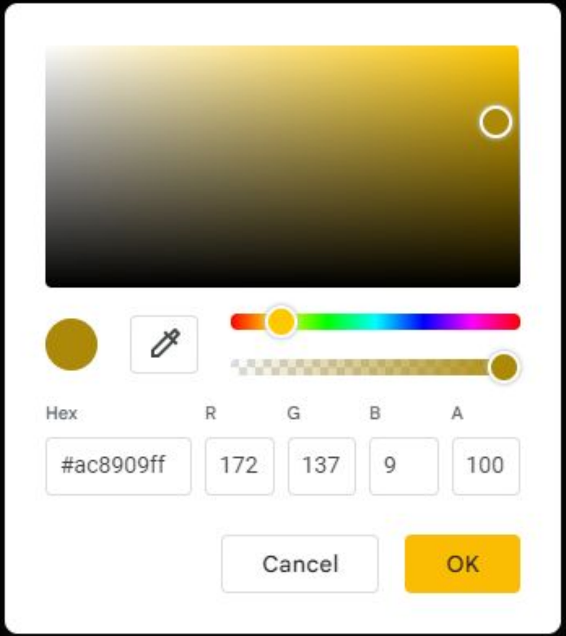

R
G
B

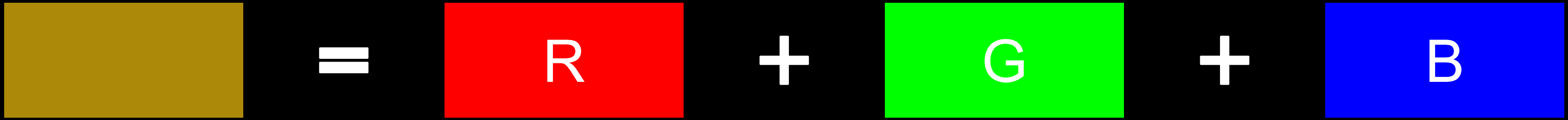

=
R
+
G
+
B
172
137
9



172 137 9

#AC8909


Hex
R
G
B
A
#ac8909ff
172
137
9
100
Cancel
OK



	172	137	9
#AC8909	AC	89	09

	=	R	+	G	+	B
		172		137		9
#AC8909		AC		89		09
		10101100		01011001		00001001

possible colors?

R

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

R

G

B

2^{23} 2^{22} 2^{21} 2^{20} 2^{19} 2^{18} 2^{17} 2^{16} 2^{15} 2^{14} 2^{13} 2^{12} 2^{11} 2^{10} 2^{9} 2^{8} 2^{7} 2^{6} 2^{5} 2^{4} 2^{3} 2^{2} 2^{1} 2^{0}

R

G

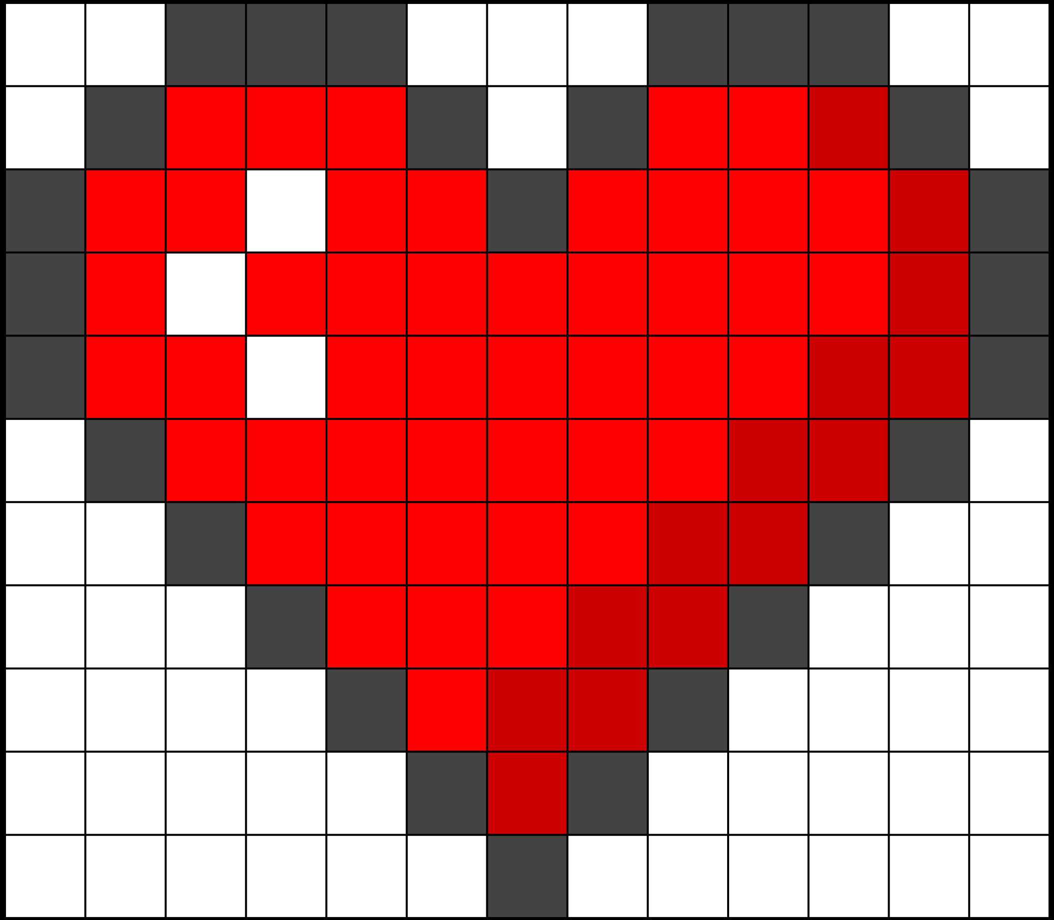
B

$2^{23}\ \underbrace{2^{22}\ 2^{21}\ 2^{20}\ 2^{19}\ 2^{18}\ 2^{17}\ 2^{16}\ \ 2^{15}\ 2^{14}\ 2^{13}\ 2^{12}\ 2^{11}\ 2^{10}\ 2^{9}\ 2^{8}\ \ 2^{7}\ 2^{6}\ 2^{5}\ 2^{4}\ 2^{3}\ 2^{2}\ 2^{1}\ 2^{0}}$

8.388.608 + 8.388.607 = 16.777.215

R G B

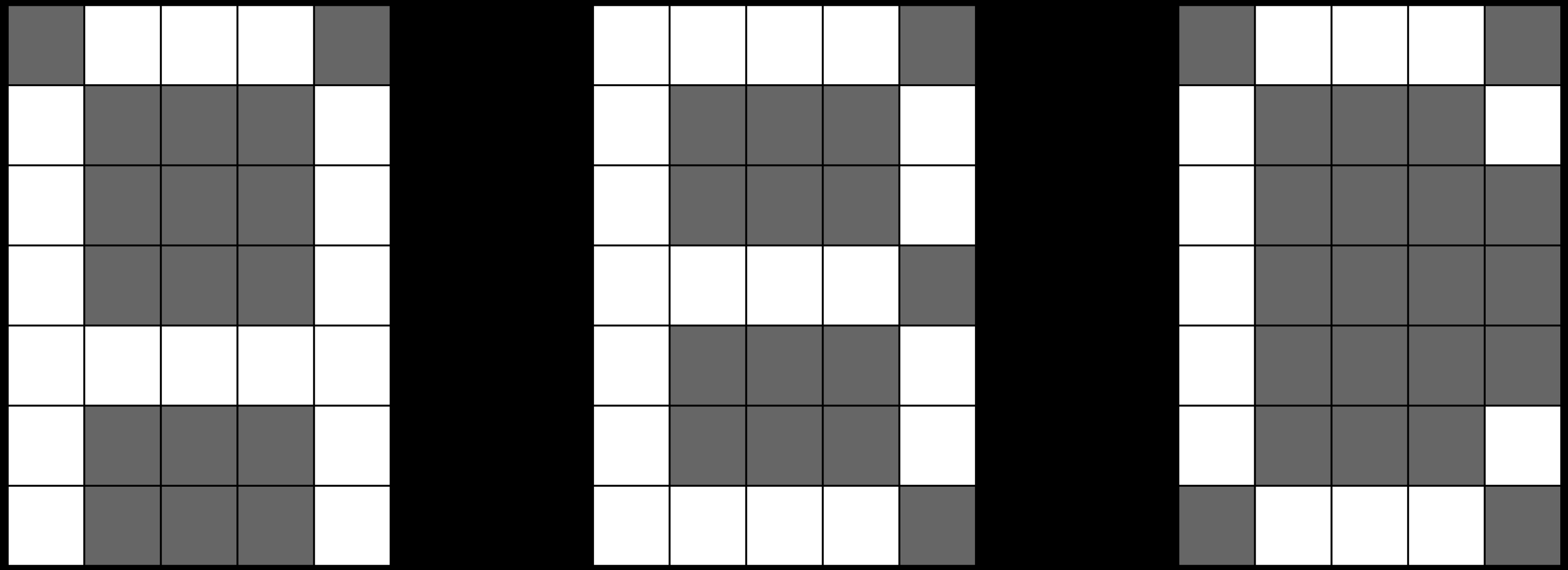
$$\underbrace{2^{23}\ 2^{22}\ 2^{21}\ 2^{20}\ 2^{19}\ 2^{18}\ 2^{17}\ 2^{16}}_{256} \times \underbrace{2^{15}\ 2^{14}\ 2^{13}\ 2^{12}\ 2^{11}\ 2^{10}\ 2^{9}\ 2^{8}}_{256} \times \underbrace{2^{7}\ 2^{6}\ 2^{5}\ 2^{4}\ 2^{3}\ 2^{2}\ 2^{1}\ 2^{0}}_{256}$$

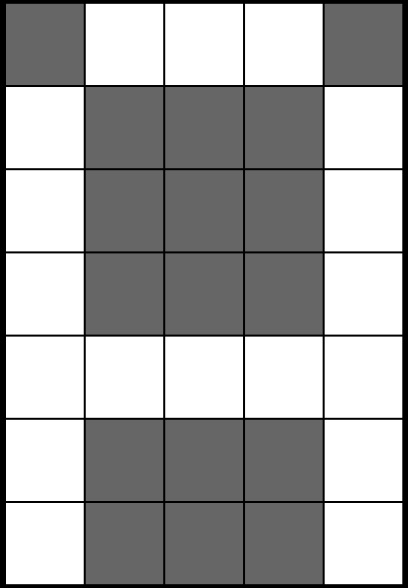


CS50 Pixel Art

🤔

compression?





0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1

0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1

0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 0
0 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1

5

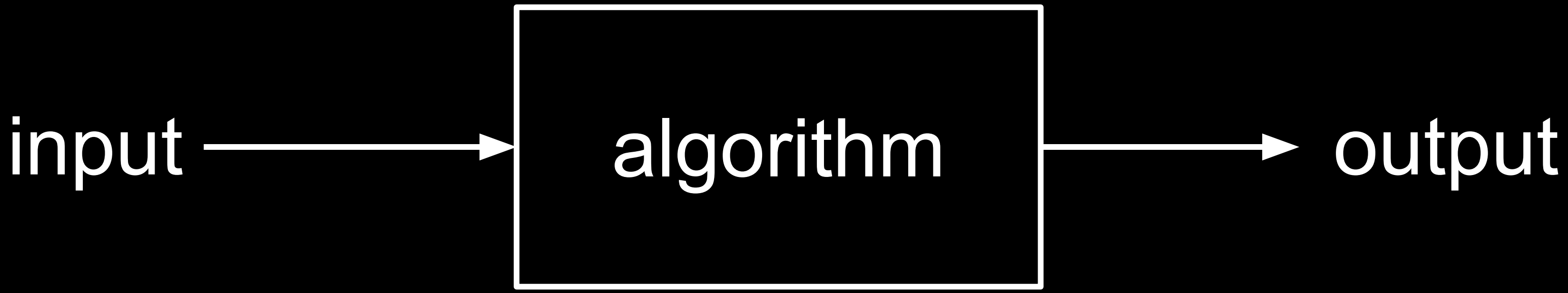
7

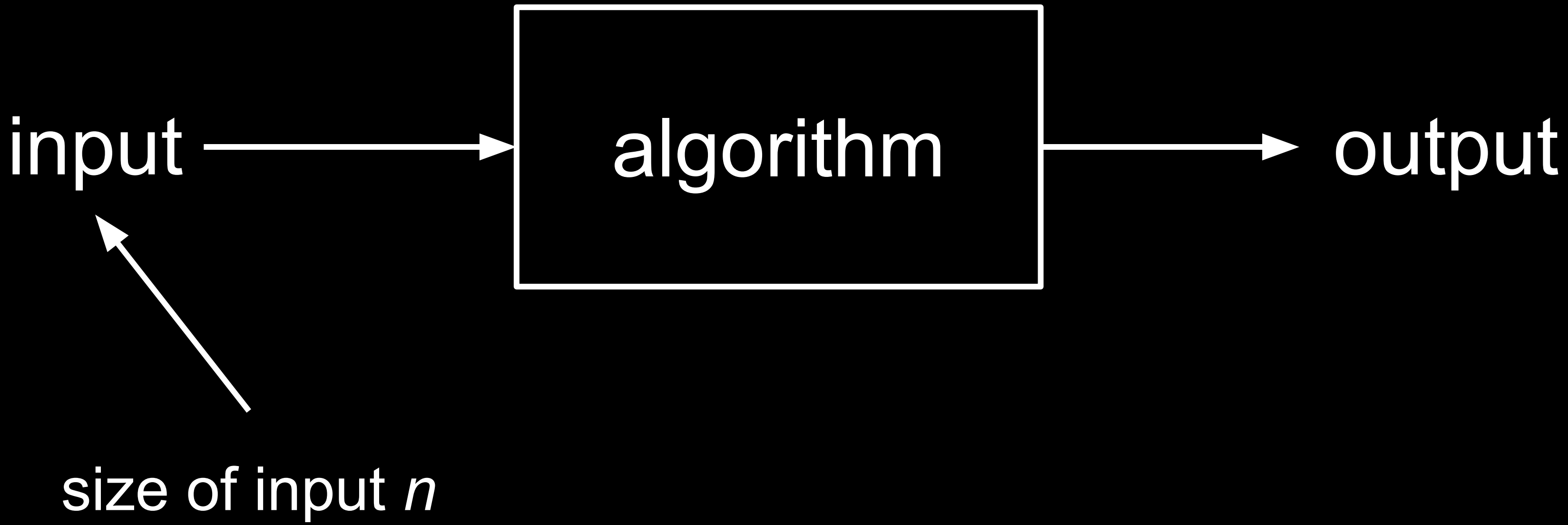
0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1

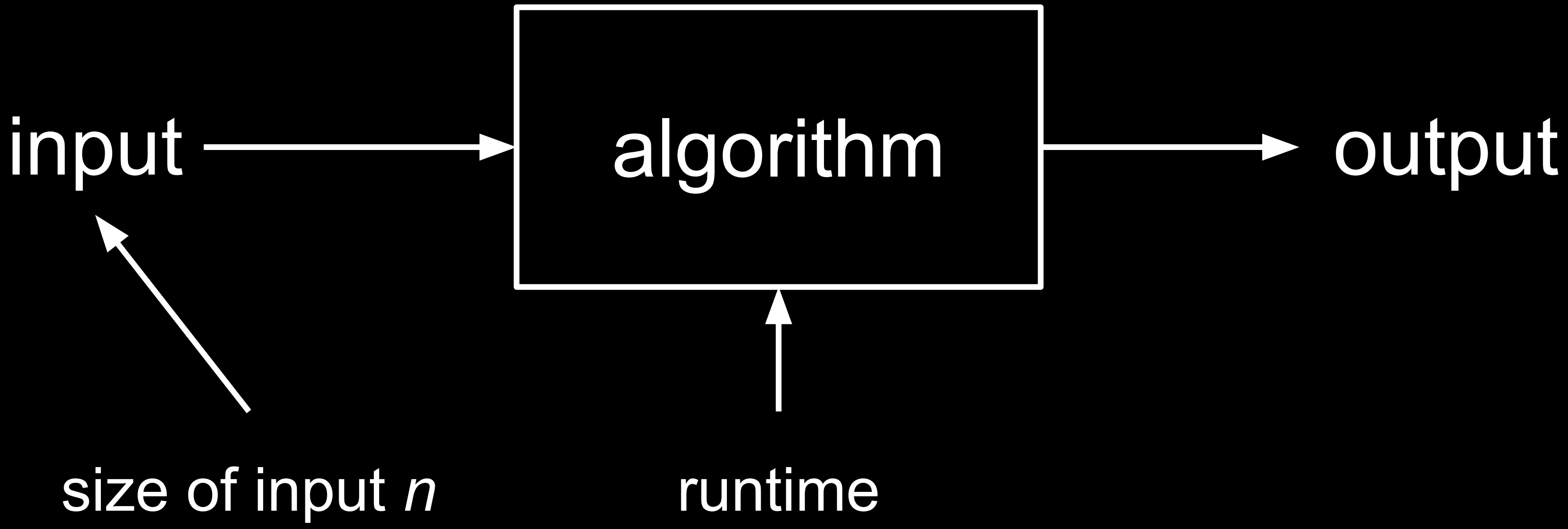
0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 0
0 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1

ALGORITHMS

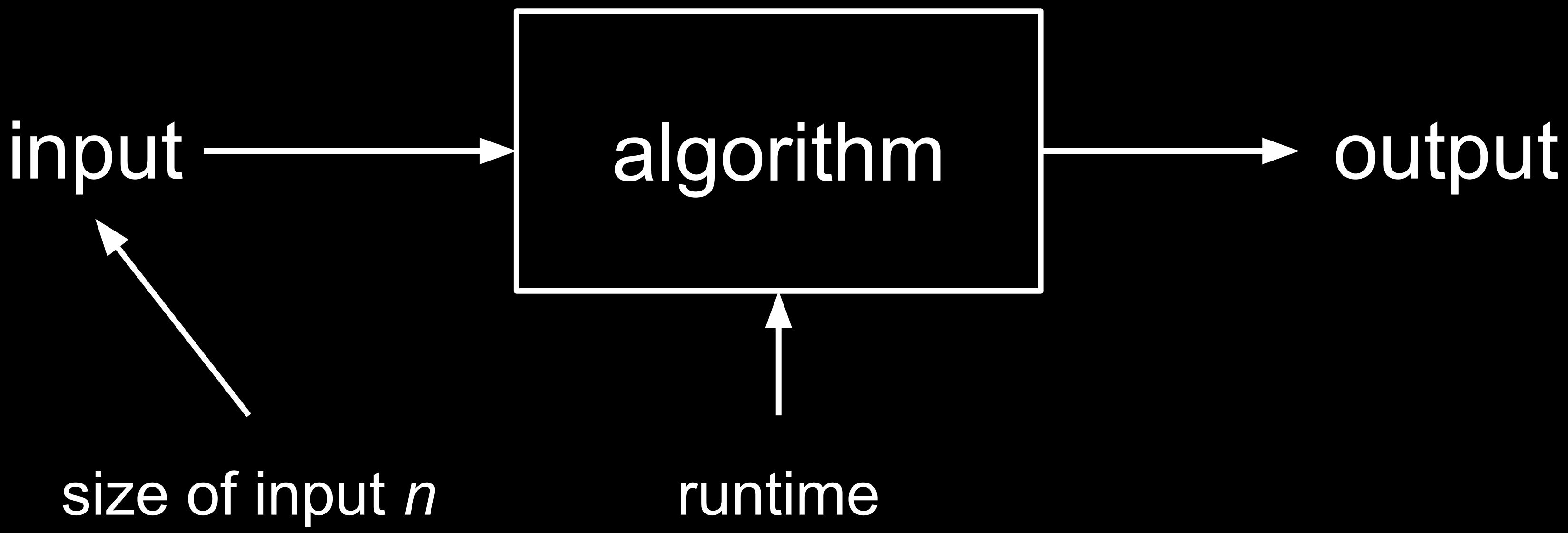
complexity


input
algorithm
output


input
algorithm
output
size of input *n*


input
algorithm
output
size of input *n*
runtime

O(n)



COMPUTERS

ARITHMETIC

MEMORY

ANALOG VS. DIGITAL