

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

The slides are meant as visual support for the lecture.
They are neither a documentation nor a script.

Please do not print the slides.

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

MOTIVATION

a few
experts

digitally uneducated
society



digitally illiterate society with a few experts

collective understanding

you?



society with a distributed and high degree of digital education

representation

representation

processing

representation

processing

programming

representation

processing

programming

digital fundamentals

data analysis

representation

processing

programming

digital fundamentals

data analysis

artificial
intelligence

representation

processing

programming

digital fundamentals

digital applications

data analysis

artificial
intelligence

representation

processing

programming

digital fundamentals

digital applications

data analysis

artificial
intelligence

representation

processing

programming

digital fundamentals



digital applications

data analysis

artificial
intelligence



representation

processing

programming



digital fundamentals



PROBLEM SOLVING

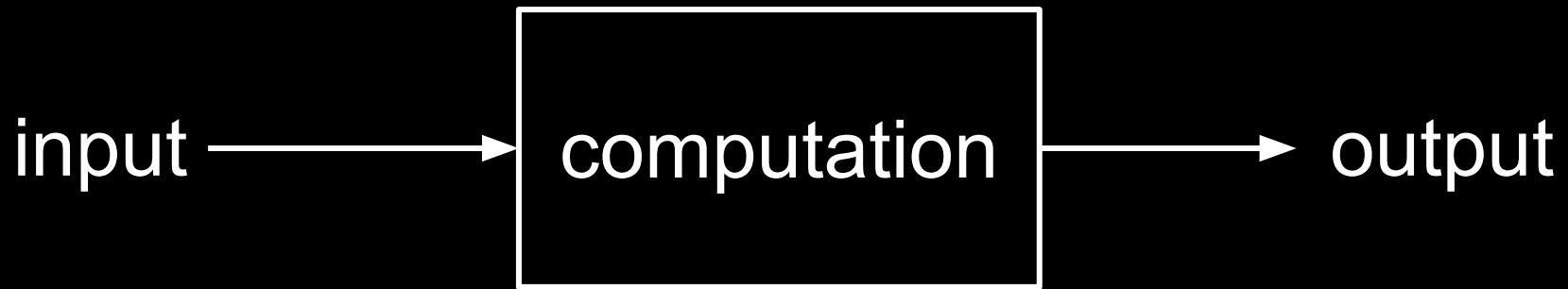
a model for solving problems



a model for solving problems



a model for solving problems









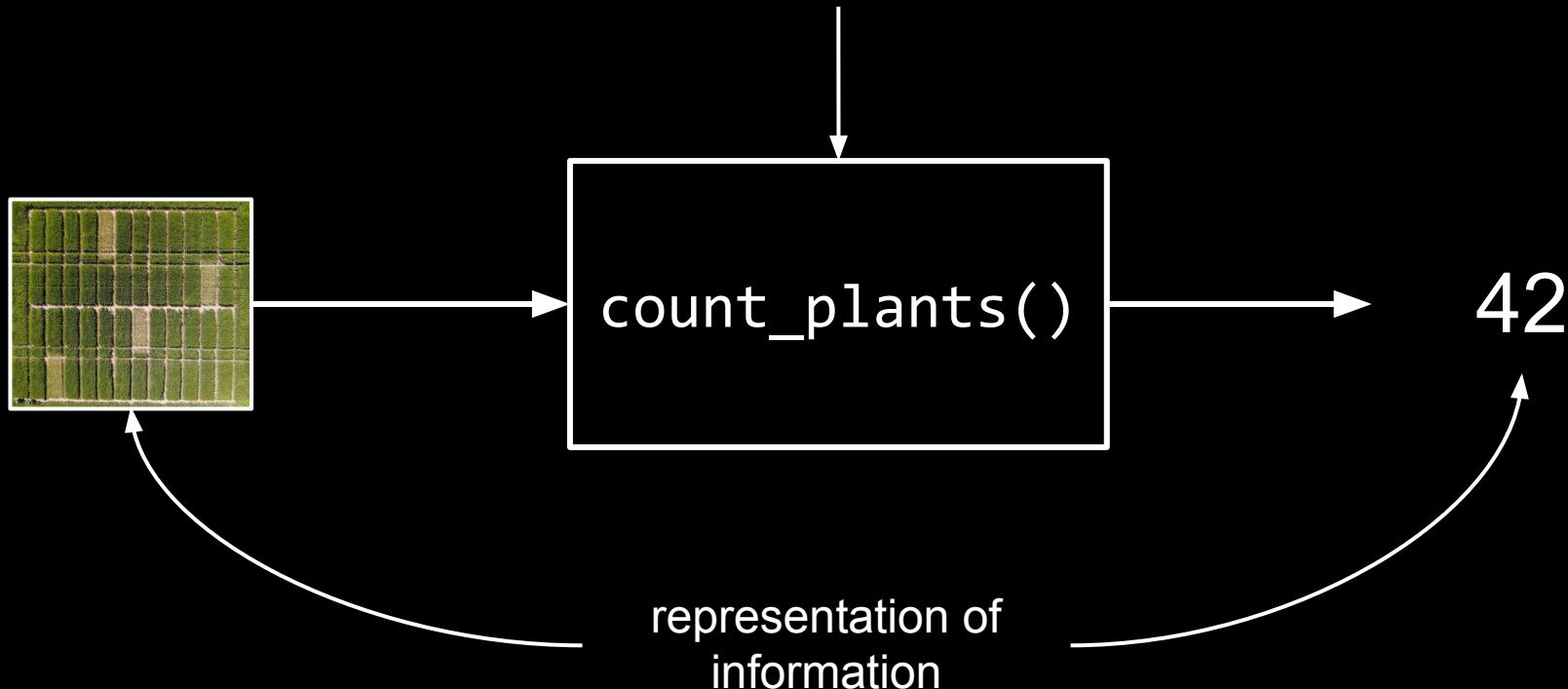
output



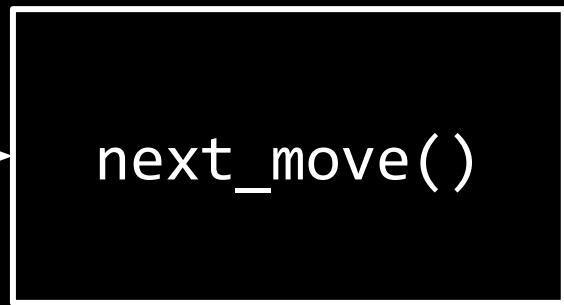
count_plants()

42

processing of
information







E2 → E4

problem solving strategies

divide and conquer

large and complex problem





sorted list +
element



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

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



large and complex
problem

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

large and complex problem

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

smaller problem

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41

smaller problem

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

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



$$67 \neq 71$$

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



$$67 \neq 71$$

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



$$67 \neq 59$$

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



$$67 = 67$$



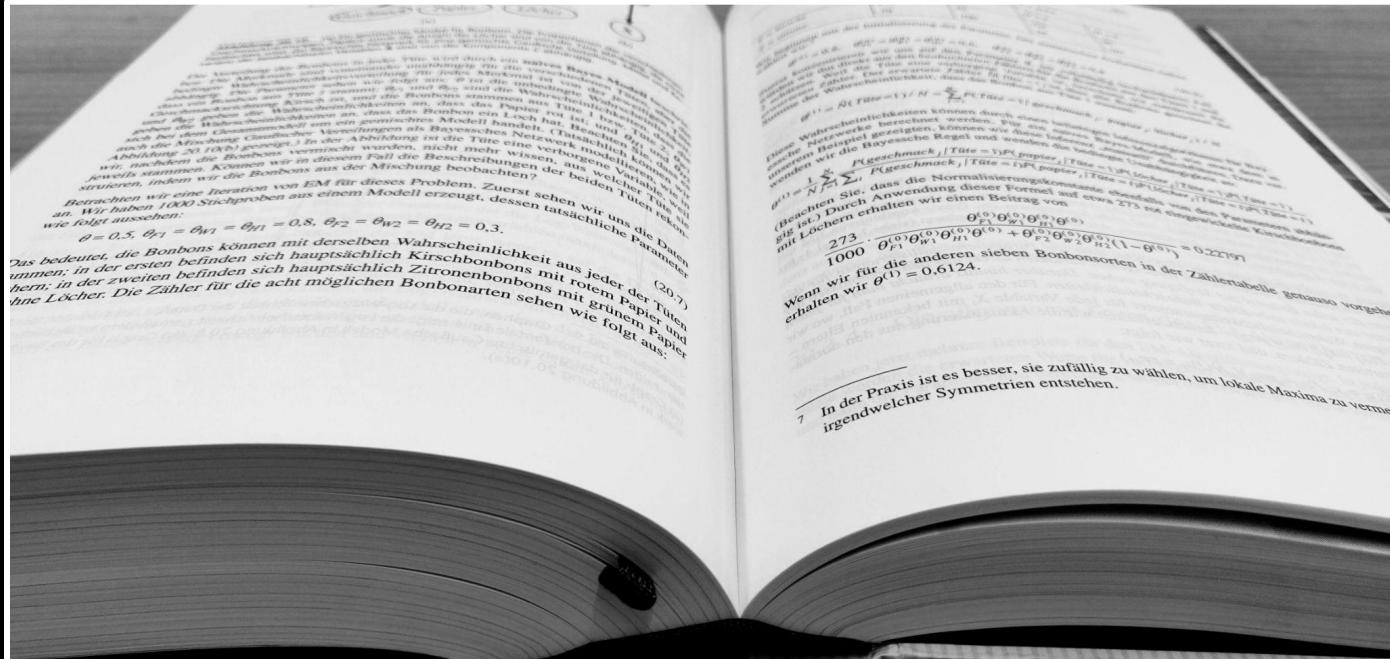
how efficient are linear and
binary search in general?



count_words()

word count

how many words are in the book?



strategies, anyone?



7 In der Praxis ist es besser, sie zufällig zu wählen, um lokale Maxima zu vermeiden.



page 1







187

212



page 1



page 2



187



page 1

212

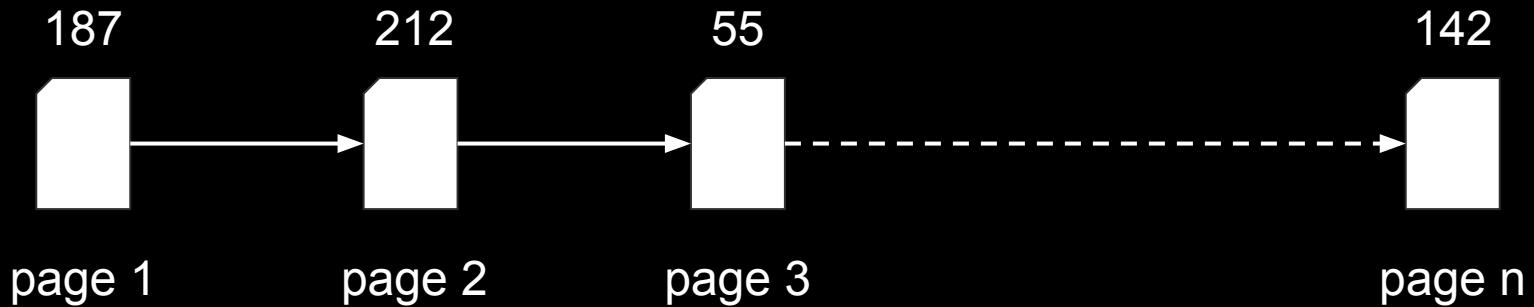


page 2

55



page 3





$n = 1327$ pages

$\varnothing 2:23$ minutes per page

~ 52.34 hours

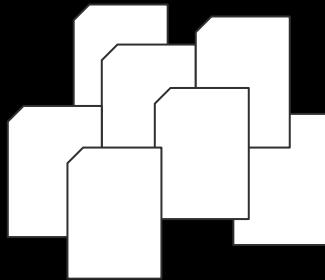


divide and conquer

+

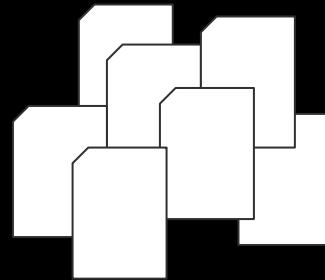
?

pages 1 - 700



student 1

pages 701 - 1327



student 2

pages 1 - 350

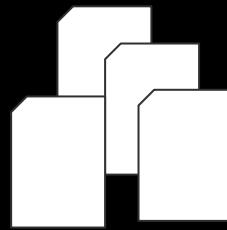
pages 351 - 700

pages 701 - 1050

pages 1051- 1327



student 1



student 2



student 3



student 4

divide and conquer

+

distribution and parallelization



INFORMATION

“Information is that which allows you to make a correct prediction with accuracy better than chance.”

Adami, Christoph. “What Is Information?” Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 374, no. 2063, Mar. 2016, p. 20150230, <https://doi.org/10.1098/rsta.2015.0230>.

“Information is that which allows you to make a correct prediction with accuracy better than chance.”

Adami, Christoph. “What Is Information?” Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 374, no. 2063, Mar. 2016, p. 20150230, <https://doi.org/10.1098/rsta.2015.0230>.

defining and measuring information

guess the number am I thinking of!

what is the most efficient approach?



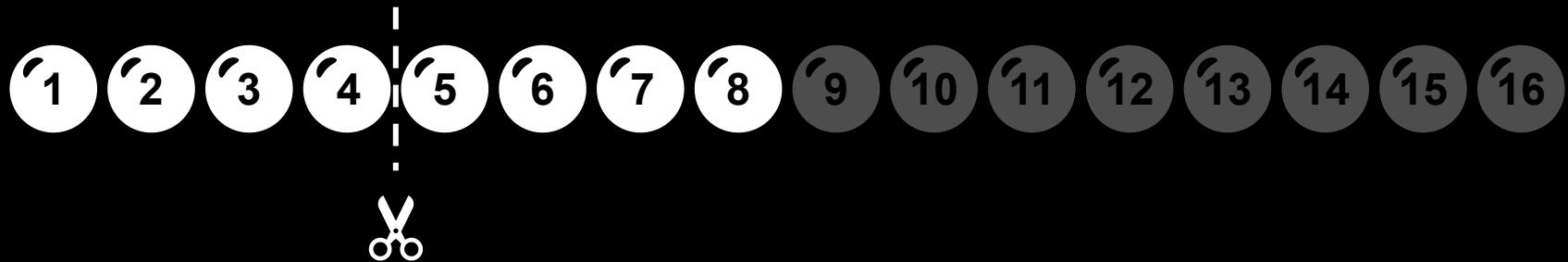
is it > 8 ?



is it $> 8?$ X



is it $> 8?$ X
is it $> 4?$



is it > 8 ? X

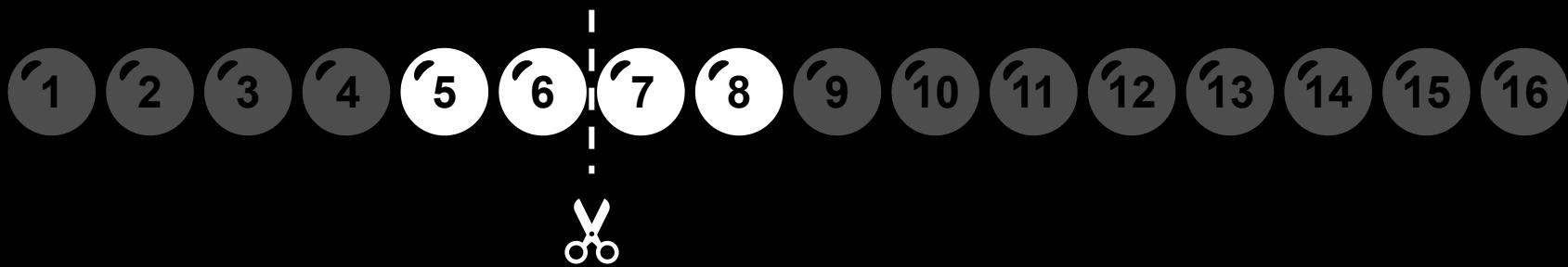
is it > 4 ? ✓



is it > 8? ✗

is it > 4? ✓

is it > 6?



is it > 8? ✗

is it > 4? ✓

is it > 6? ✓

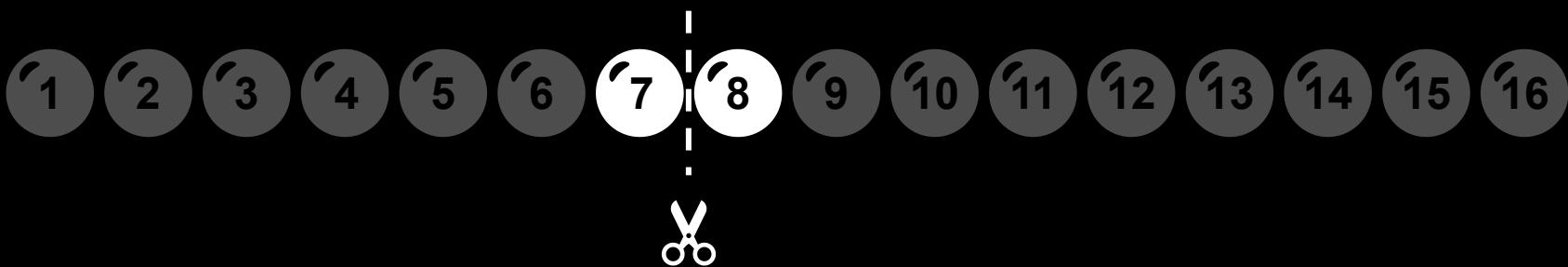


is it > 8? X

is it > 4? ✓

is it > 6? ✓

is it > 7?



is it > 8? ✗

is it > 4? ✓

is it > 6? ✓

is it > 7? ✗



is it > 8? ✗

is it > 4? ✓

is it > 6? ✓

is it > 7? ✗



with 4 questions from 16 to 1

| | | |
|------------|---|---|
| is it > 8? | ✗ | 0 |
| is it > 4? | ✓ | 1 |
| is it > 6? | ✓ | 1 |
| is it > 7? | ✗ | 0 |



with 4 questions from 16 to 1

where is the information?

where is the information?



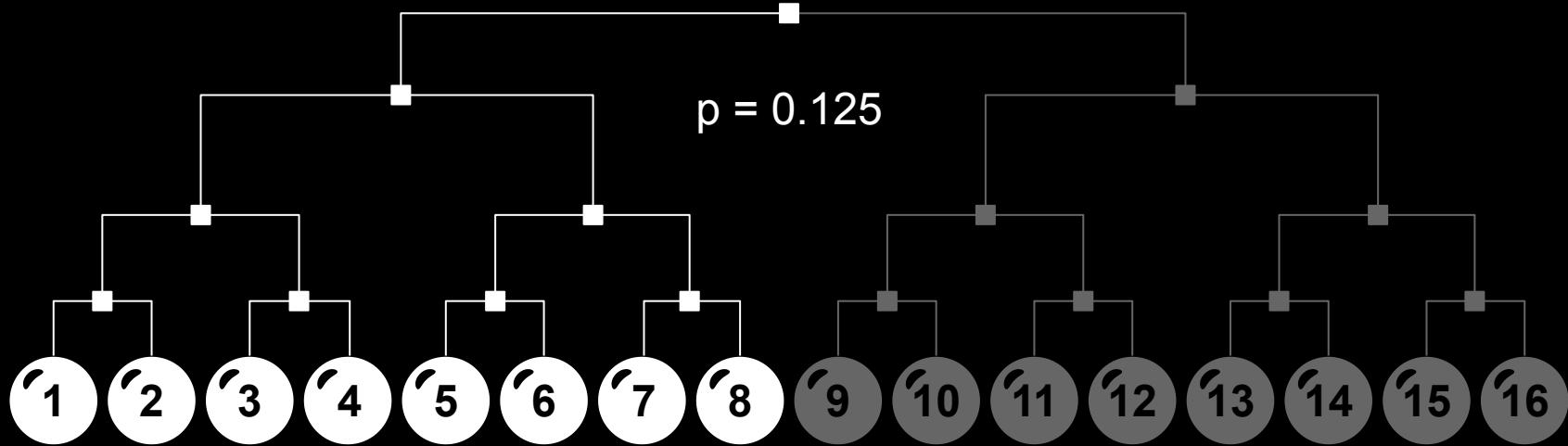
where is the information?

$N = 16$



where is the information?

$$\begin{aligned} N &= 16 \\ N &= 8 \end{aligned}$$



where is the information?

$\overline{N} = 16$

$\overline{N} = 8$

$N = 4$



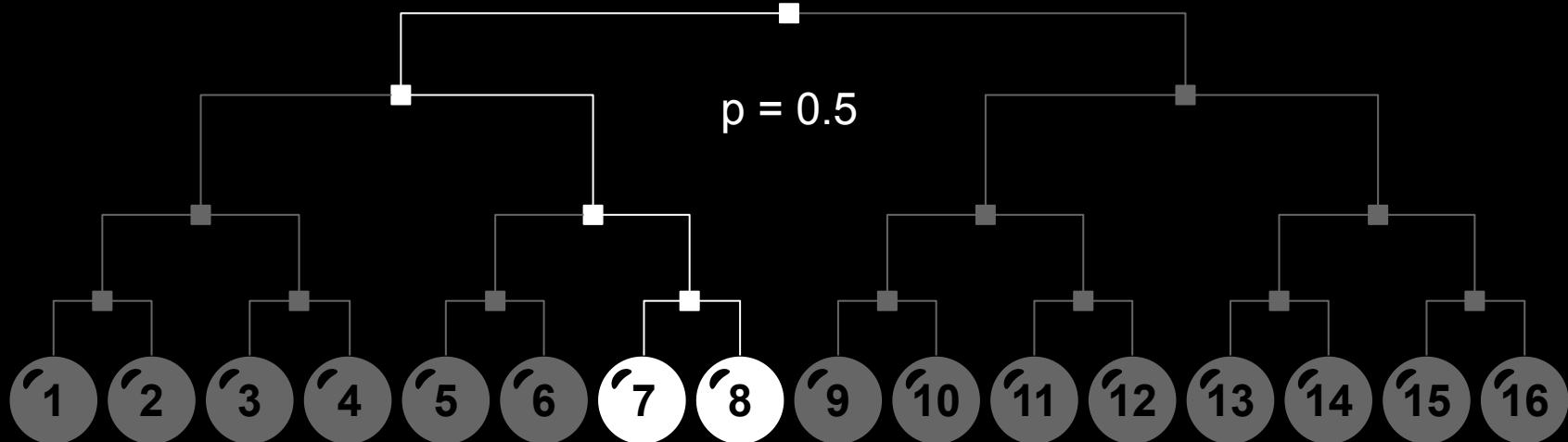
where is the information?

$N = 16$

$N = 8$

$N = 4$

$N = 2$



where is the information?

$N = 16$

$N = 8$

$N = 4$

$N = 2$

$N = 1$



information = reduced uncertainty
uncertainty is measured with the logarithm of N

$$H = \log_2(N)$$



or: how often can we cut the remaining possibilities in half?

$$H = \log_2(N)$$



$$H_0 = \log_2(16) = 4$$



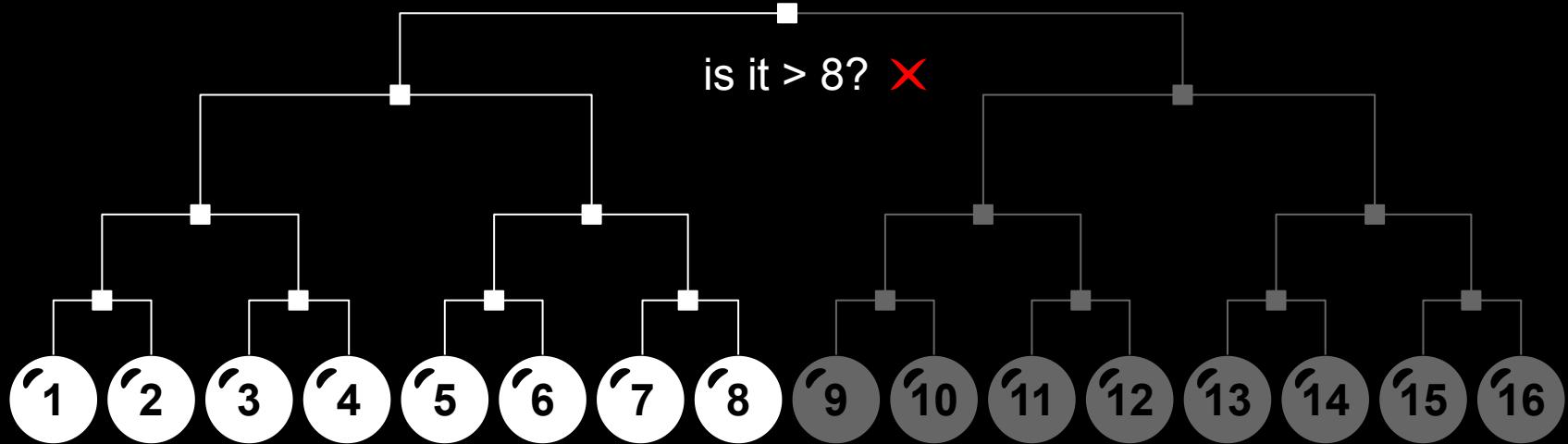
$$H_1 = \log_2(8) = 3$$



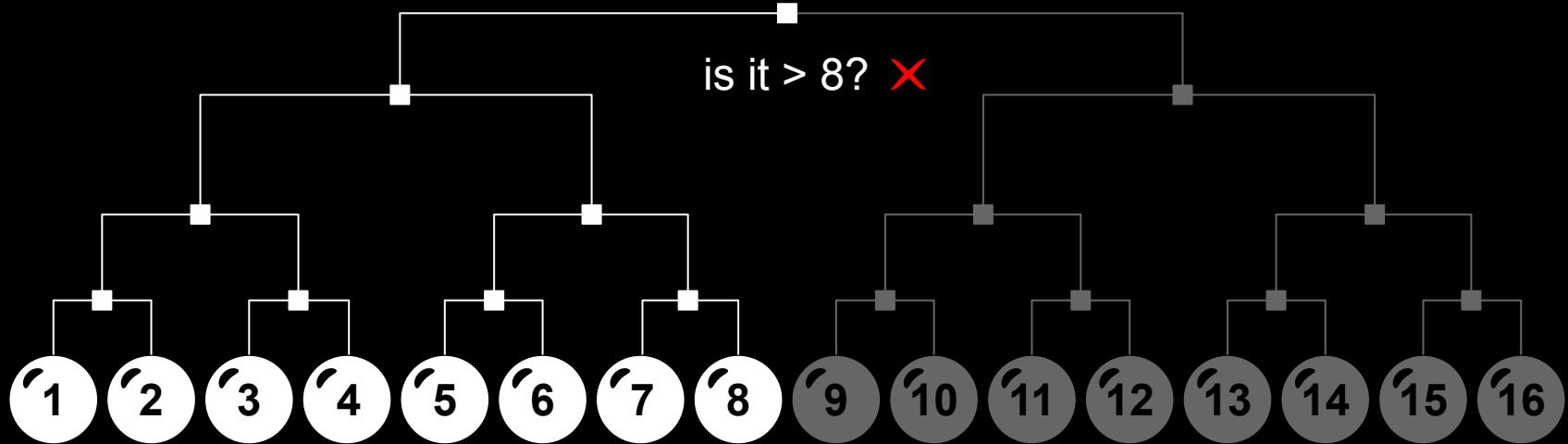
$$I = H_0 - H_1$$



$$I = \log_2(16) - \log_2(8)$$



$$I = 4 - 3 = 1$$



uncertainty and information are
measured in **bits**

how many yes/no questions to reduce uncertainty to zero?

$$H = 0 = \log_2(1)$$

how many yes/no questions to reduce uncertainty to zero?

$$H = 0 = \log_2(1)$$

$$H = \log_2(N)$$

poker

which card am I holding?



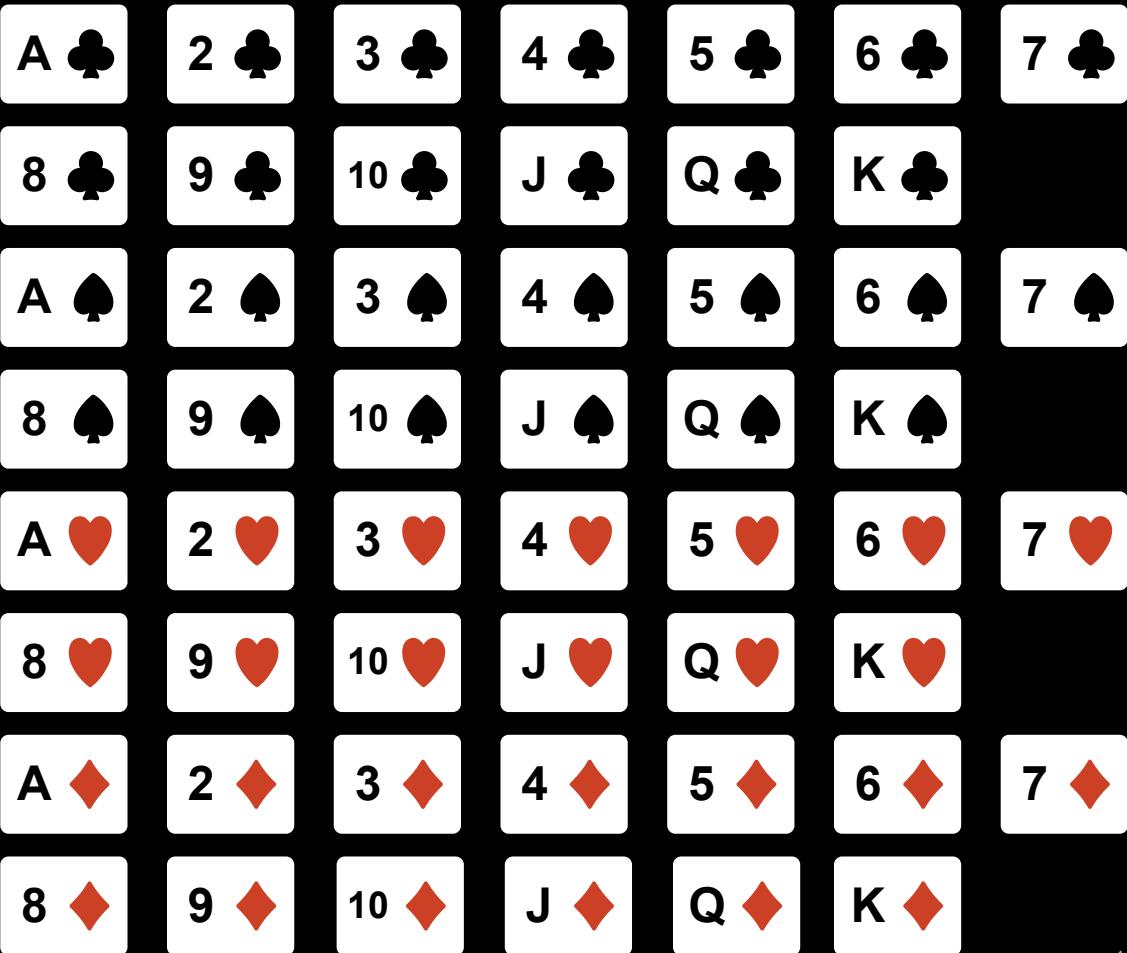
52 card poker deck



52 possible cards



is the card black?



is the card black?

no



is the card black?

no

is it hearts?

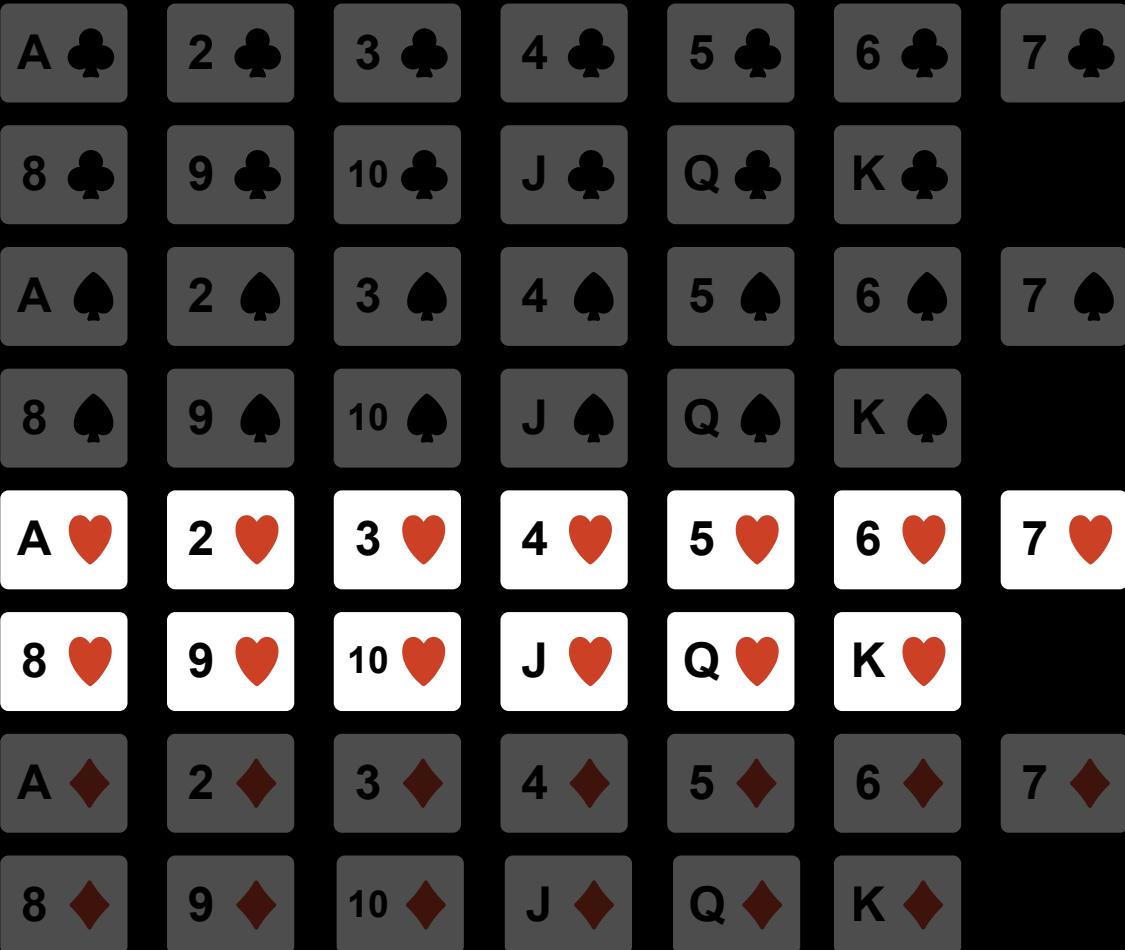


is the card black?

no

is it hearts?

yes



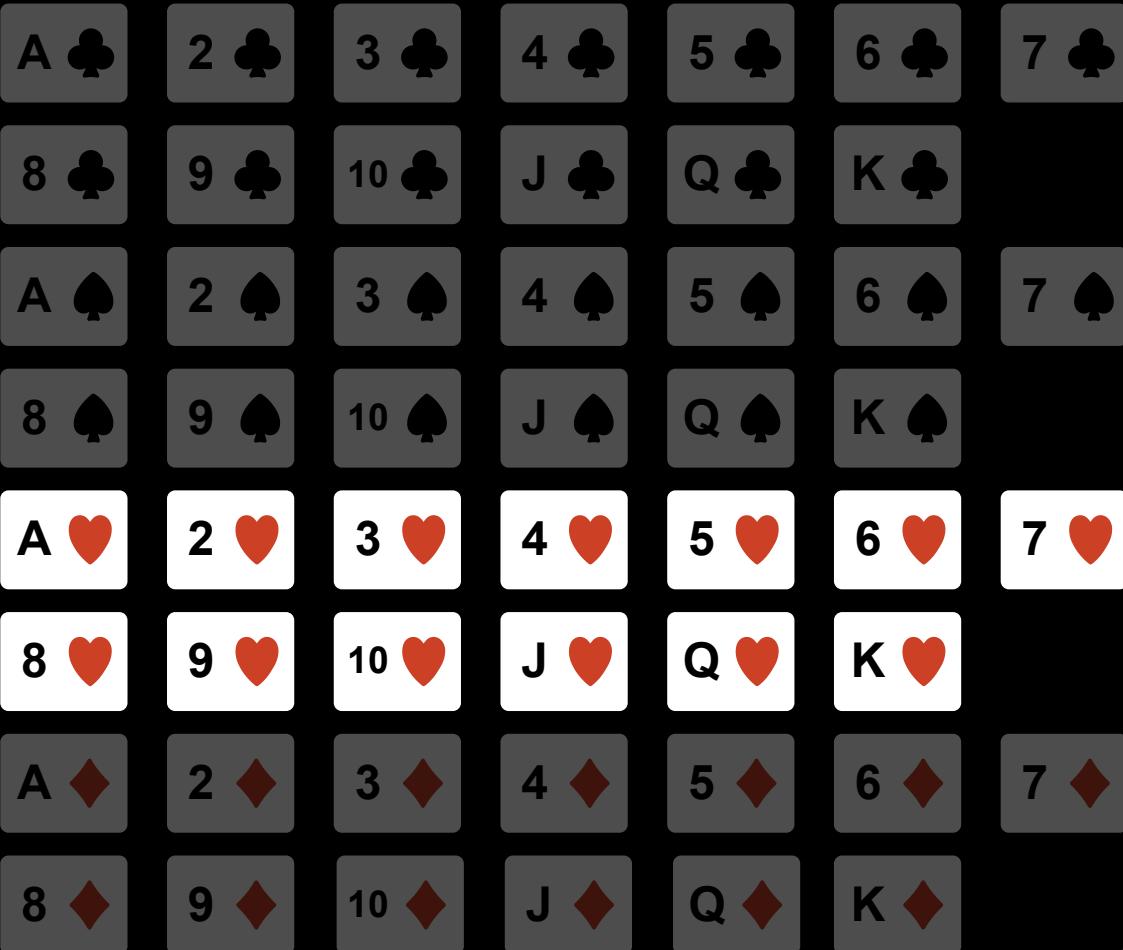
is the card black?

no

is it hearts?

yes

is it 8 or above?



is the card black?

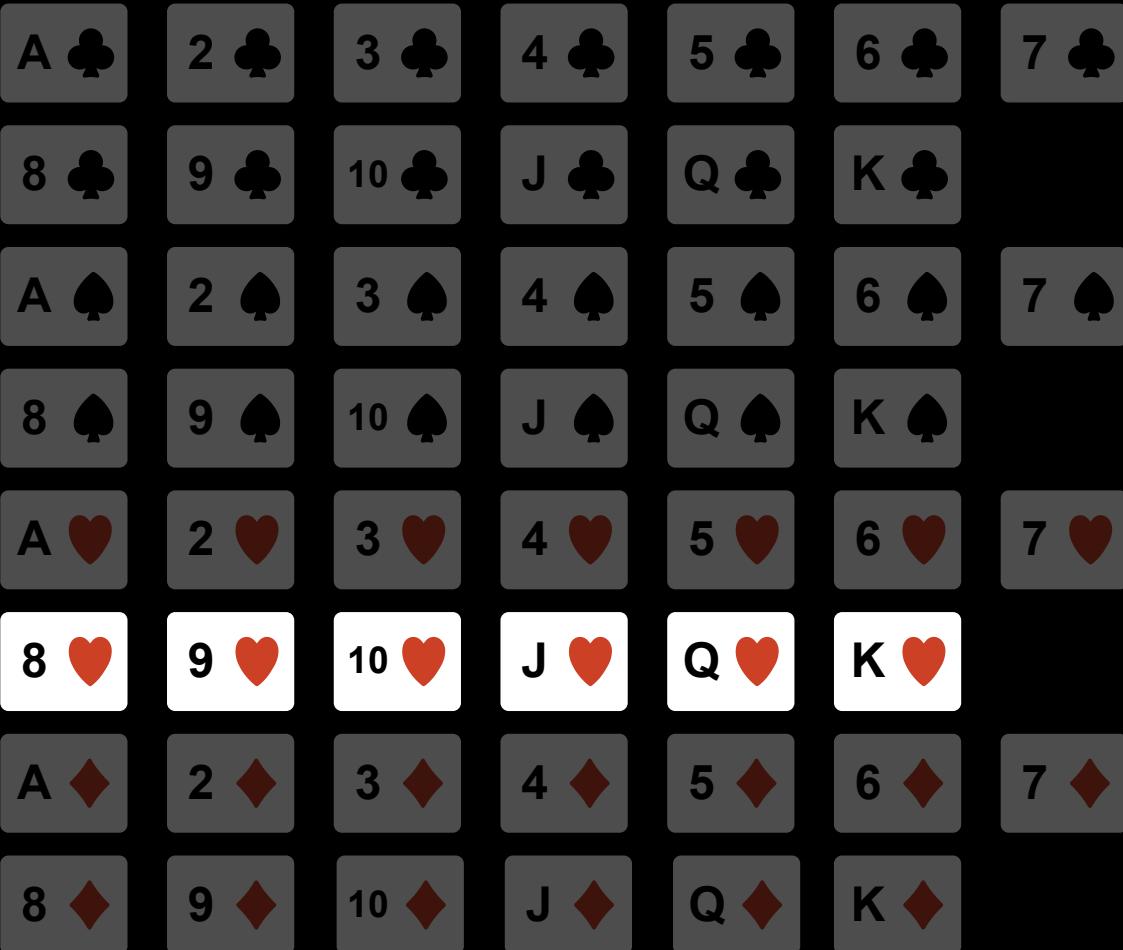
no

is it hearts?

yes

is it 8 or above?

yes



is the card black?

no

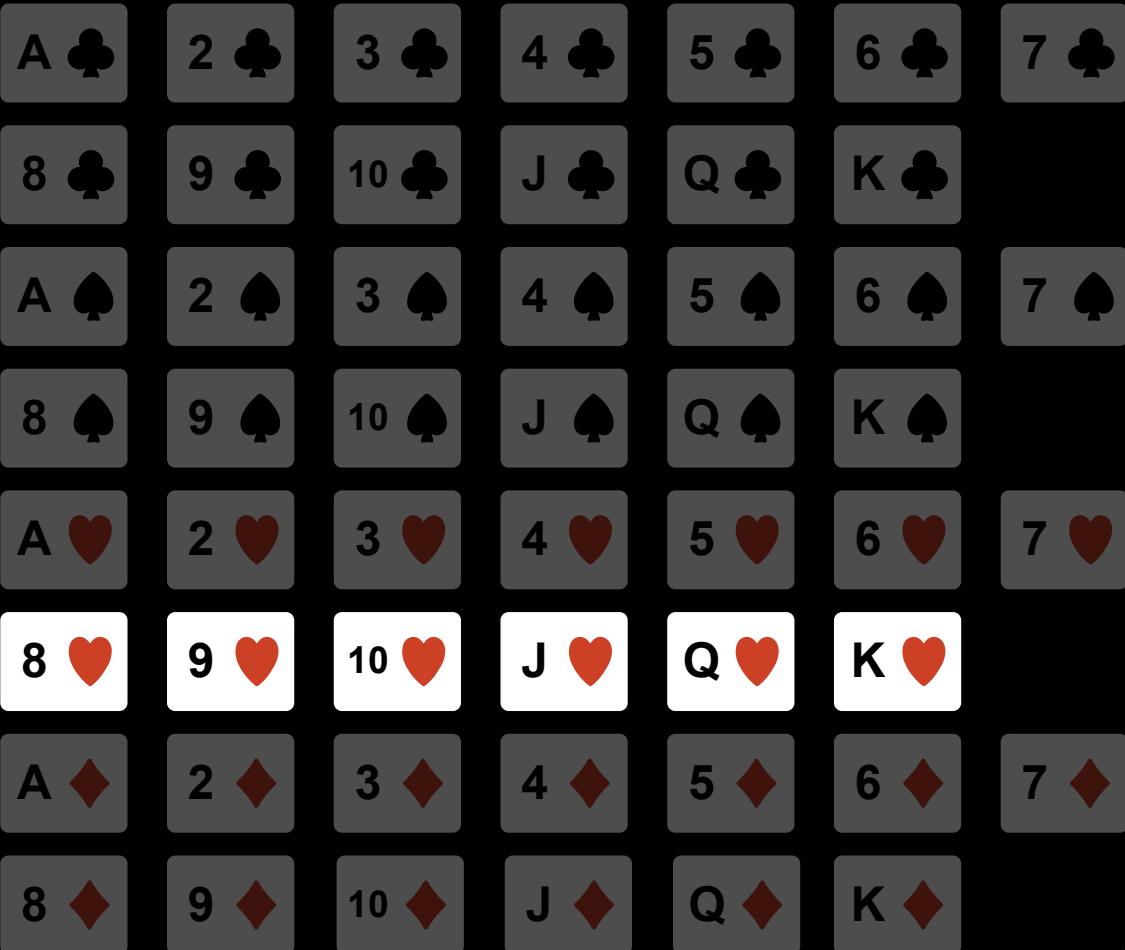
is it hearts?

yes

is it 8 or above?

yes

is it jack or above?



is the card black?

no

is it hearts?

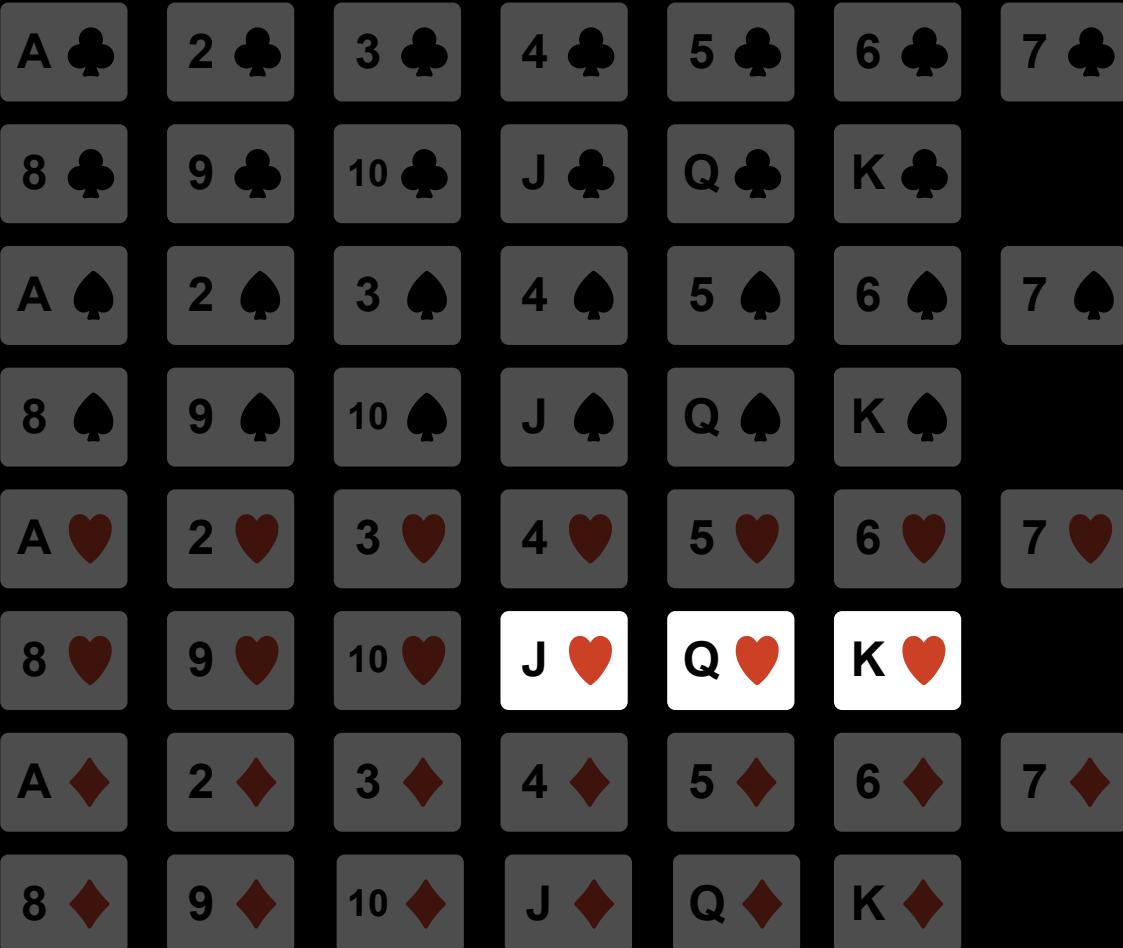
yes

is it 8 or above?

yes

is it jack or above?

yes



is the card black?

no

is it hearts?

yes

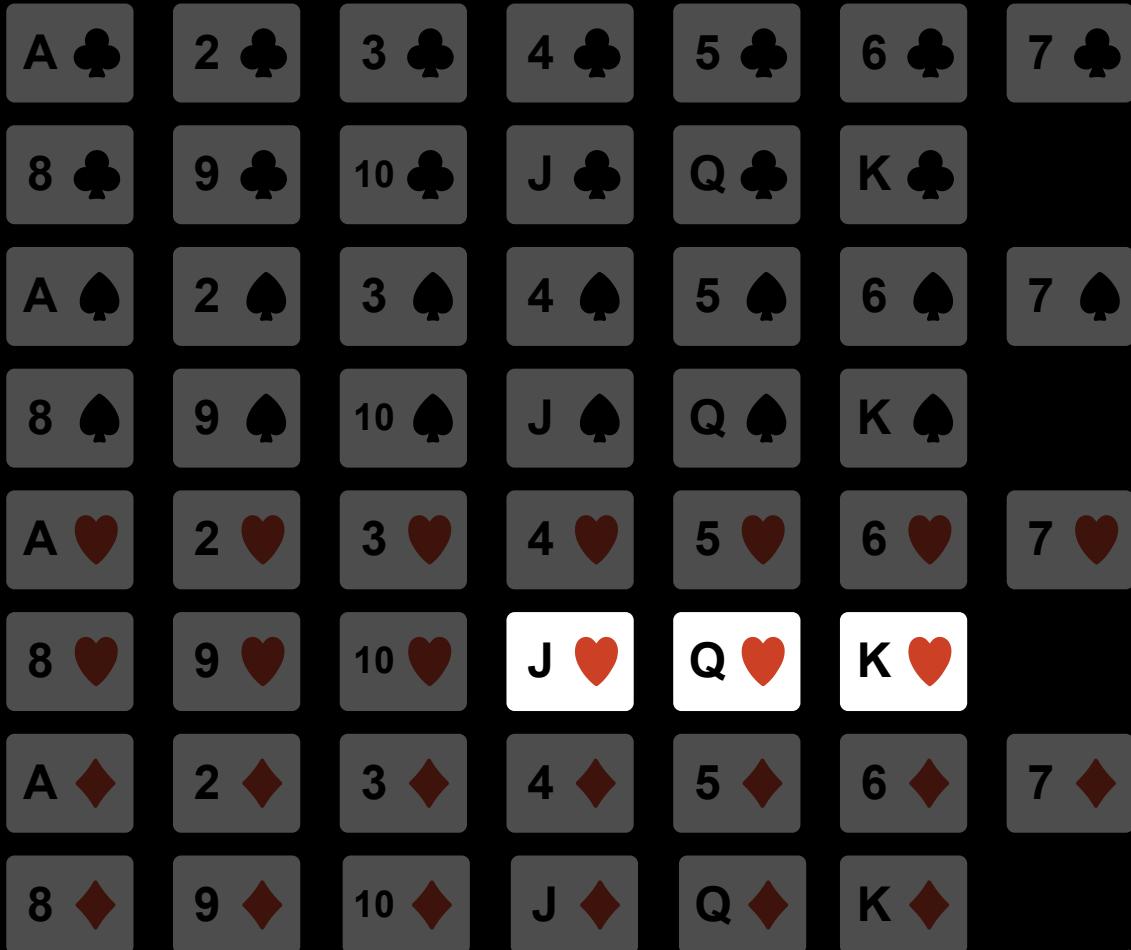
is it 8 or above?

yes

is it jack or above?

yes

is it queen or above?



is the card black?

no

is it hearts?

yes

is it 8 or above?

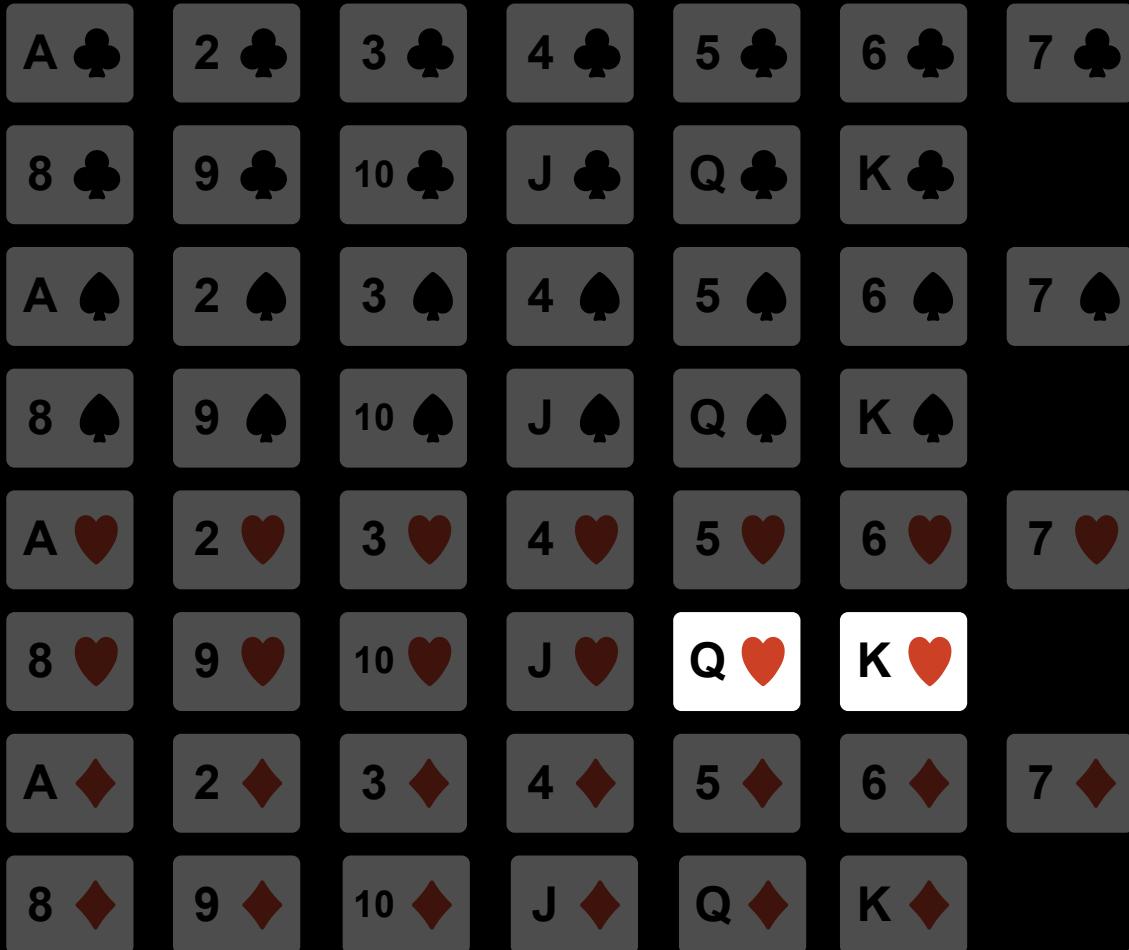
yes

is it jack or above?

yes

is it queen or above?

yes



is the card black?

no

is it hearts?

yes

is it 8 or above?

yes

is it jack or above?

yes

is it queen or above?

yes

is it king?



is the card black?

no

is it hearts?

yes

is it 8 or above?

yes

is it jack or above?

yes

is it queen or above?

yes

is it king?

no



with 6 questions
from 52 to 1

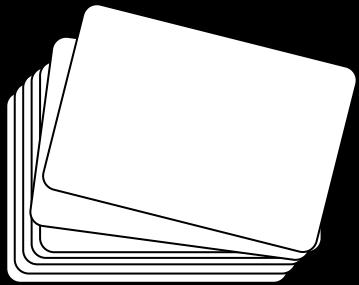


uncertainty with $N = 52$ possibilities?



$$H = \log_2(52) \approx 5.7$$

uncertainty with $N = 52$ possibilities?



average # of yes/no
questions



$$H = \log_2(52) \approx 5.7$$

one bit of information with each answer...

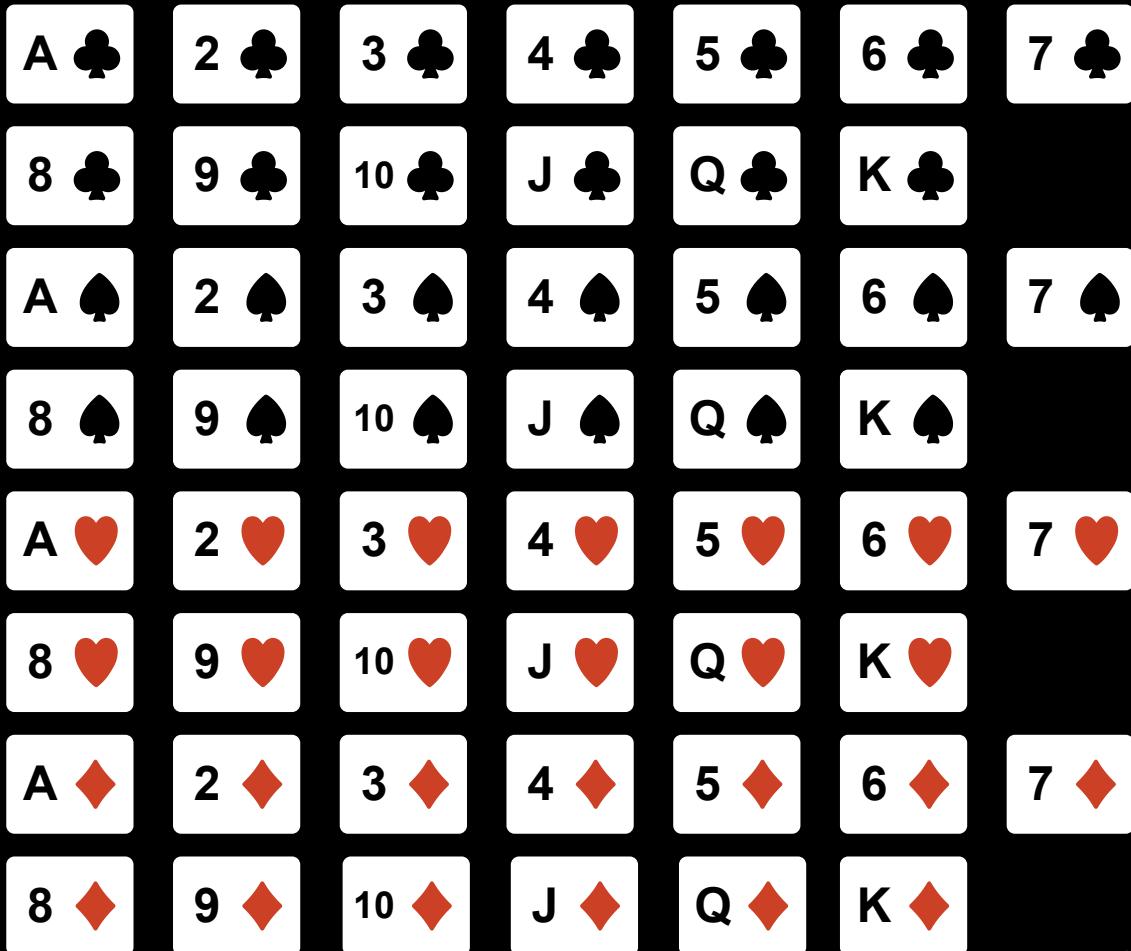
$$\log_2(52) - \log_2(26) = 1$$

one bit of information with each answer...

$$\log_2(52) - \log_2(26) = 1$$

...that cuts the remaining options in half

is it a spades card?



is it a spades card?

no



is it a spades card?

no

how much information?



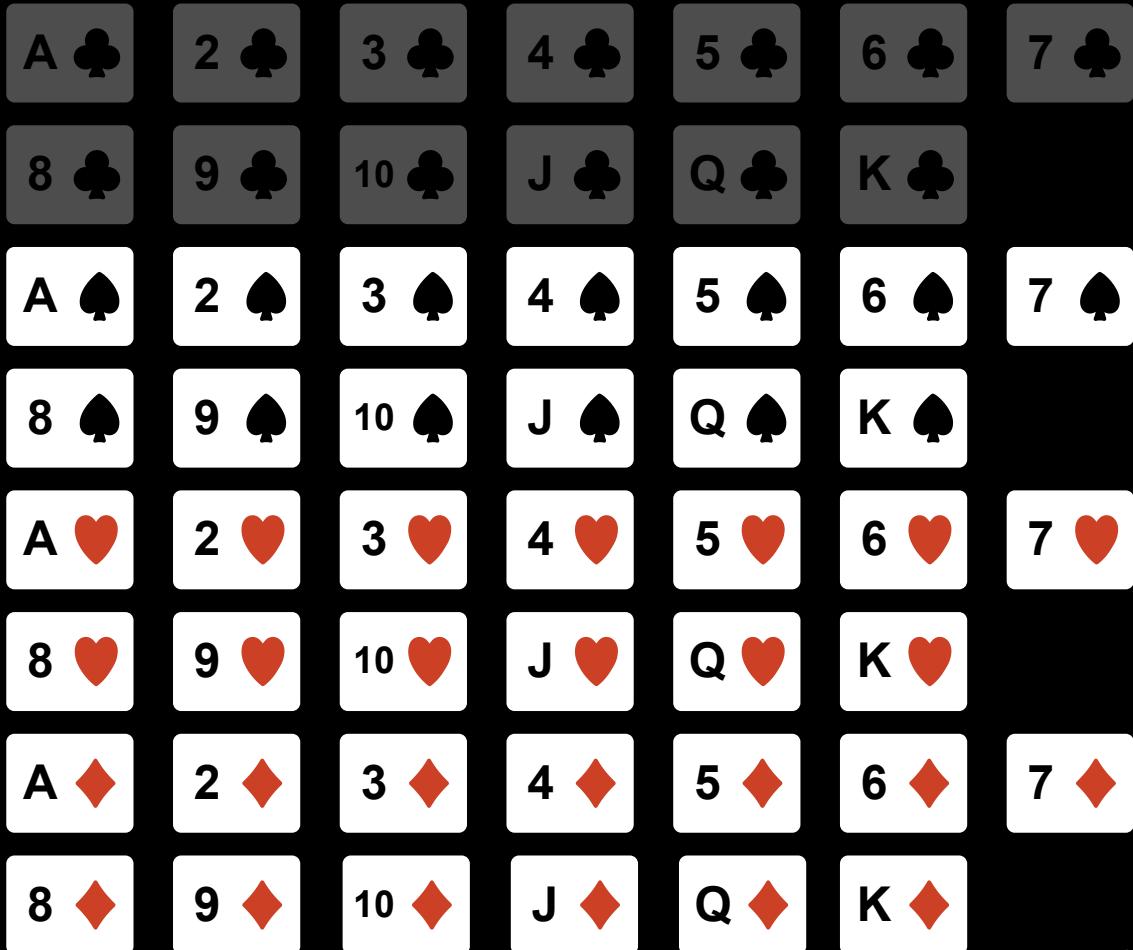
is it a spades card?

no

how much information?

$$H_0 = \log_2(52) \approx 5.7$$

$$H_1 = \log_2(39) \approx 5.29$$



is it a spades card?

no

how much information?

$$H_0 = \log_2(52) \approx 5.7$$

$$H_1 = \log_2(39) \approx 5.29$$

$$H_0 - H_1 \approx 0.41$$



is it a spades card?

no

how much information?

$$H_0 = \log_2(52) \approx 5.7$$

$$H_1 = \log_2(39) \approx 5.29$$

$$H_0 - H_1 \approx 0.41$$

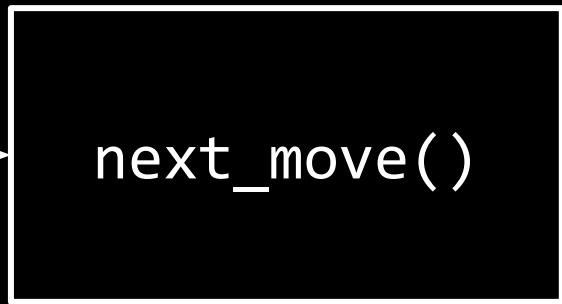
that's less than 1 bit



chess



how much information is
one move?









c2 → c4



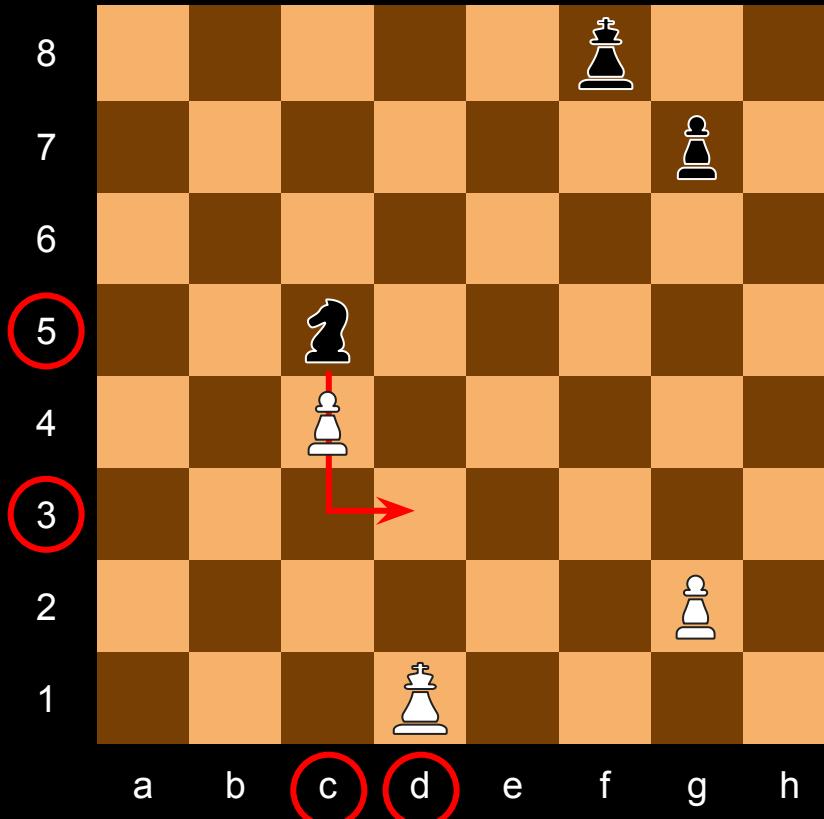
c2 → c4



c2 → c4



$c2 \rightarrow c4$
 $c5 \rightarrow d3$



$c2 \rightarrow c4$
 $c5 \rightarrow d3$
...



c2 → c4

how many possibilities?

64 fields x 64 fields

how many possibilities?

$$64 \times 64 = 4096$$

possible moves*

*disregarding impossible moves

$$64 \times 64 = 4096$$

$$H = \log_2(4096) = 12$$

$$64 \times 64 = 4096$$

$$H = \log_2(4096) = 12$$

one chess move is 12 bits of information

an alternative way to calculate # bits

c 2 c 4

4 digits

c 2 c 4

4 digits

8 possible symbols per digit

c 2 c 4

4 digits

8 possible symbols per digit
how many bits per digit?

c 2 c 4

4 digits

8 possible symbols per digit
how many bits per digit?

$$H_{digit} = \log_2(8) = 3$$

c 2 c 4

4 digits

8 possible symbols per digit
how many bits per digit?

$$H_{digit} = \log_2(8) = 3$$

$$H_{move} = \log_2(8) \times 4 = 12$$

$$H_{avg} = \log_2(S) \times n$$

S: number of possible symbols

n: number of digits in our message

$$H_{max} = \lceil \log_2(S) \rceil \times n$$

when calculating bits for storage, we must
always consider the worst case

digits and # symbols

{A}

— —

{A}

A A

{A, 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 # digits n = 2

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

with length $n = 2$

| # symbols | $f(x)$ | # messages |
|-----------|--------|------------|
| 1 | | 1 |
| 2 | $f(x)$ | 4 |
| 3 | → | 9 |
| 4 | | 16 |
| 5 | | 25 |

and more digits?

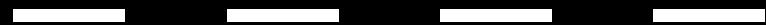
{A, B}

— — —

{A, B}

AAA, AAB, ABA, ABB,
BBB, BBA, BAA, BAB

{A, B}



{A, B}

AAAA, AAAB, AABA, AABB,
ABAA, ABAB, ABBA, ABBB,
BAAA, BAAB, BABA, BABB,
BBAA, BBAB, BBBA, BBBB

with # symbols $S = 2$

| # digits | # messages |
|----------|------------|
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| 4 | 16 |
| 5 | 32 |

with # symbols $S = 2$

| # digits | $f(x)$ | # messages |
|----------|--------|------------|
| 1 | | 2 |
| 2 | $f(x)$ | 4 |
| 3 | → | 8 |
| 4 | | 16 |
| 5 | | 32 |

with # symbols $S = 2$

| # digits | $f(x)$ | # messages |
|----------|--------|------------|
| 1 | | 2 |
| 2 | $f(x)$ | 4 |
| 3 | → | 8 |
| 4 | | 16 |
| 5 | | 32 |

possible messages with n digits and S symbols

$$N = S^n$$

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

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



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

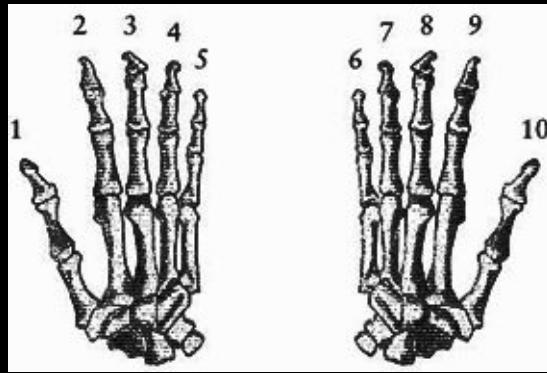
$$= 4 \times 1000 + 1 \times 100 + 2 \times 10 + 3 \times 1$$



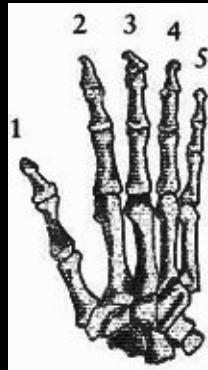
$$= 4 \times 10^3 + 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$$

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

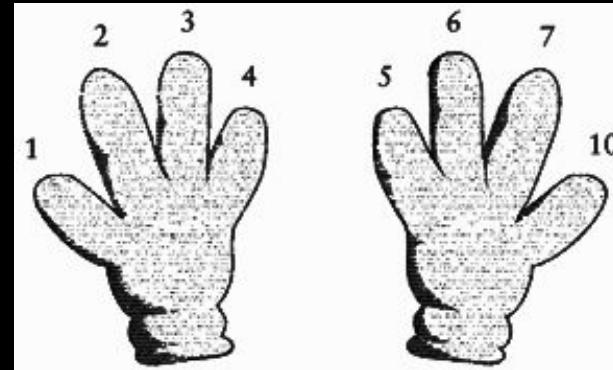
$$= 4123$$



human hand



human hand



cartoon character's hand

1

2

3

(octal)

1

2

3

(octal)

8^2

8^1

8^0

1

2

3

(octal)

8^2

8^1

8^0

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

1

2

3

(octal)

8^2

8^1

8^0

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

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

1

2

3

(octal)

8^2

8^1

8^0

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

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

$$= 83 \text{ (decimal)}$$

decimal

octal

8



?

decimal

octal

?



7

decimal

16

octal

?



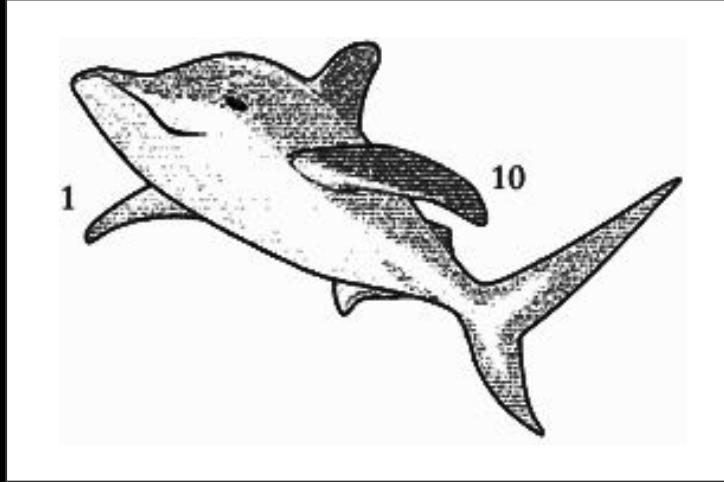
decimal

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

$$\begin{array}{r} 1 & 1 & 0 \\ \hline 2^2 & 2^1 & 2^0 \end{array}$$

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

$$\begin{array}{r} 1 & 1 & 0 \\ \hline 2^2 & 2^1 & 2^0 \end{array}$$

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

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

$$\begin{array}{r} 1 & 1 & 0 \\ \hline 2^2 & 2^1 & 2^0 \end{array}$$

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

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

$$= 6 \text{ (decimal)}$$

2 3 4 5 6

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

place value systems

$$N = d_n * R^{n-1} + \dots + d_2 * R^1 + d_1 * R^0$$

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

n = number of digits

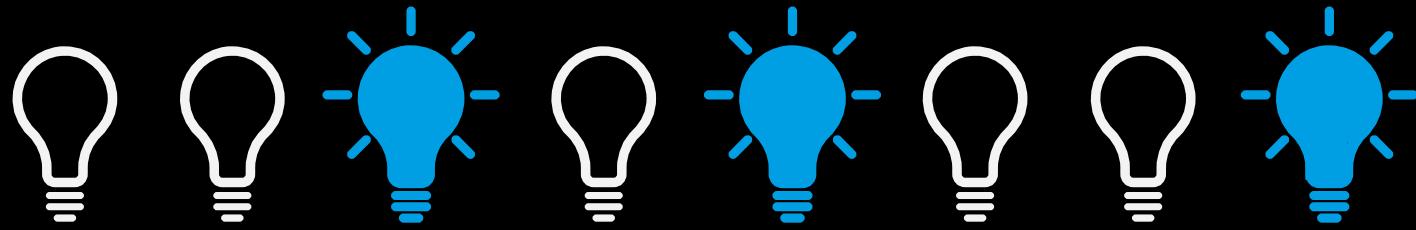
R = base

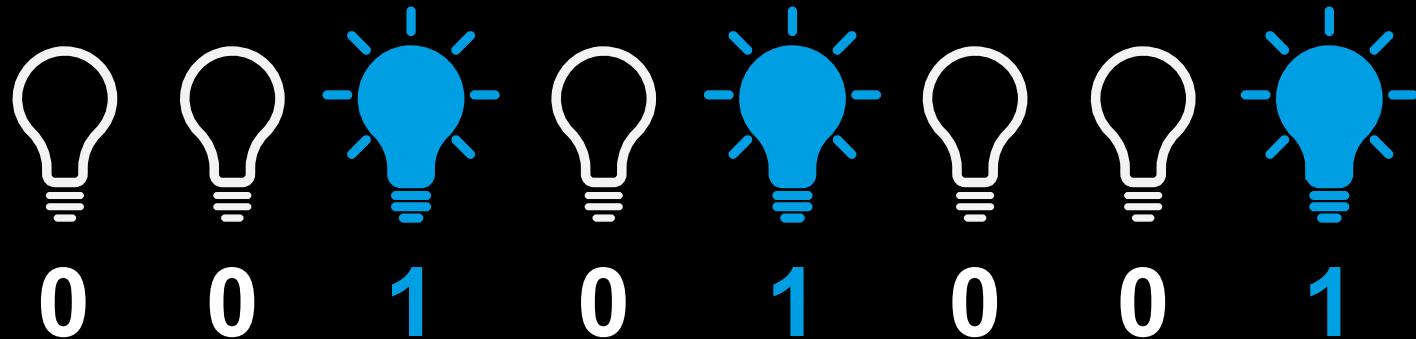
$R \geq 2$

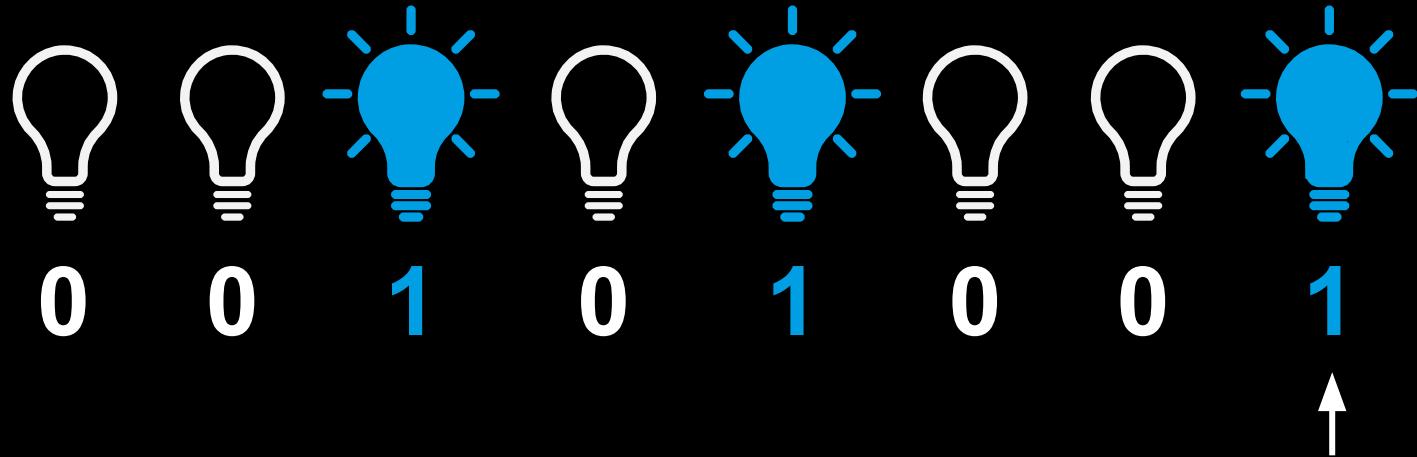
BITS

why do computers think binary?

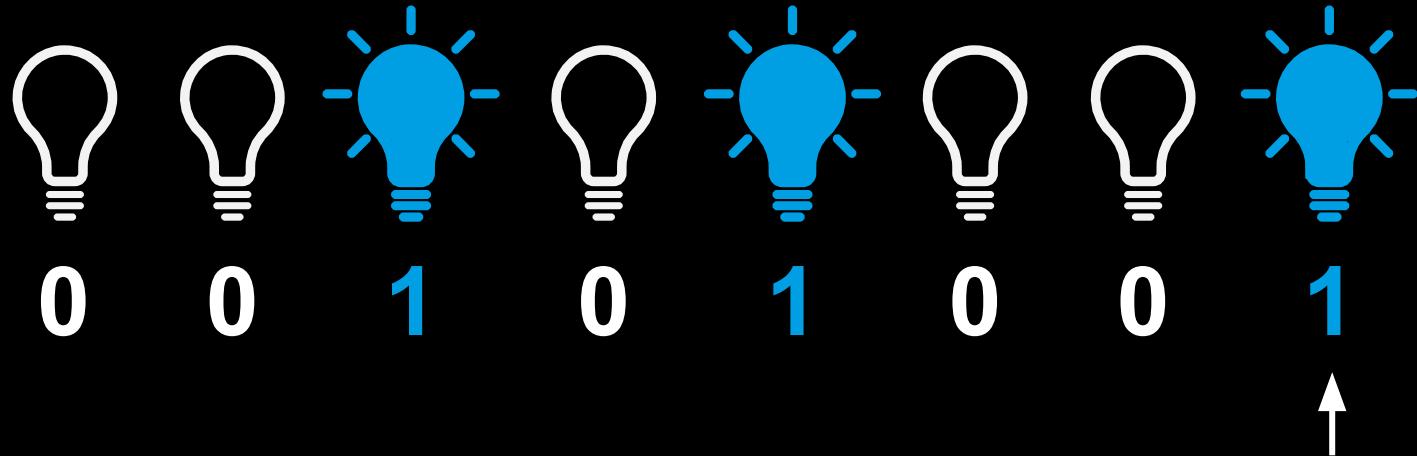








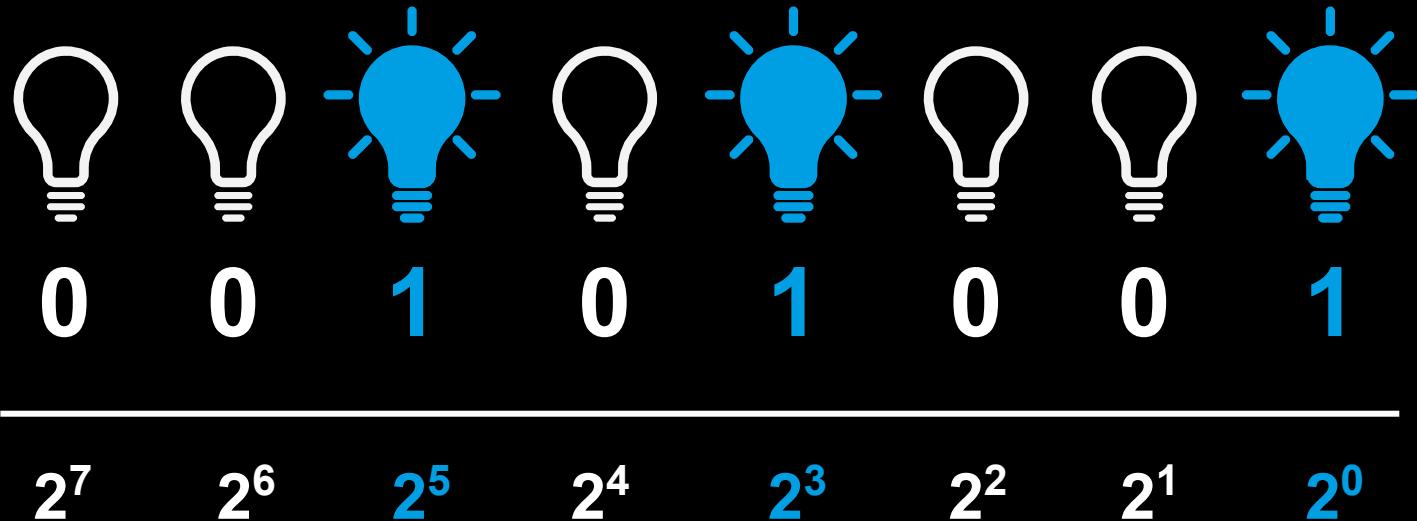
a **bit** (binary digit)

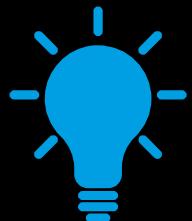
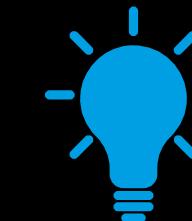
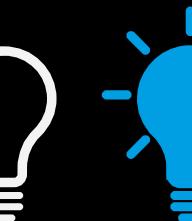


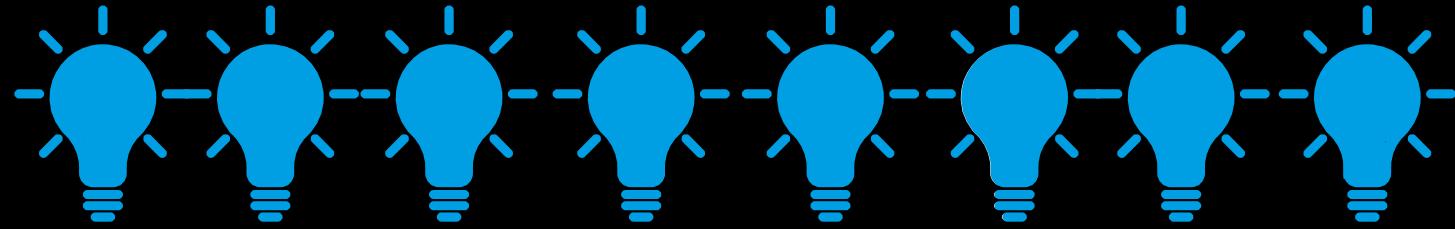
a **bit** (binary digit)

{

a **byte** (8 bits)



| | | | | | | | |
|---|---|---|---|--|---|---|---|
|  |  |  |  |  |  |  |  |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| <hr/> | | | | | | | |
| 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?

what comes after the byte?

2^{10} bytes = 1.024 bytes = 1 Kibibyte (KiB)

2^{20} bytes = 1.048.576 bytes = 1 Mebibyte (MiB)

2^{30} bytes = 1.073.741.824 bytes = 1 Gibibyte (GiB)

10^3 bytes = 1.000 bytes = 1 Kilobyte (KB)

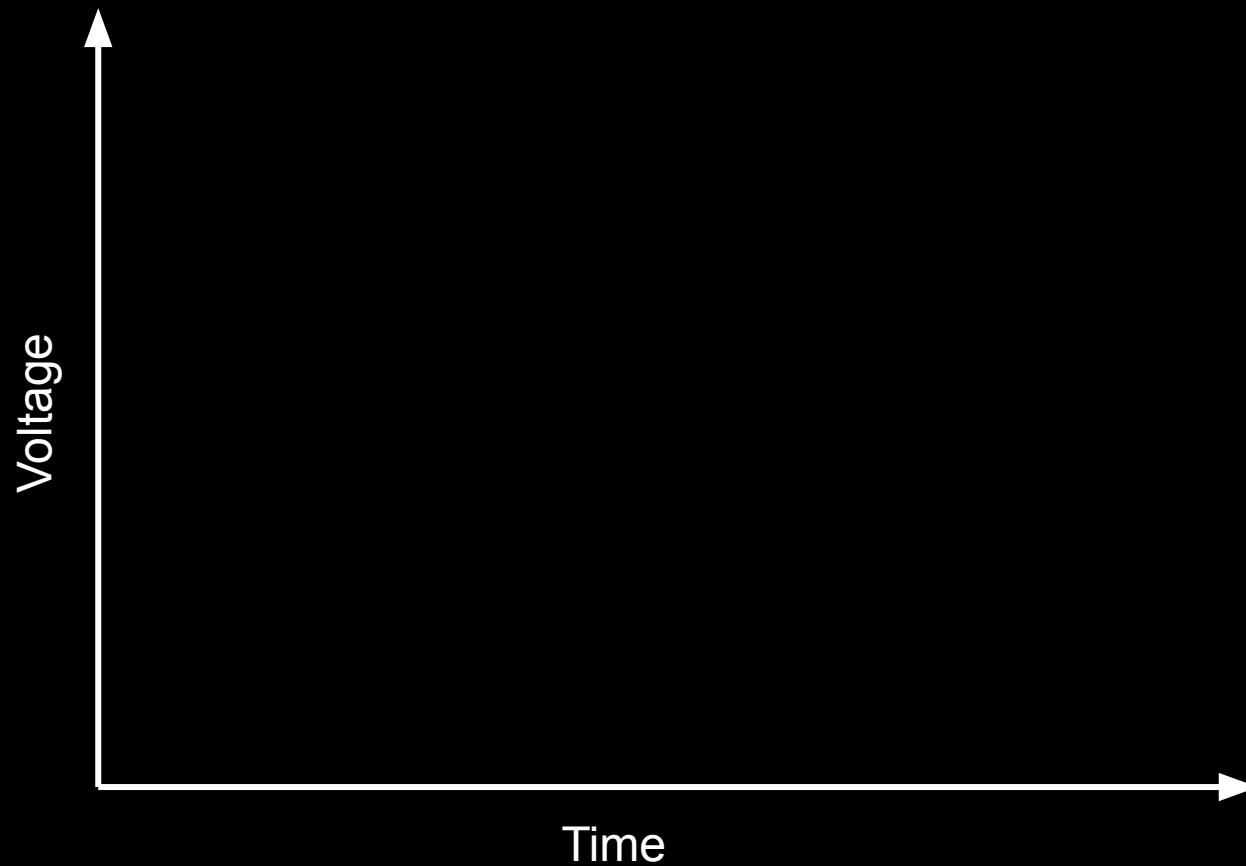
10^6 bytes = 1.000.000 bytes = 1 Megabyte (MB)

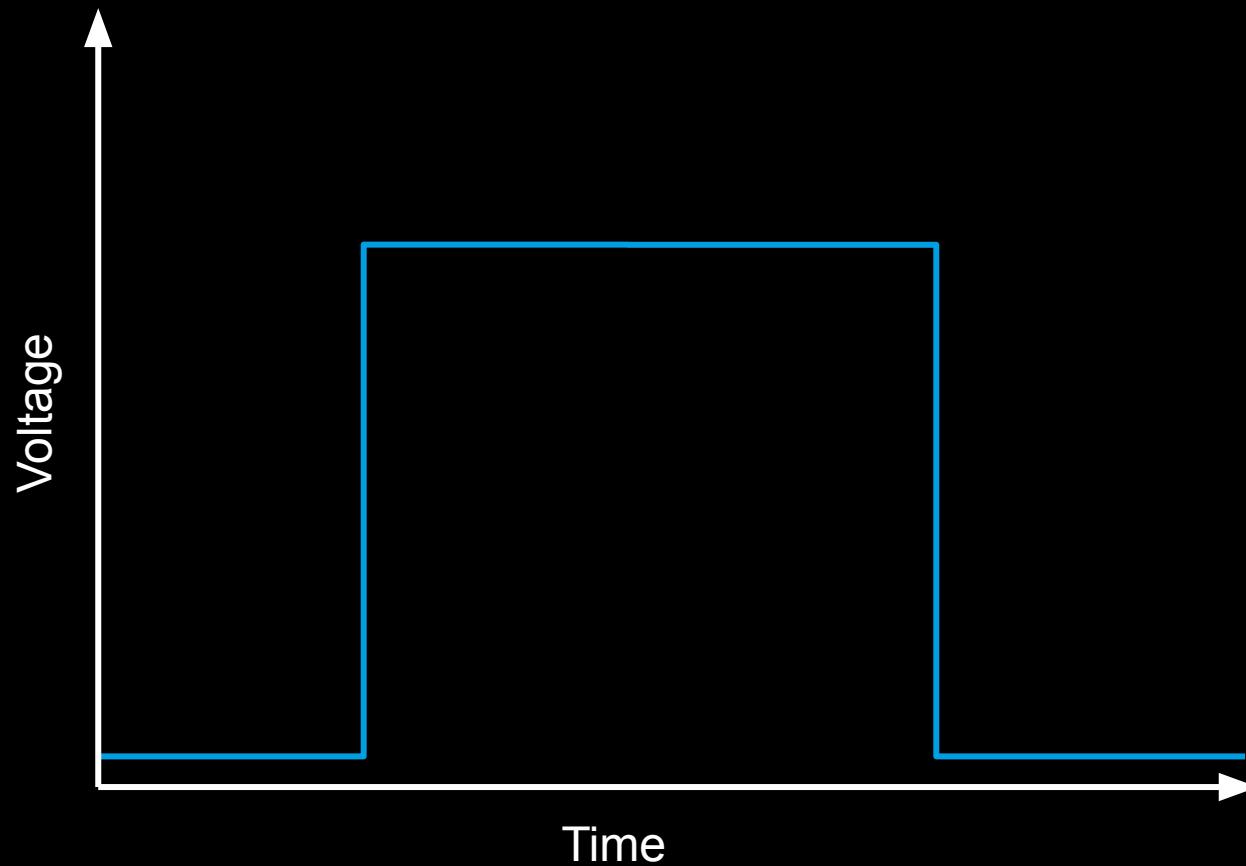
10^9 bytes = 1.000.000.000 bytes = 1 Gigabyte (GB)

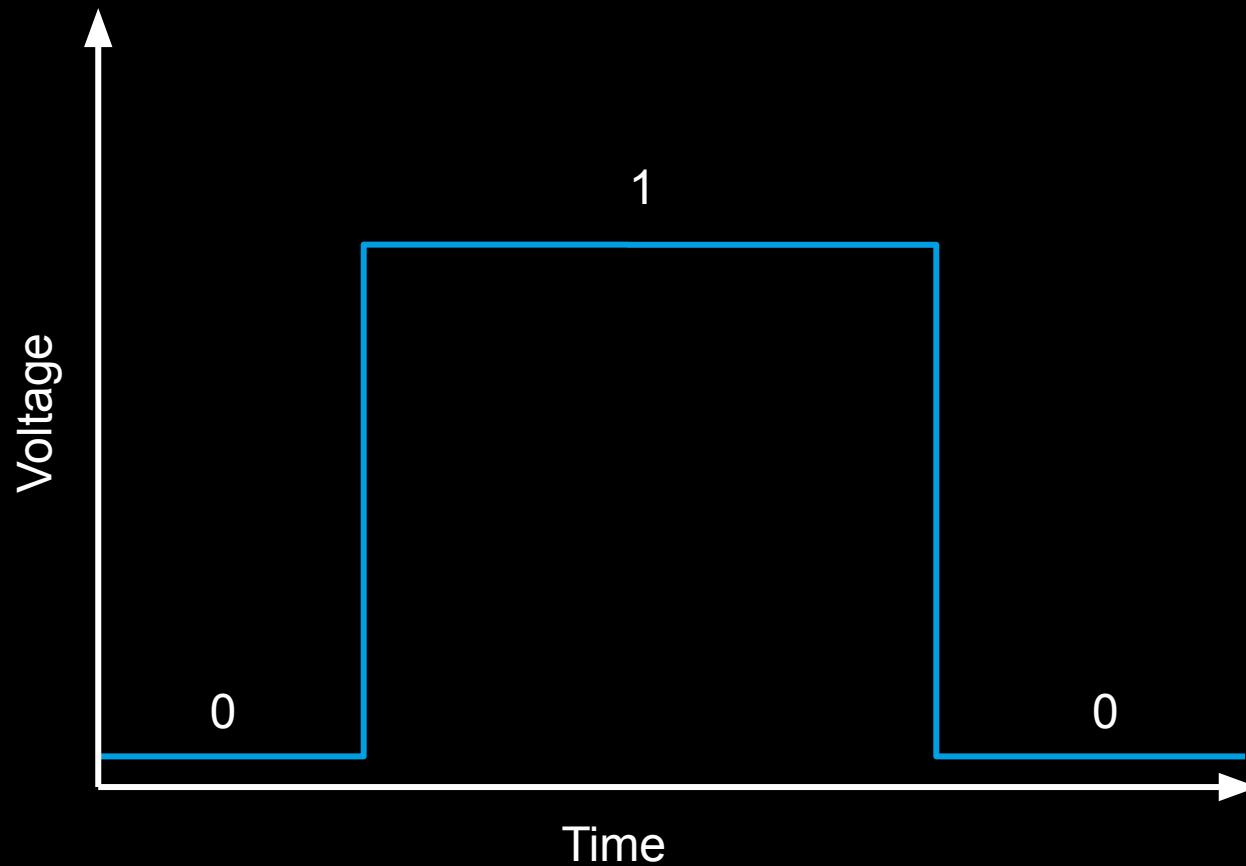
10^{12} bytes = ?

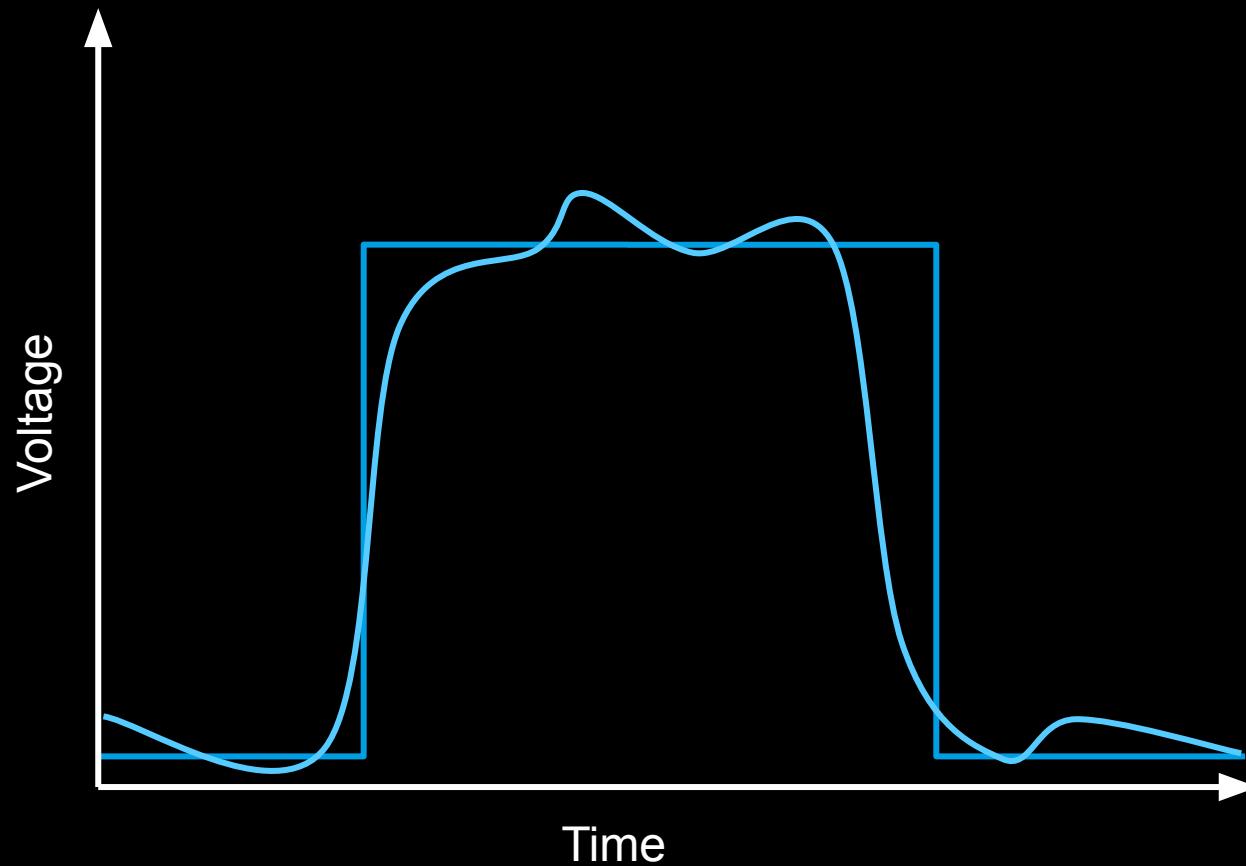
how many bits are on a DVD with
4.7 GB capacity?

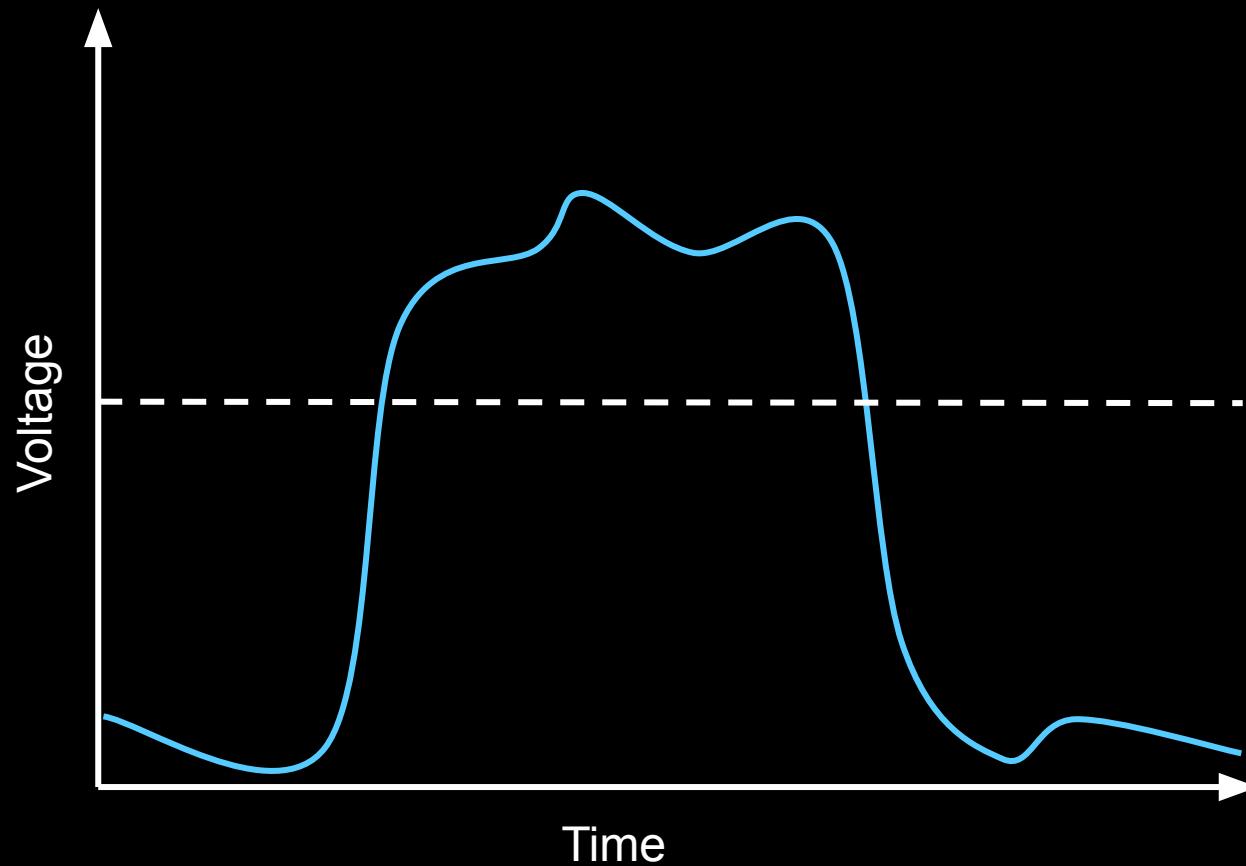
are we stuck with binary?

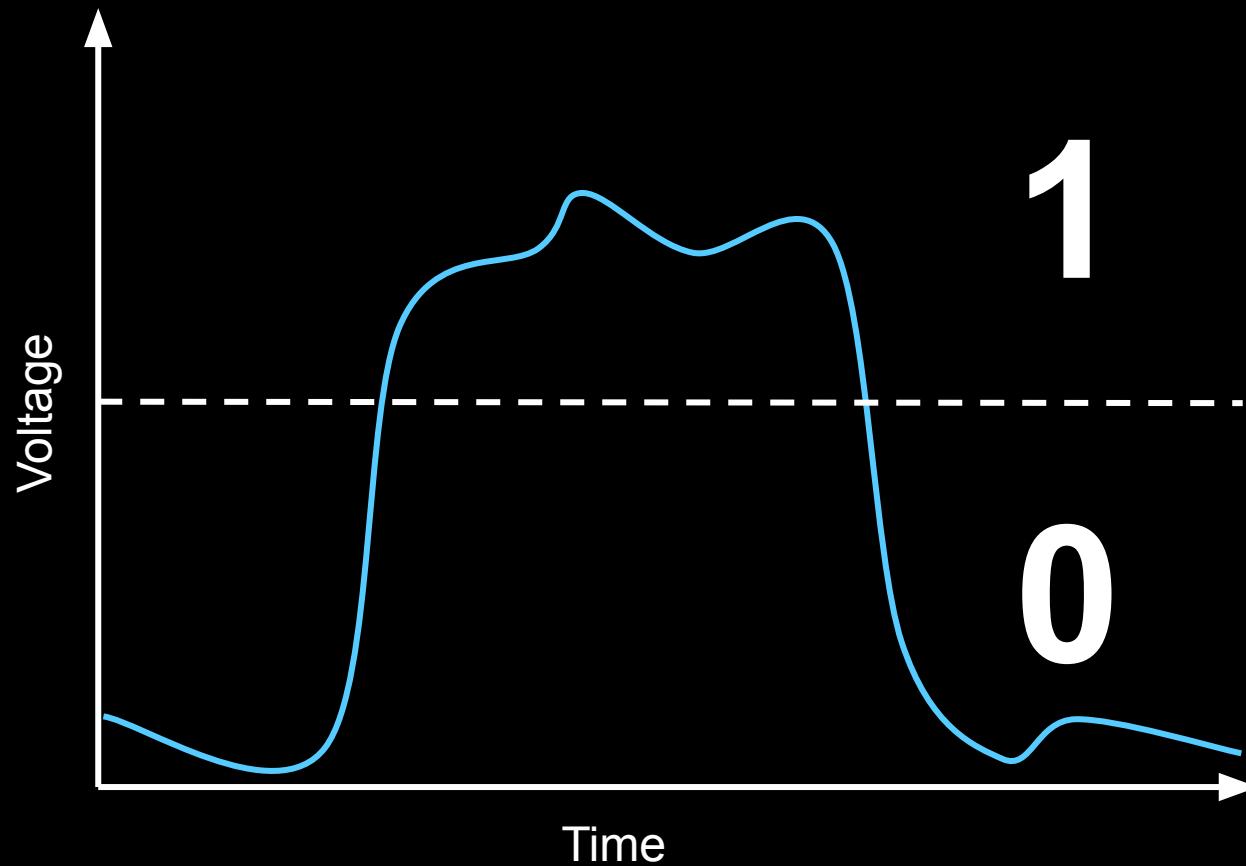


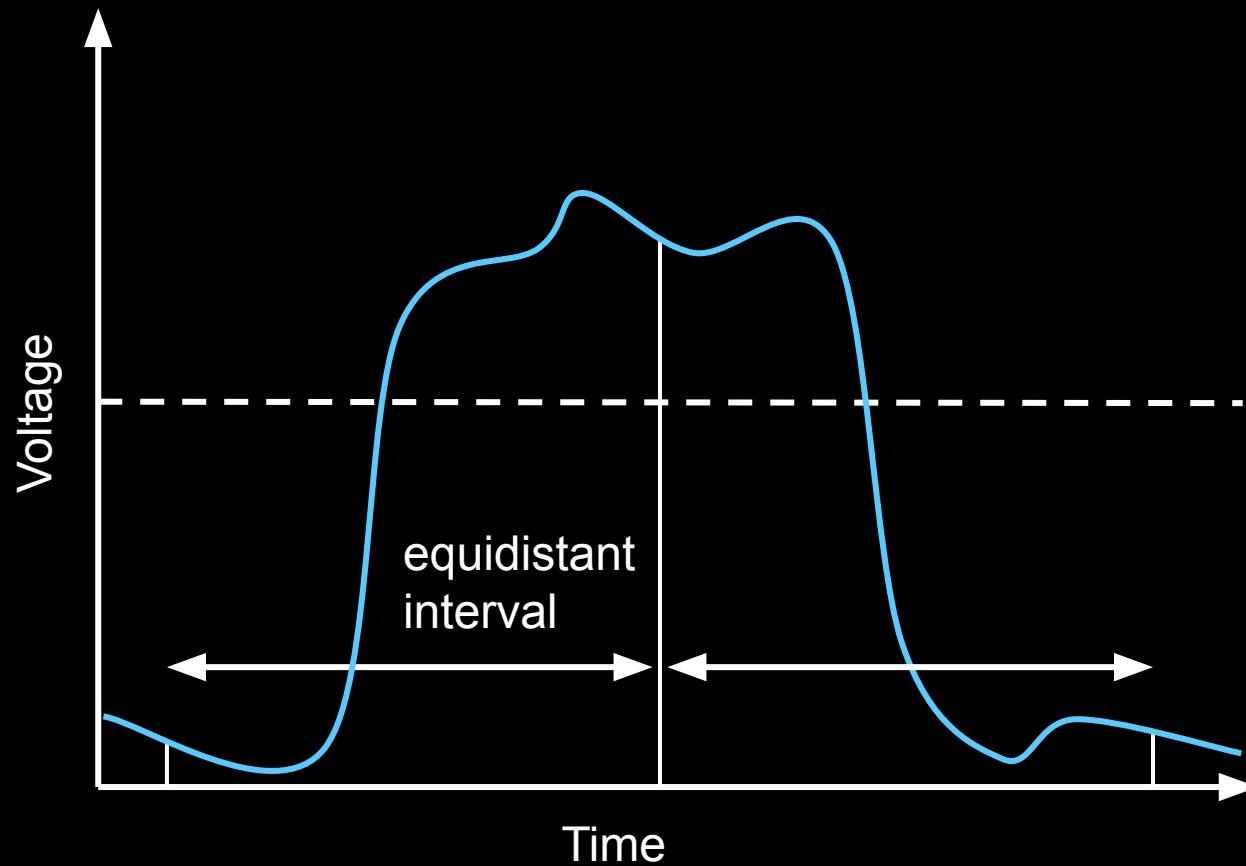


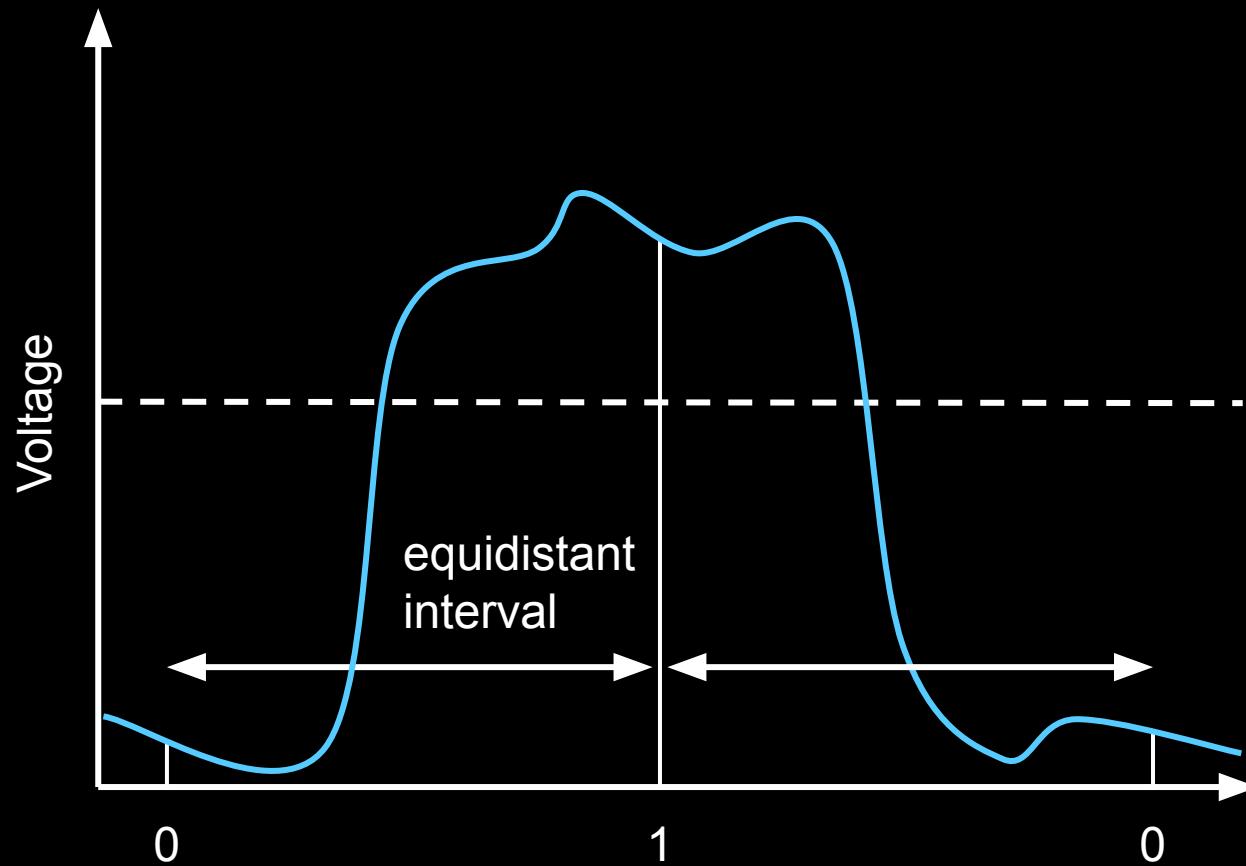


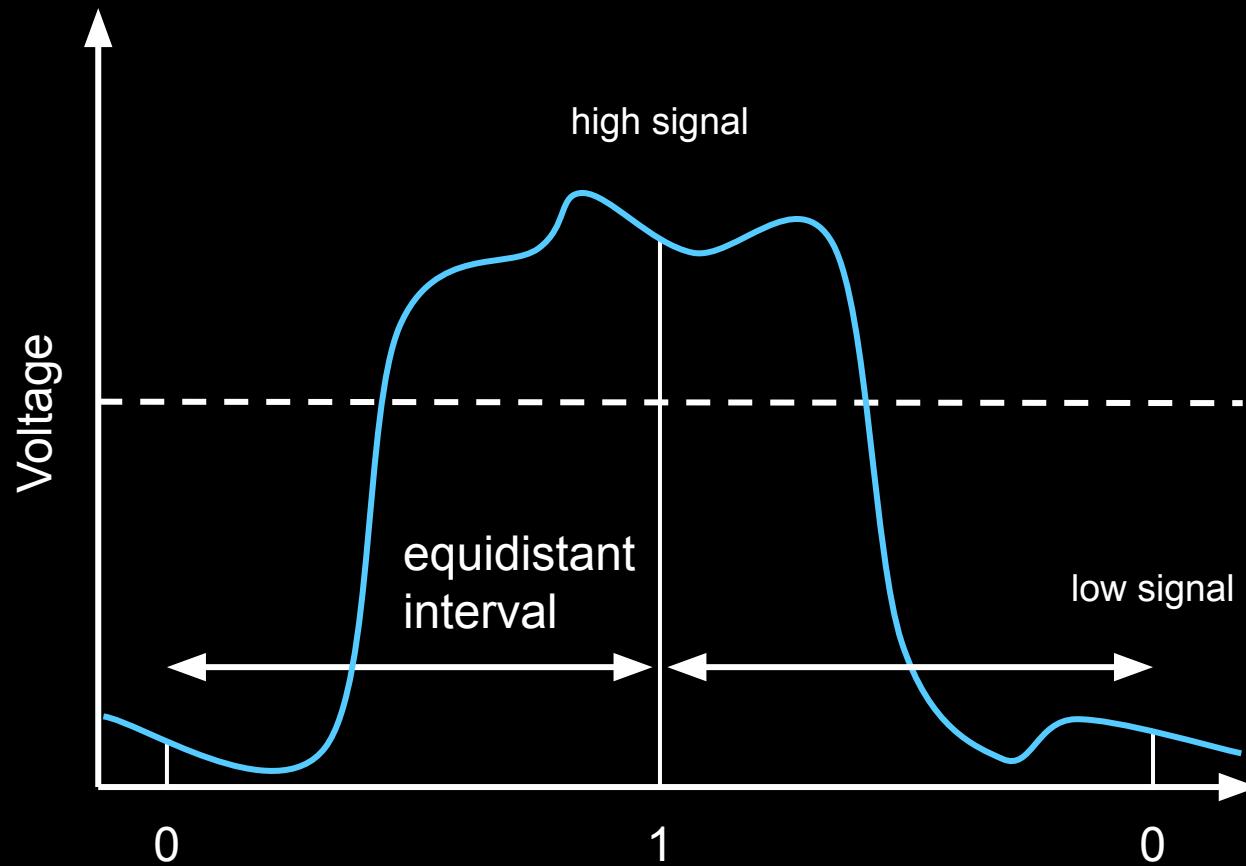




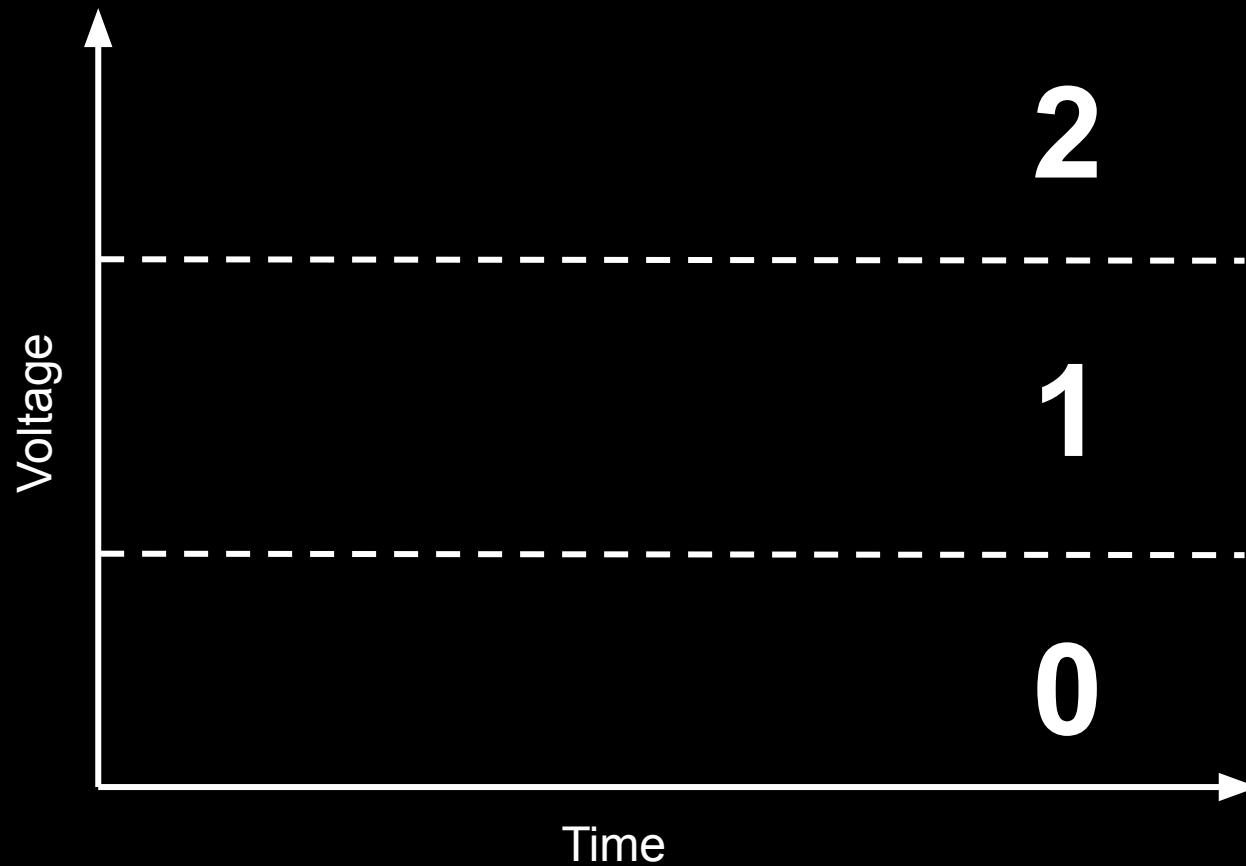


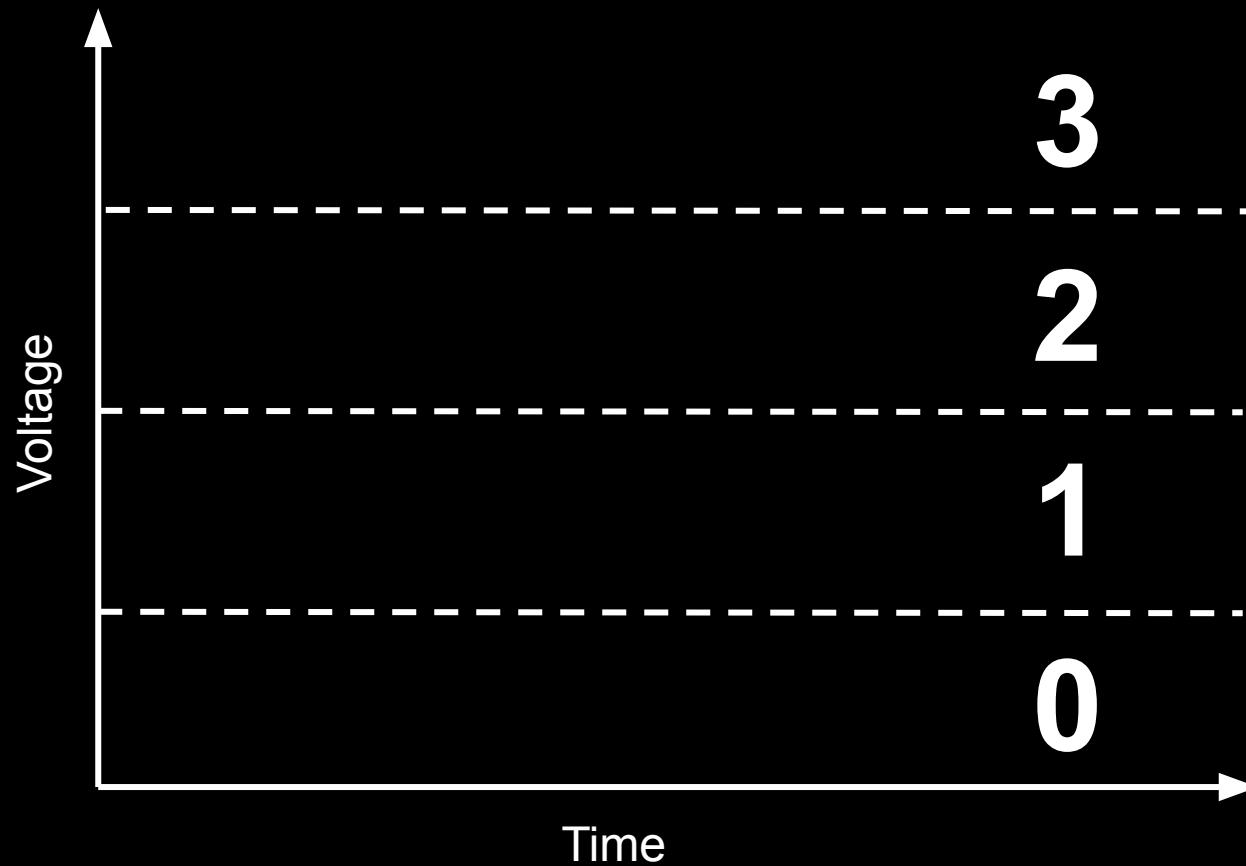


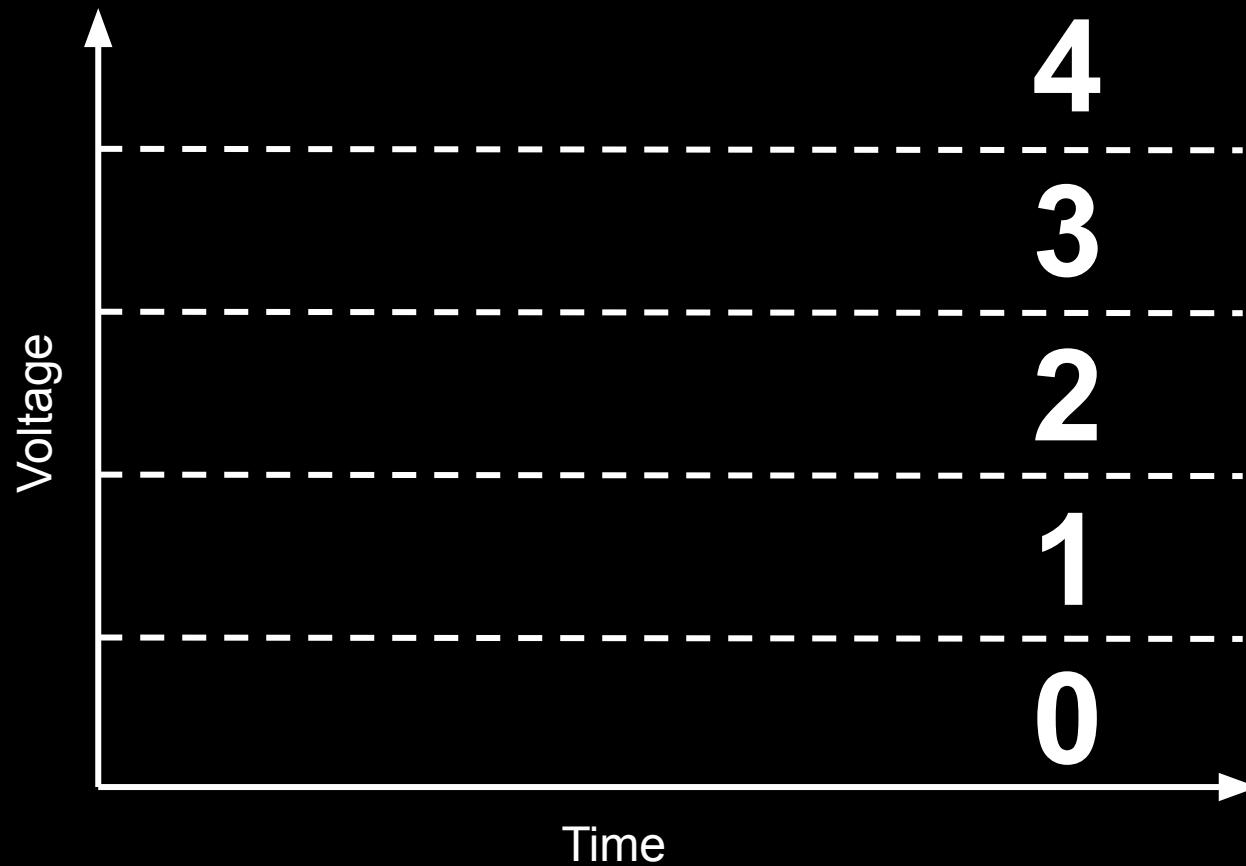


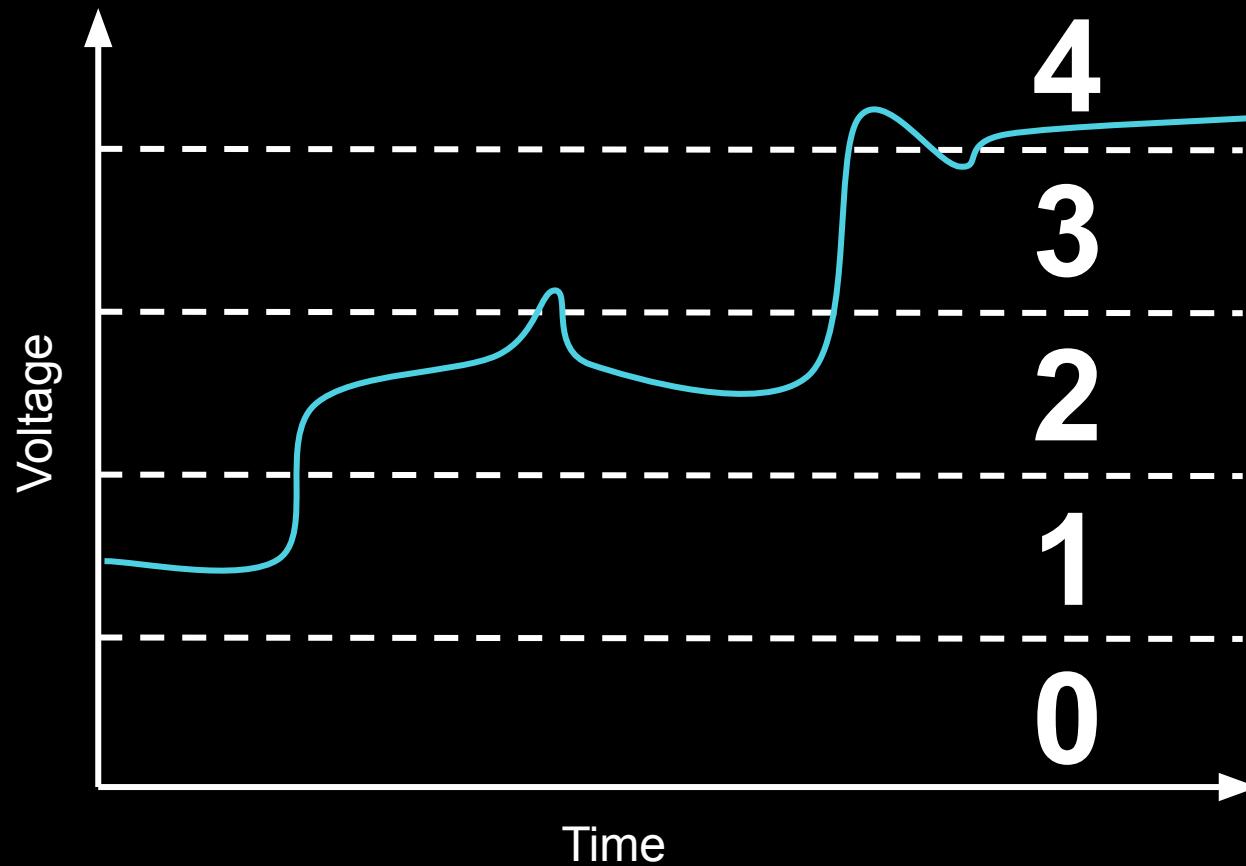


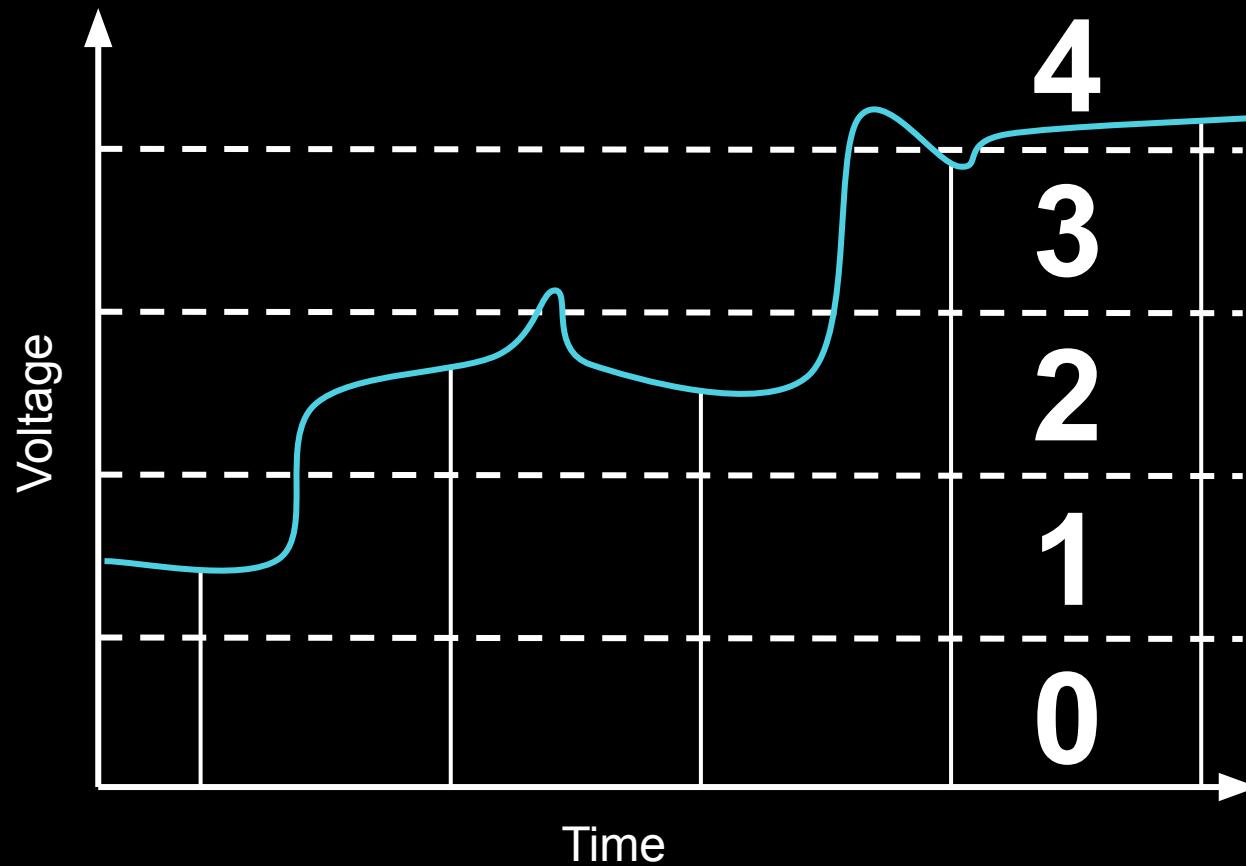
what about $R > 2$?

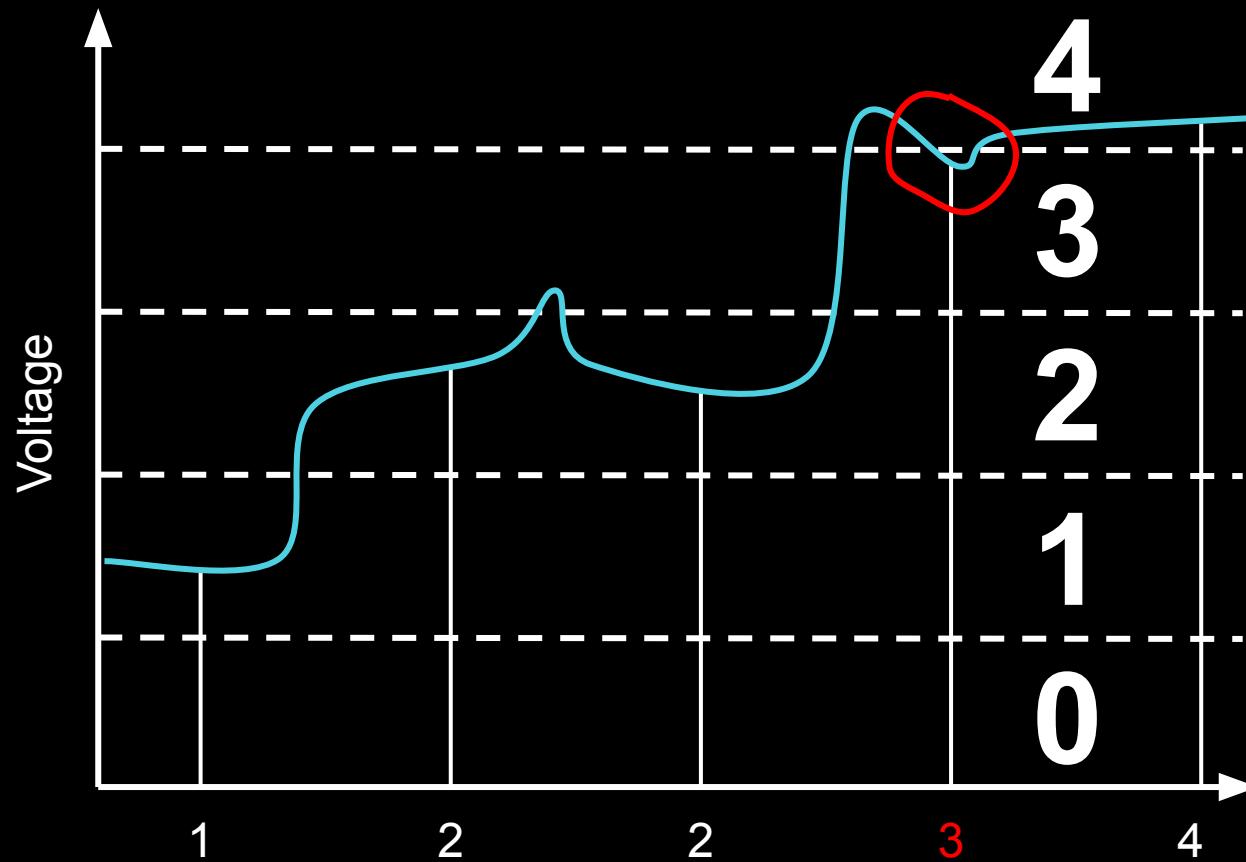












a higher base means less hardware

a higher base means less hardware
but more complex devices

a higher base means less hardware
but more complex devices
and more errors

CODES



MORSE CODE

(ALPHABETICAL)

| | | | |
|---|-----------|---|-----------|
| A | • — | N | — • |
| B | — • • • | O | — — — |
| C | — • — • | P | • — — • |
| D | — • • | Q | — — • — |
| E | • | R | • — • |
| F | • • — • | S | • • • |
| G | — — • | T | — |
| H | • • • • | U | • • — |
| I | • • | V | • • • — |
| J | • — — — | W | • — — |
| K | — • — | X | — • • — |
| L | • — • • | Y | — • — — |
| M | — — | Z | — — • • |
| | | | |
| 1 | • — — — — | 6 | — • • • • |
| 2 | • • — — — | 7 | — — • • • |
| 3 | • • • — — | 8 | — — — • • |
| 4 | • • • • — | 9 | — — — — • |
| 5 | • • • • • | 0 | — — — — — |

representing text

| | | | | | | | | |
|----|----|----|----|-----|----|----|----|-----|
| 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 |

| Decimal | Binary | Octal | Hex | ASCII | Decimal | Binary | Octal | Hex | ASCII | Decimal | Binary | Octal | Hex | ASCII | Decimal | Binary | Octal | Hex | ASCII |
|---------|----------|-------|-----|-------|---------|----------|-------|-----|-------|---------|----------|-------|-----|-------|---------|----------|-------|-----|-------|
| 0 | 00000000 | 000 | 00 | NUL | 32 | 00100000 | 040 | 20 | SP | 64 | 01000000 | 100 | 40 | @ | 96 | 01100000 | 140 | 60 | ` |
| 1 | 00000001 | 001 | 01 | SOH | 33 | 00100001 | 041 | 21 | ! | 65 | 01000001 | 101 | 41 | A | 97 | 01100001 | 141 | 61 | a |
| 2 | 00000010 | 002 | 02 | STX | 34 | 00100010 | 042 | 22 | " | 66 | 01000010 | 102 | 42 | B | 98 | 01100010 | 142 | 62 | b |
| 3 | 00000011 | 003 | 03 | ETX | 35 | 00100011 | 043 | 23 | # | 67 | 01000011 | 103 | 43 | C | 99 | 01100011 | 143 | 63 | c |
| 4 | 00000100 | 004 | 04 | EOT | 36 | 00100100 | 044 | 24 | \$ | 68 | 01000100 | 104 | 44 | D | 100 | 01100100 | 144 | 64 | d |
| 5 | 00000101 | 005 | 05 | ENQ | 37 | 00100101 | 045 | 25 | % | 69 | 01000101 | 105 | 45 | E | 101 | 01100101 | 145 | 65 | e |
| 6 | 00000110 | 006 | 06 | ACK | 38 | 00100110 | 046 | 26 | & | 70 | 01000110 | 106 | 46 | F | 102 | 01100110 | 146 | 66 | f |
| 7 | 00000111 | 007 | 07 | BEL | 39 | 00100111 | 047 | 27 | ' | 71 | 01000111 | 107 | 47 | G | 103 | 01100111 | 147 | 67 | g |
| 8 | 00001000 | 010 | 08 | BS | 40 | 00101000 | 050 | 28 | (| 72 | 01001000 | 110 | 48 | H | 104 | 01101000 | 150 | 68 | h |
| 9 | 00001001 | 011 | 09 | HT | 41 | 00101001 | 051 | 29 |) | 73 | 01001001 | 111 | 49 | I | 105 | 01101001 | 151 | 69 | i |
| 10 | 00001010 | 012 | 0A | LF | 42 | 00101010 | 052 | 2A | * | 74 | 01001010 | 112 | 4A | J | 106 | 01101010 | 152 | 6A | j |
| 11 | 00001011 | 013 | 0B | VT | 43 | 00101011 | 053 | 2B | + | 75 | 01001011 | 113 | 4B | K | 107 | 01101011 | 153 | 6B | k |
| 12 | 00001100 | 014 | 0C | FF | 44 | 00101100 | 054 | 2C | . | 76 | 01001100 | 114 | 4C | L | 108 | 01101100 | 154 | 6C | l |
| 13 | 00001101 | 015 | 0D | CR | 45 | 00101101 | 055 | 2D | - | 77 | 01001101 | 115 | 4D | M | 109 | 01101101 | 155 | 6D | m |
| 14 | 00001110 | 016 | 0E | SO | 46 | 00101110 | 056 | 2E | . | 78 | 01001110 | 116 | 4E | N | 110 | 01101110 | 156 | 6E | n |
| 15 | 00001111 | 017 | 0F | SI | 47 | 00101111 | 057 | 2F | / | 79 | 01001111 | 117 | 4F | O | 111 | 01101111 | 157 | 6F | o |
| 16 | 00010000 | 020 | 10 | DLE | 48 | 00110000 | 060 | 30 | 0 | 80 | 01010000 | 120 | 50 | P | 112 | 01110000 | 160 | 70 | p |
| 17 | 00010001 | 021 | 11 | DC1 | 49 | 00110001 | 061 | 31 | 1 | 81 | 01010001 | 121 | 51 | Q | 113 | 01110001 | 161 | 71 | q |
| 18 | 00010010 | 022 | 12 | DC2 | 50 | 00110010 | 062 | 32 | 2 | 82 | 01010010 | 122 | 52 | R | 114 | 01110010 | 162 | 72 | r |
| 19 | 00010011 | 023 | 13 | DC3 | 51 | 00110011 | 063 | 33 | 3 | 83 | 01010011 | 123 | 53 | S | 115 | 01110011 | 163 | 73 | s |
| 20 | 00010100 | 024 | 14 | DC4 | 52 | 00110100 | 064 | 34 | 4 | 84 | 01010100 | 124 | 54 | T | 116 | 01110100 | 164 | 74 | t |
| 21 | 00010101 | 025 | 15 | NAK | 53 | 00110101 | 065 | 35 | 5 | 85 | 01010101 | 125 | 55 | U | 117 | 01110101 | 165 | 75 | u |
| 22 | 00010110 | 026 | 16 | SYN | 54 | 00110110 | 066 | 36 | 6 | 86 | 01010110 | 126 | 56 | V | 118 | 01110110 | 166 | 76 | v |
| 23 | 00010111 | 027 | 17 | ETB | 55 | 00110111 | 067 | 37 | 7 | 87 | 01010111 | 127 | 57 | W | 119 | 01110111 | 167 | 77 | w |
| 24 | 00011000 | 030 | 18 | CAN | 56 | 00111000 | 070 | 38 | 8 | 88 | 01011000 | 130 | 58 | X | 120 | 01111000 | 170 | 78 | x |
| 25 | 00011001 | 031 | 19 | EM | 57 | 00111001 | 071 | 39 | 9 | 89 | 01011001 | 131 | 59 | Y | 121 | 01111001 | 171 | 79 | y |
| 26 | 00011010 | 032 | 1A | SUB | 58 | 00111010 | 072 | 3A | : | 90 | 01011010 | 132 | 5A | Z | 122 | 01111010 | 172 | 7A | z |
| 27 | 00011011 | 033 | 1B | ESC | 59 | 00111011 | 073 | 3B | ; | 91 | 01011011 | 133 | 5B | [| 123 | 01111011 | 173 | 7B | { |
| 28 | 00011100 | 034 | 1C | FS | 60 | 00111100 | 074 | 3C | < | 92 | 01011100 | 134 | 5C | \ | 124 | 01111100 | 174 | 7C | |
| 29 | 00011101 | 035 | 1D | GS | 61 | 00111101 | 075 | 3D | = | 93 | 01011101 | 135 | 5D |] | 125 | 01111101 | 175 | 7D | } |
| 30 | 00011110 | 036 | 1E | RS | 62 | 00111110 | 076 | 3E | > | 94 | 01011110 | 136 | 5E | ^ | 126 | 01111110 | 176 | 7E | ~ |
| 31 | 00011111 | 037 | 1F | US | 63 | 00111111 | 077 | 3F | ? | 95 | 01011111 | 137 | 5F | - | 127 | 01111111 | 177 | 7F | DEL |



1F600



1F601



1F602



1F603

...



1F648



1F649



1F64A



1F64B

Unicode



1F600



1F601



1F602



1F603

...



1F648



1F649



1F64A



1F64B

hexadecimal

1

A

D

(hexadecimal)

16^2

16^1

16^0

1

A

D

(hexadecimal)

16^2

16^1

16^0

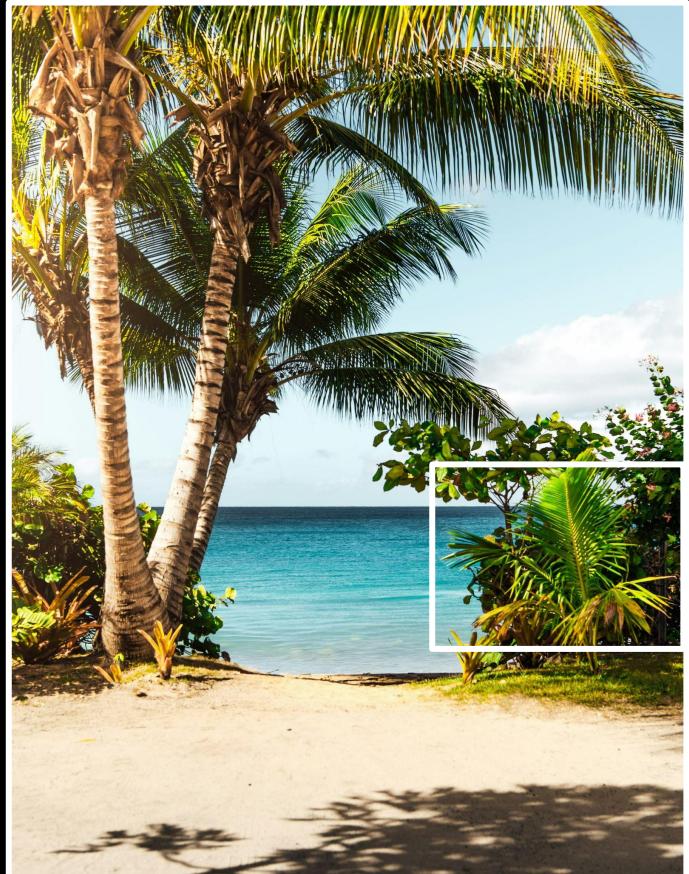
$$= 1 \times 16^2 + 10 \times 16^1 + 13 \times 16^0$$

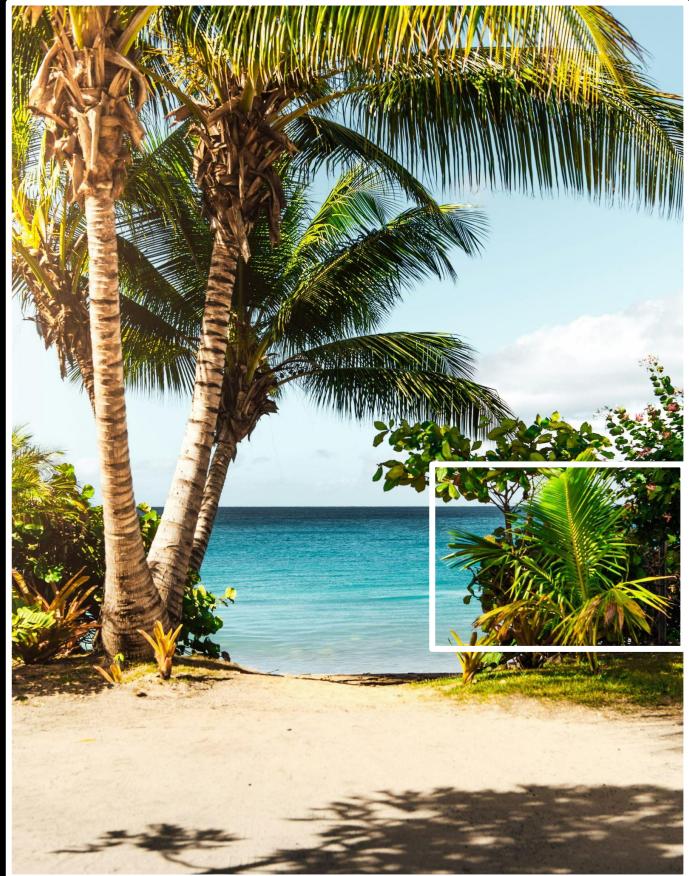
$$= 1 \times 256 + 10 \times 16 + 13 \times 1$$

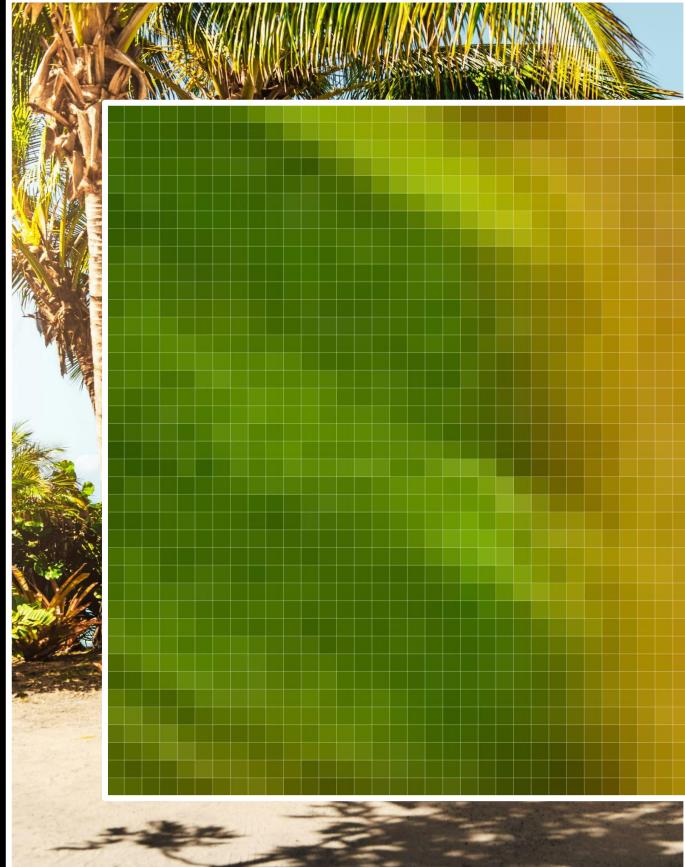
$$= \mathbf{429} \text{ (decimal)}$$

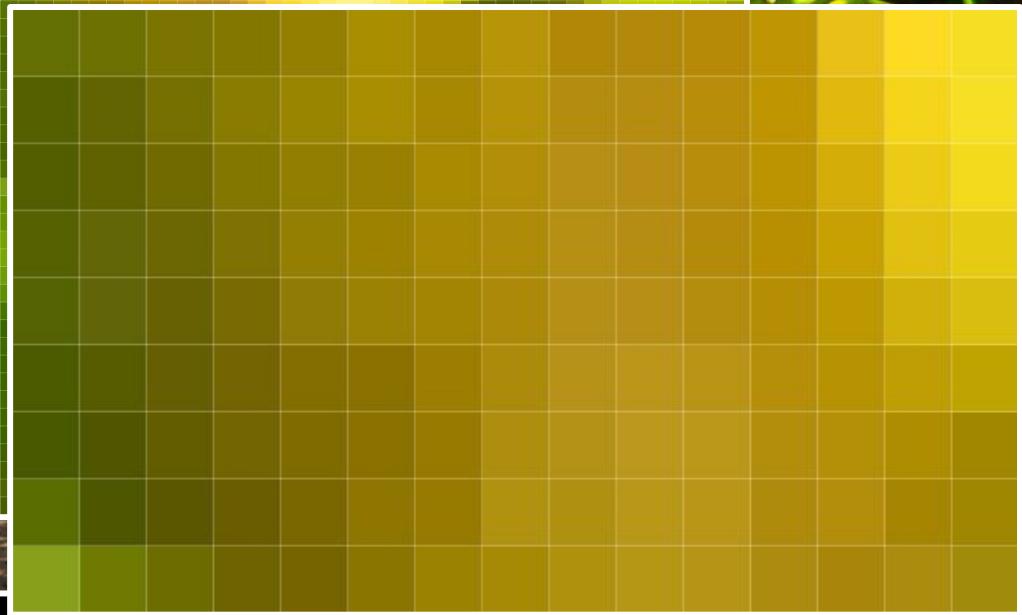
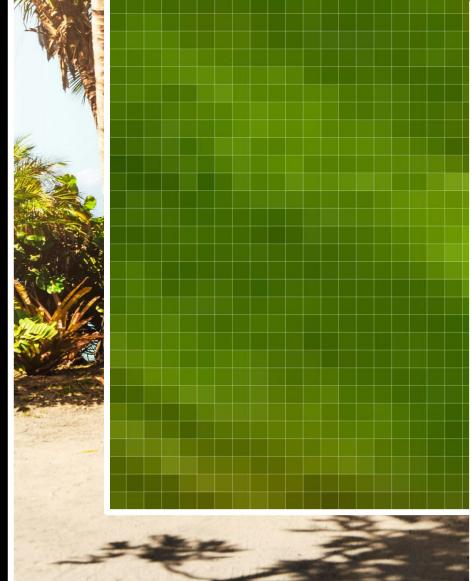
representing images

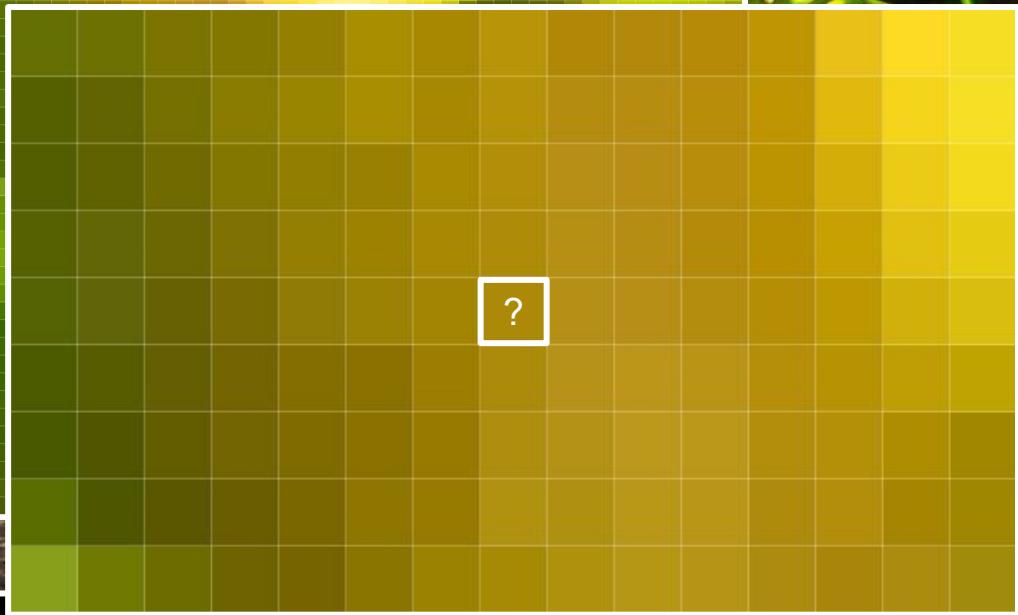
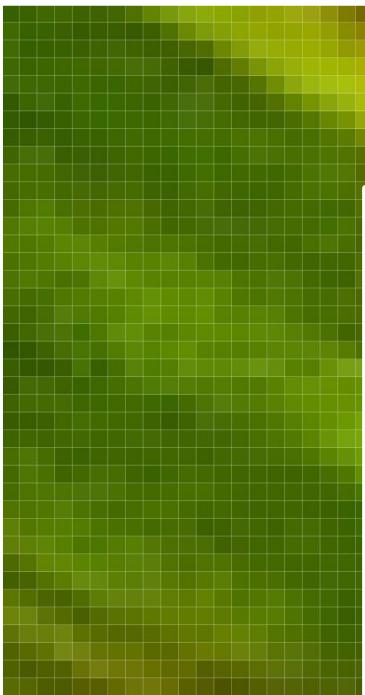
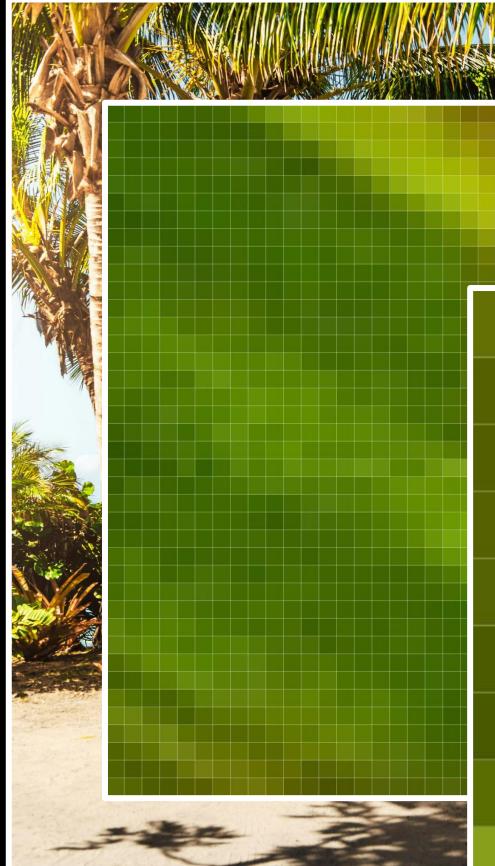




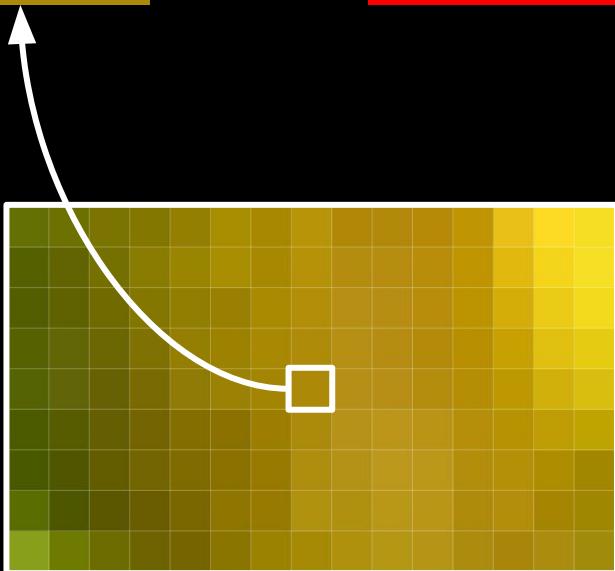




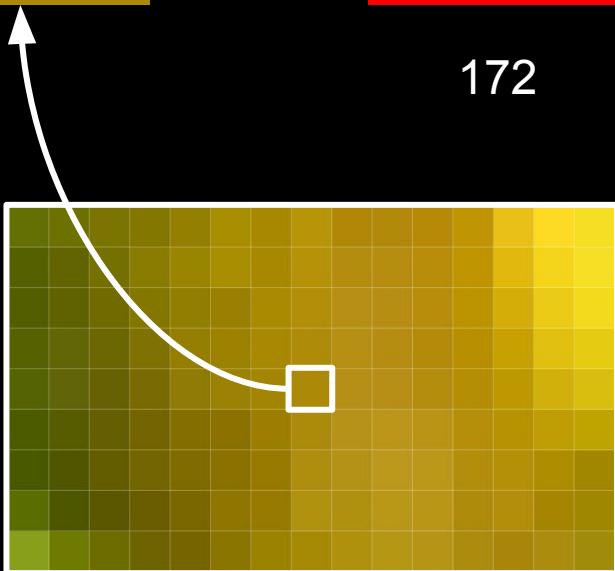




$$\text{Yellow} = \text{R} + \text{G} + \text{B}$$



$$\text{Yellow} = \text{R} + \text{G} + \text{B}$$



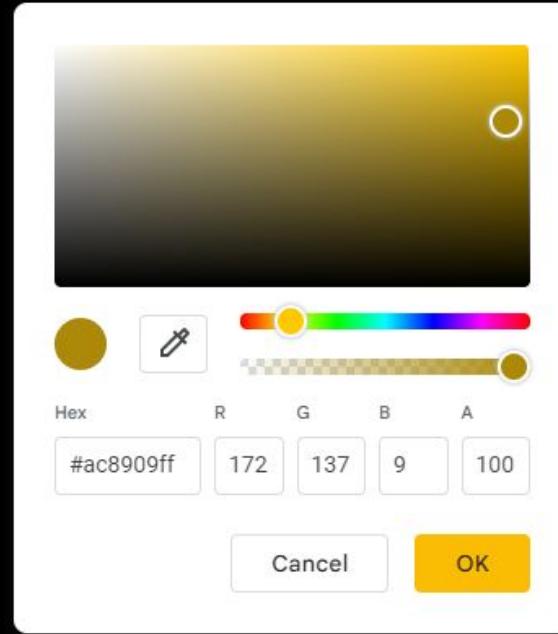
172

137

9

$$\begin{matrix} \text{Yellow} \\ = \end{matrix} \begin{matrix} \text{R} \\ 172 \end{matrix} + \begin{matrix} \text{G} \\ 137 \end{matrix} + \begin{matrix} \text{B} \\ 9 \end{matrix}$$

#AC8909



$$\text{#AC8909} = \text{R} + \text{G} + \text{B}$$

172

137

9

#AC8909

AC

89

09

$$\begin{array}{c} \text{Yellow} = \text{R} + \text{G} + \text{B} \\ \\ 172 \qquad \qquad \qquad 137 \qquad \qquad \qquad 9 \\ \\ \#AC8909 \qquad \qquad \qquad AC \qquad \qquad \qquad 89 \qquad \qquad \qquad 09 \\ \\ 10101100 \qquad \qquad \qquad 01011001 \qquad \qquad \qquad 00001001 \end{array}$$

possible colors?

R

$2^7 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$

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

$2^{23} 2^{22} 2^{21} 2^{20} 2^{19} 2^{18} 2^{17} 2^{16}$

$2^{15} 2^{14} 2^{13} 2^{12} 2^{11} 2^{10} 2^9 2^8$

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

{

{

{

256

X

256

X

256

R

G

B

$2^{23} 2^{22} 2^{21} 2^{20} 2^{19} 2^{18} 2^{17} 2^{16}$

$2^{15} 2^{14} 2^{13} 2^{12} 2^{11} 2^{10} 2^9 2^8$

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

 256

 256

 256

256

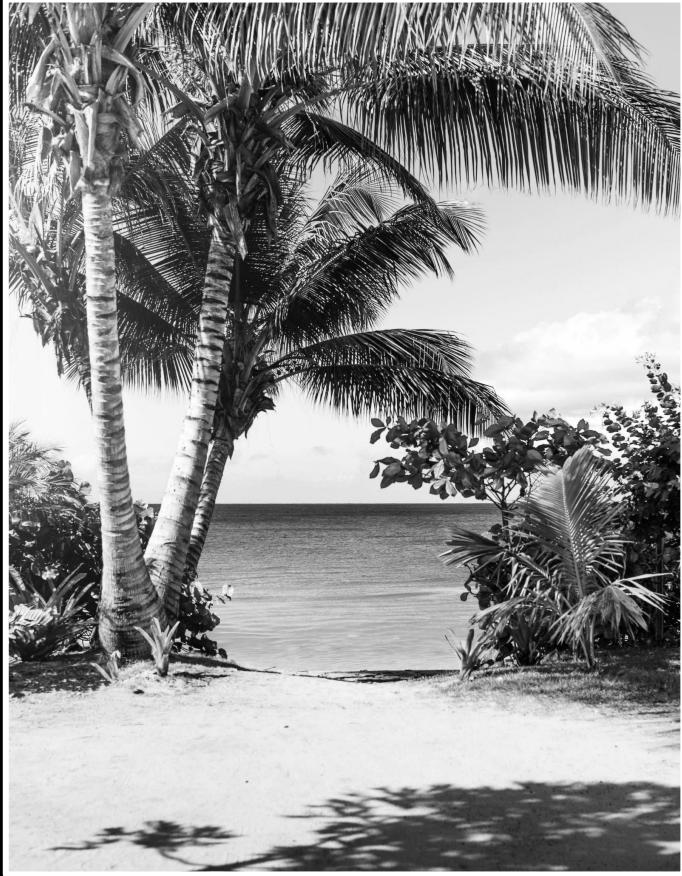


256

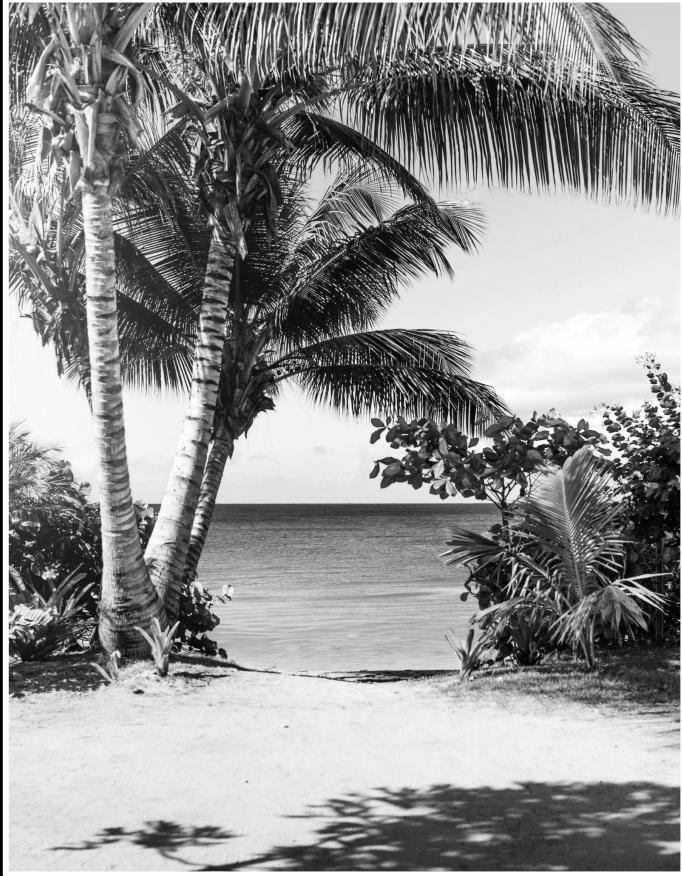


256

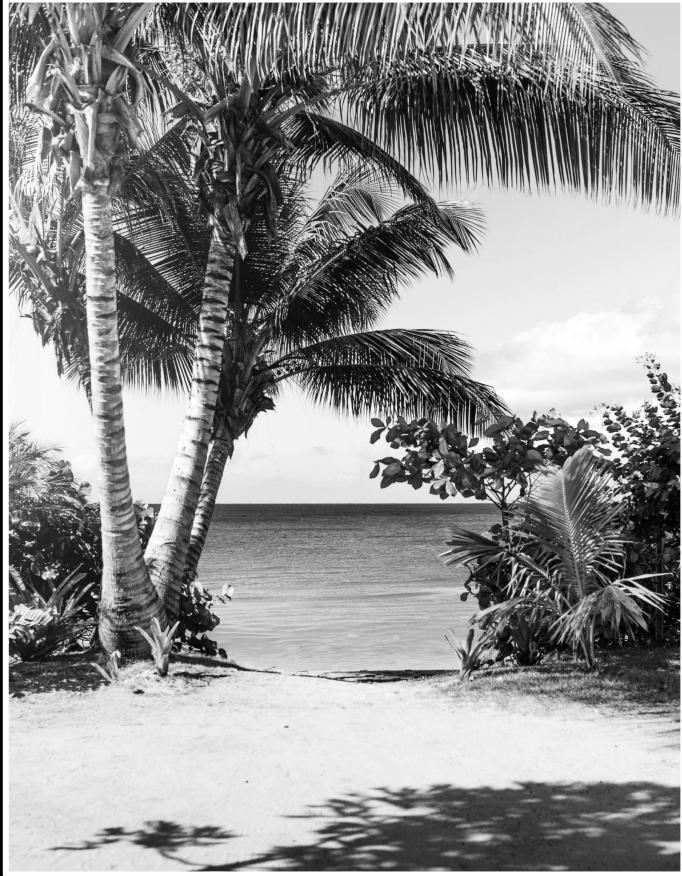
16.777.216



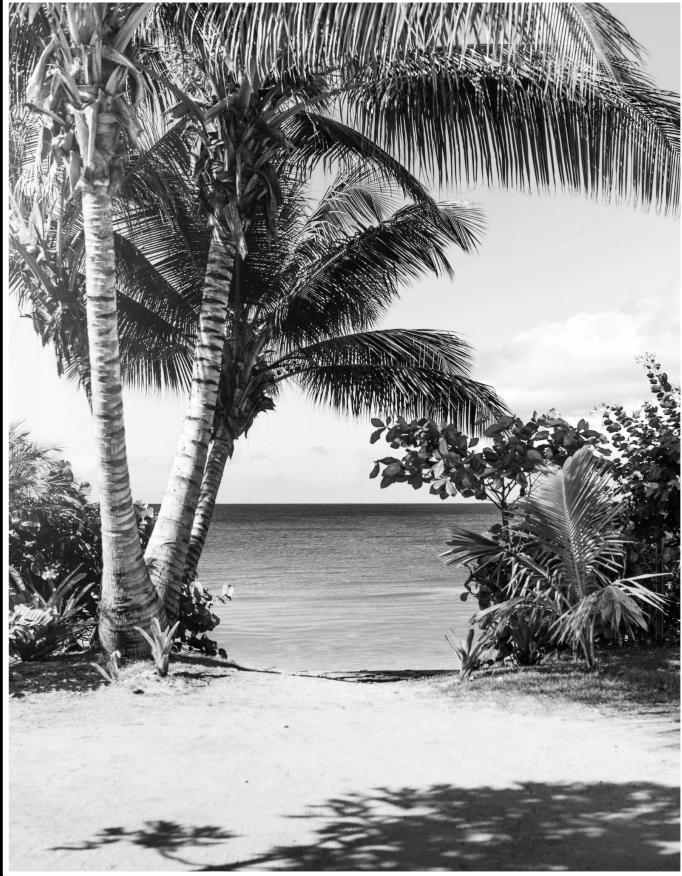
grayscale



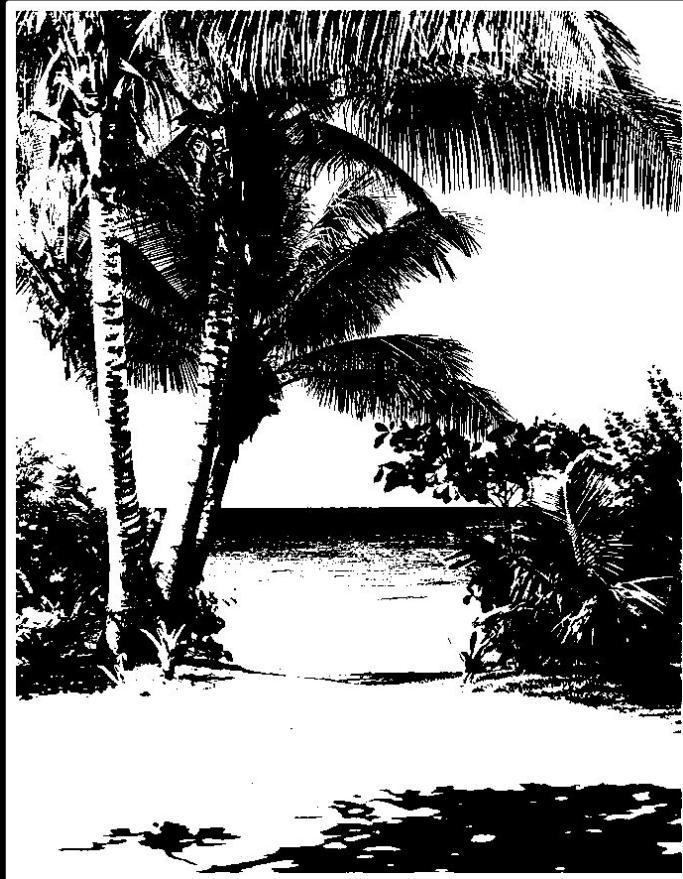
grayscale
256 shades of gray



grayscale
256 shades of gray
how many bits ?

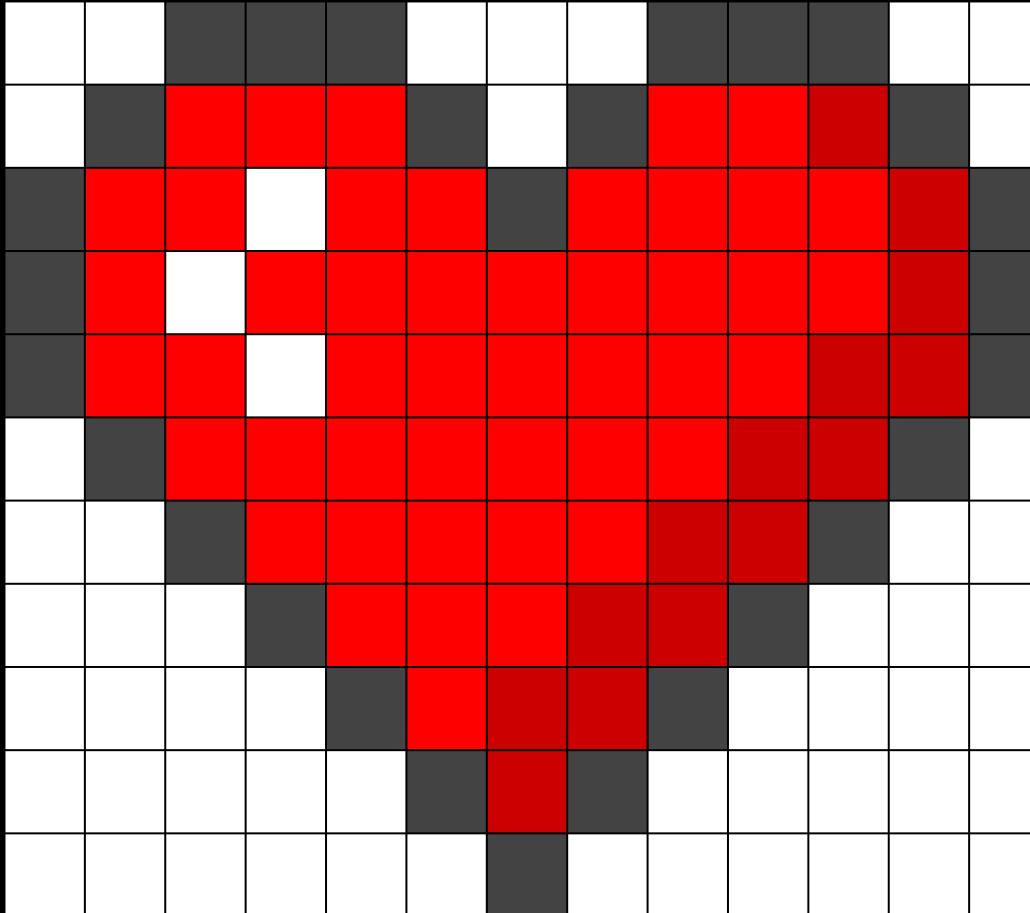


grayscale
256 shades of gray
how many bits ?



black/white

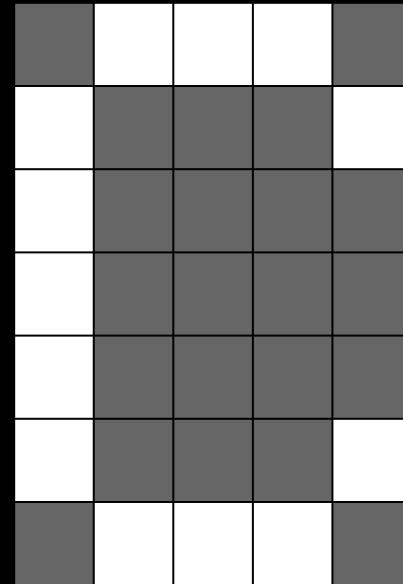
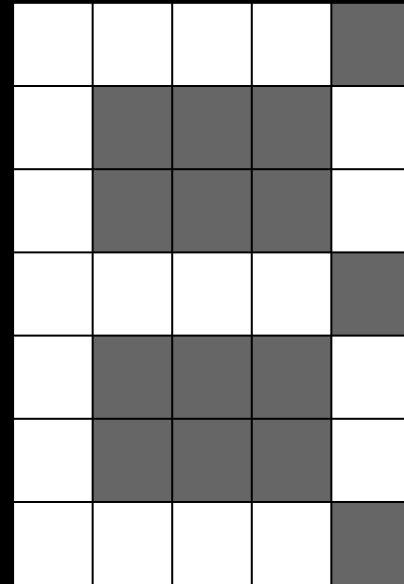
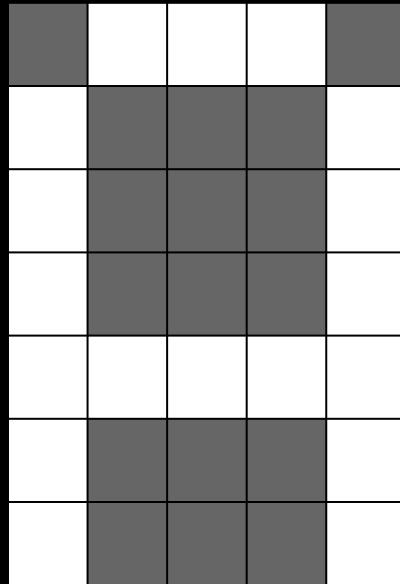


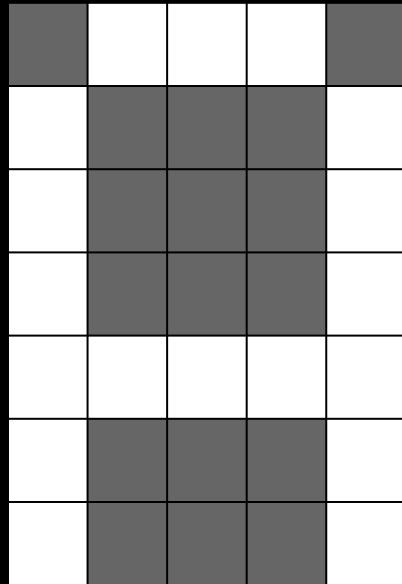




compression?

fonts





| | | | | |
|---|---|---|---|---|
| 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 0 0 0 1 1 0 0 0 1

5

| | | | | |
|---|---|---|---|---|
| 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 |

7



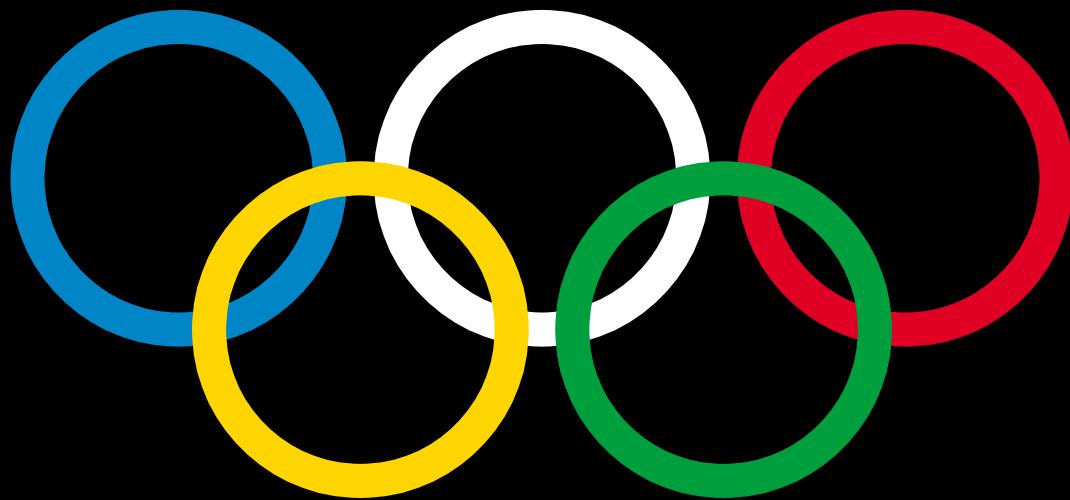
011101000110001100
0111111000110001

bitmap

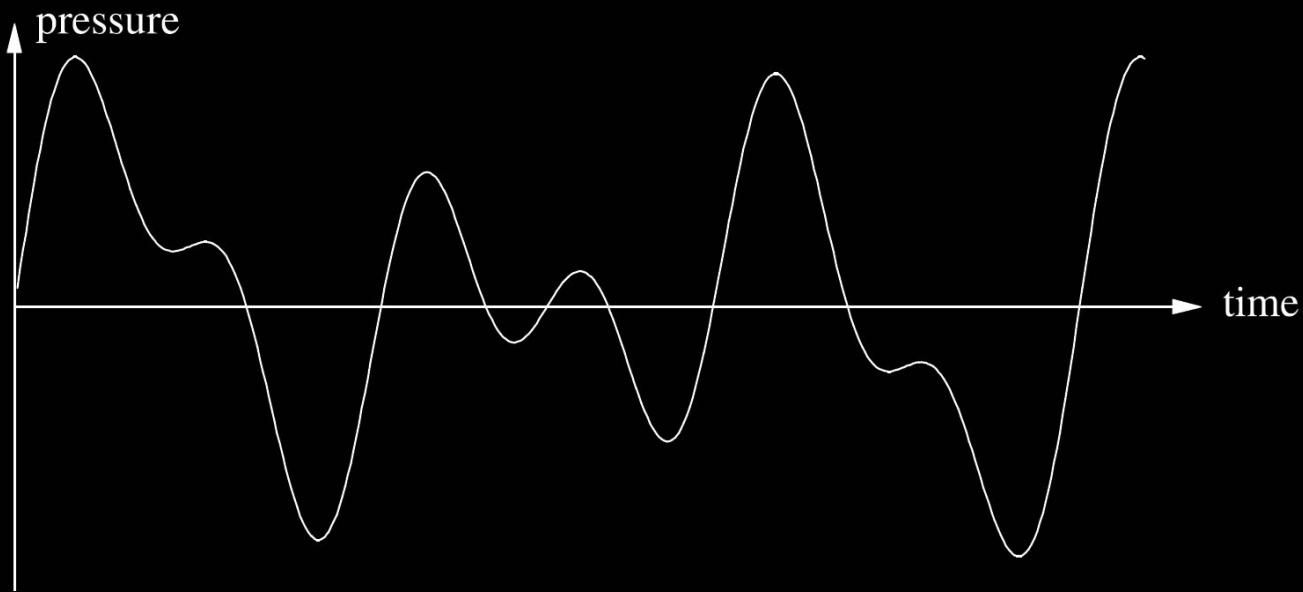
vector graphics

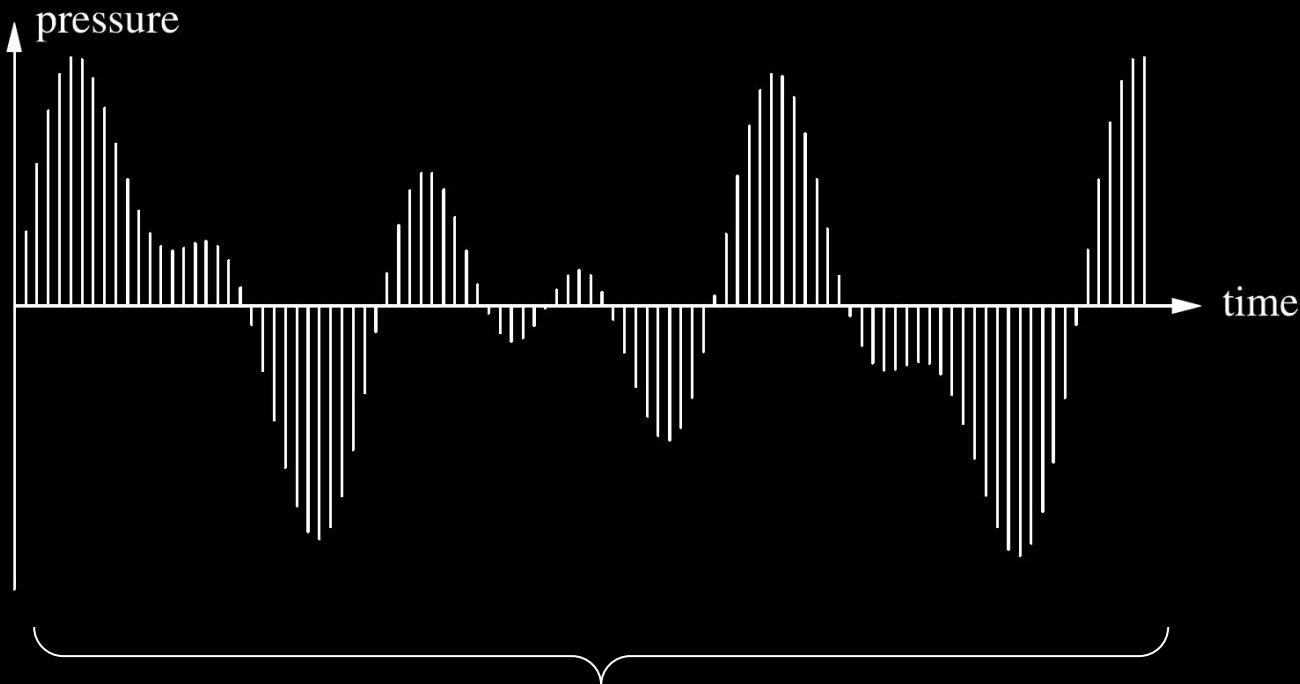
```
<svg width="440" height="220" xmlns="http://www.w3.org/2000/svg">  
  <circle cx="60" cy="60" r="50" stroke="#0085C7" stroke-width="10" fill="none" />  
  <circle cx="180" cy="60" r="50" stroke="#000000" stroke-width="10" fill="none" />  
  <circle cx="300" cy="60" r="50" stroke="#DF0024" stroke-width="10" fill="none" />  
  <circle cx="120" cy="110" r="50" stroke="#FFD500" stroke-width="10" fill="none" />  
  <circle cx="240" cy="110" r="50" stroke="#009F3D" stroke-width="10" fill="none" />  
</svg>
```

```
<svg width="440" height="220" xmlns="http://www.w3.org/2000/svg">  
  <circle cx="60" cy="60" r="50" stroke="#0085C7" stroke-width="10" fill="none" />  
  <circle cx="180" cy="60" r="50" stroke="#000000" stroke-width="10" fill="none" />  
  <circle cx="300" cy="60" r="50" stroke="#DF0024" stroke-width="10" fill="none" />  
  <circle cx="120" cy="110" r="50" stroke="#FFD500" stroke-width="10" fill="none" />  
  <circle cx="240" cy="110" r="50" stroke="#009F3D" stroke-width="10" fill="none" />  
</svg>
```



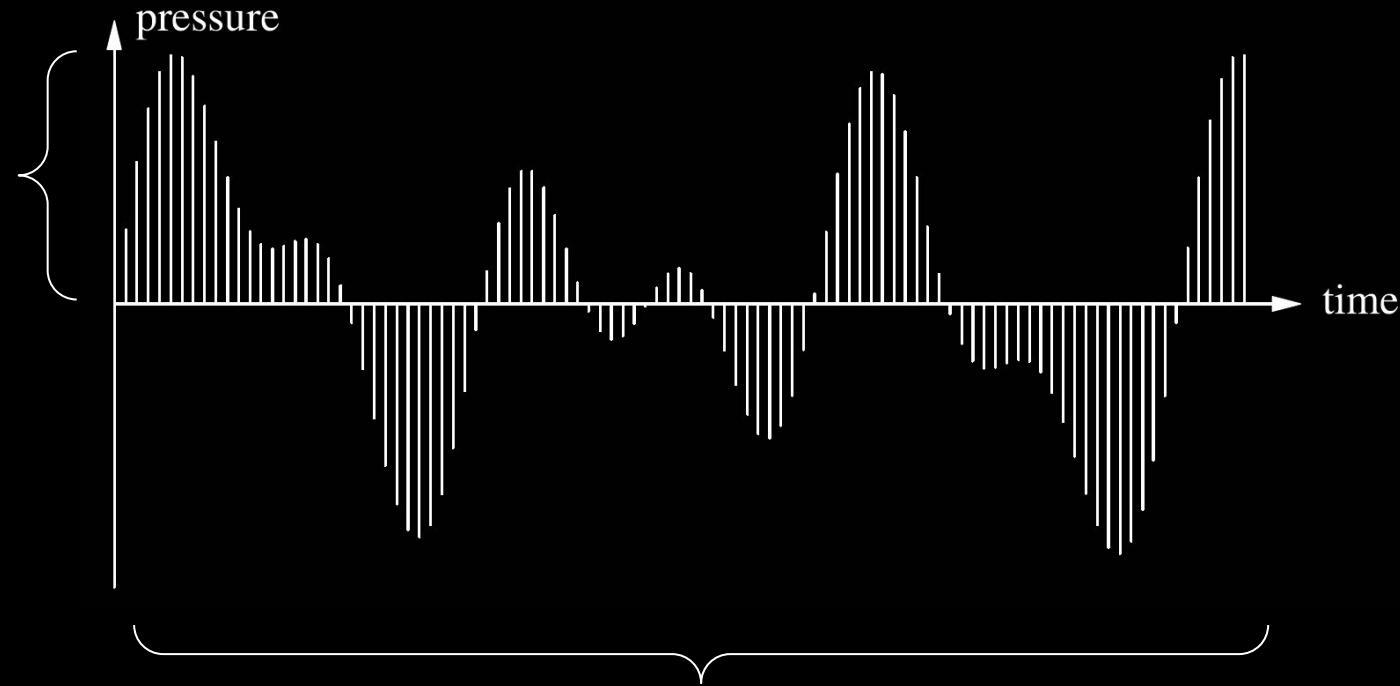
representing sound





how many samples per time unit?

bits per sample?

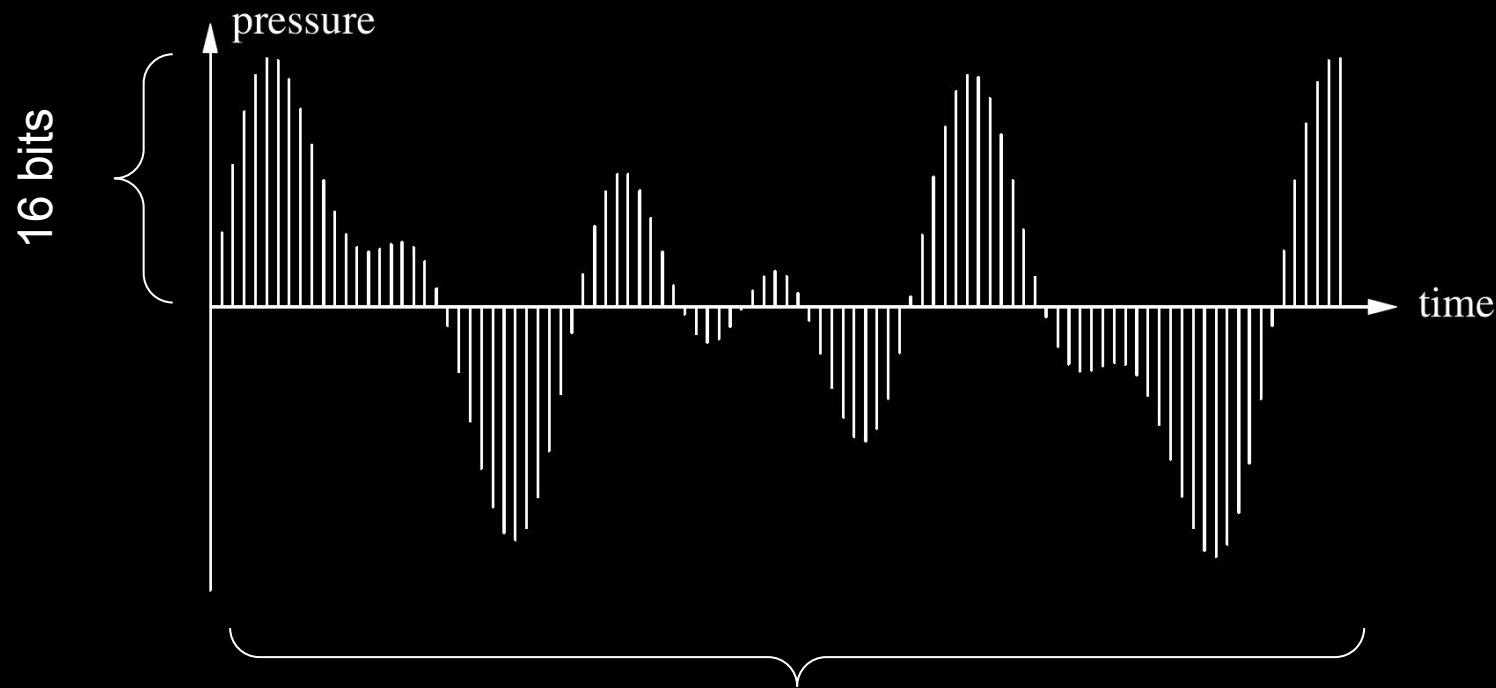


samples per time unit?

telephone



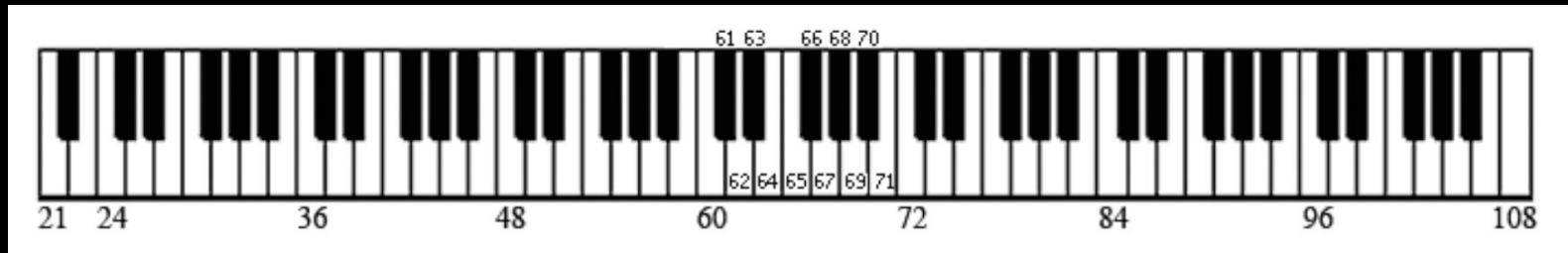
music



44.1 kHz

musical instrument digital interface (MIDI)

byte 1 = note
byte 2 = velocity



what code systems exist outside
of computers?



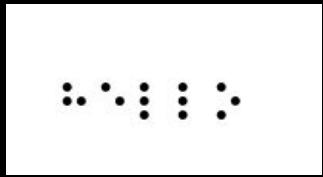
a)



b)

| MORSE CODE (ALPHABETICAL) | |
|------------------------------|-----------|
| A | • — |
| B | — • • • |
| C | — • — • |
| D | — • • |
| E | • |
| F | • • — • |
| G | — — • |
| H | • • • • |
| I | • • |
| J | — • — — |
| K | — • — |
| L | — • • • |
| M | — — |
| N | — • |
| O | — — — |
| P | • — — — |
| Q | — — • — |
| R | • — • |
| S | • • • |
| T | — |
| U | • • — |
| V | • • — — |
| W | — • — |
| X | — — • • |
| Y | — — • — |
| Z | — — — • |
| 1 | • — — — — |
| 2 | • • — — — |
| 3 | • • • — — |
| 4 | • • • • — |
| 5 | • • • • • |
| 6 | — • • • • |
| 7 | — — • • • |
| 8 | — — — • • |
| 9 | — — — — • |
| 0 | — — — — — |

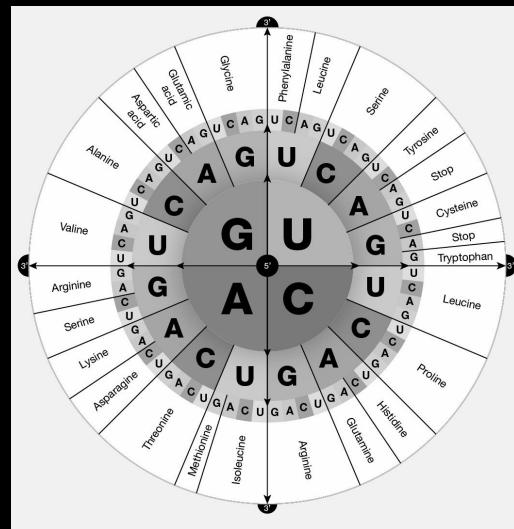
c)



d)



f)



ANALOG VS. DIGITAL

where can you stand on each side?



where can you stand on each side?



where can you stand on each side?



where can you stand on each side?



where can you stand on each side?



analog and digital clocks



Image source: [Wikipedia](#)



Image source: [Wikipedia](#)

“Digitization is the representation of an object, image, sound, document or signal by generating a series of numbers that describe a discrete set of its points or samples”

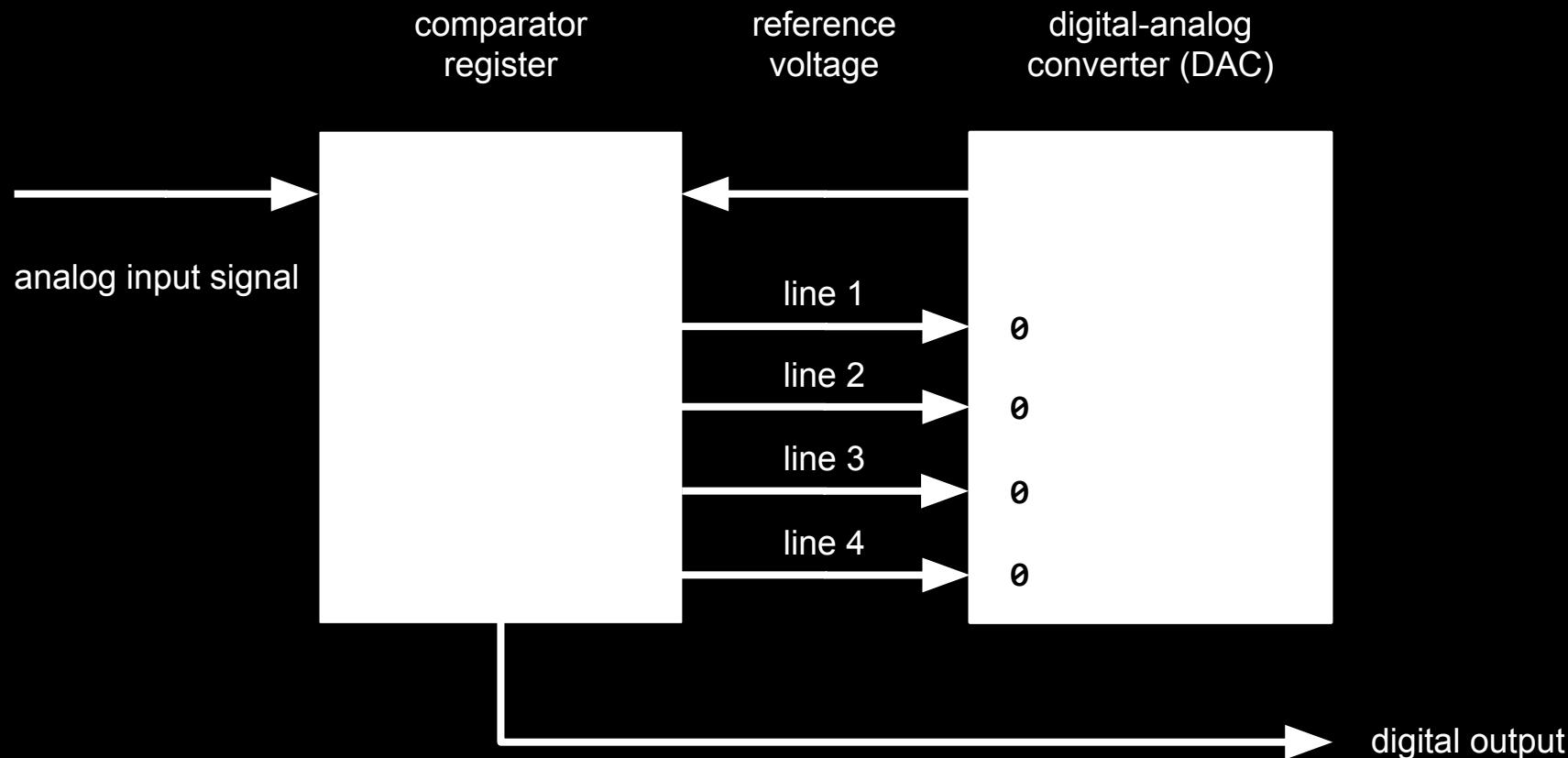
analog-to-digital converter (ADC)

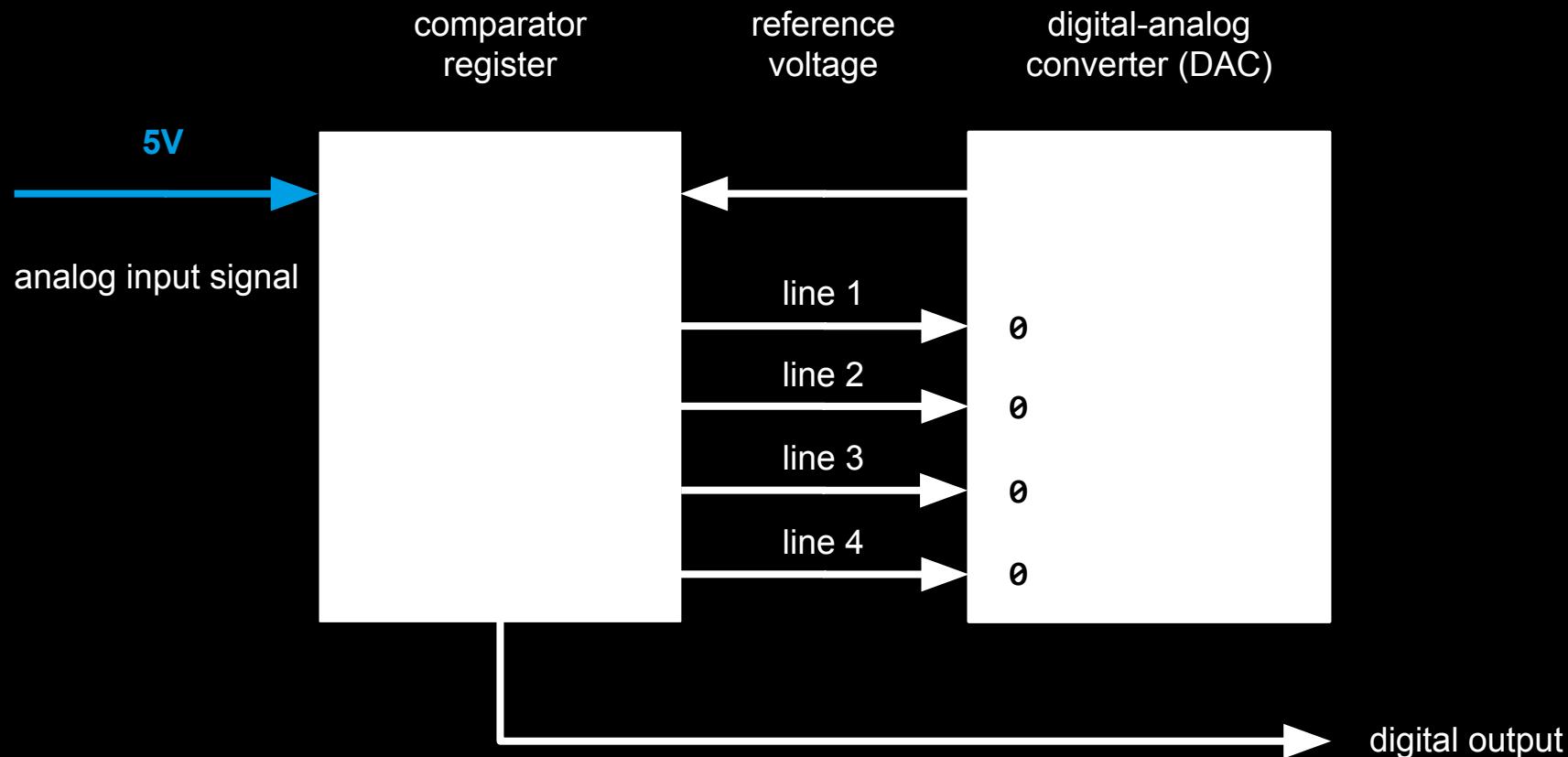


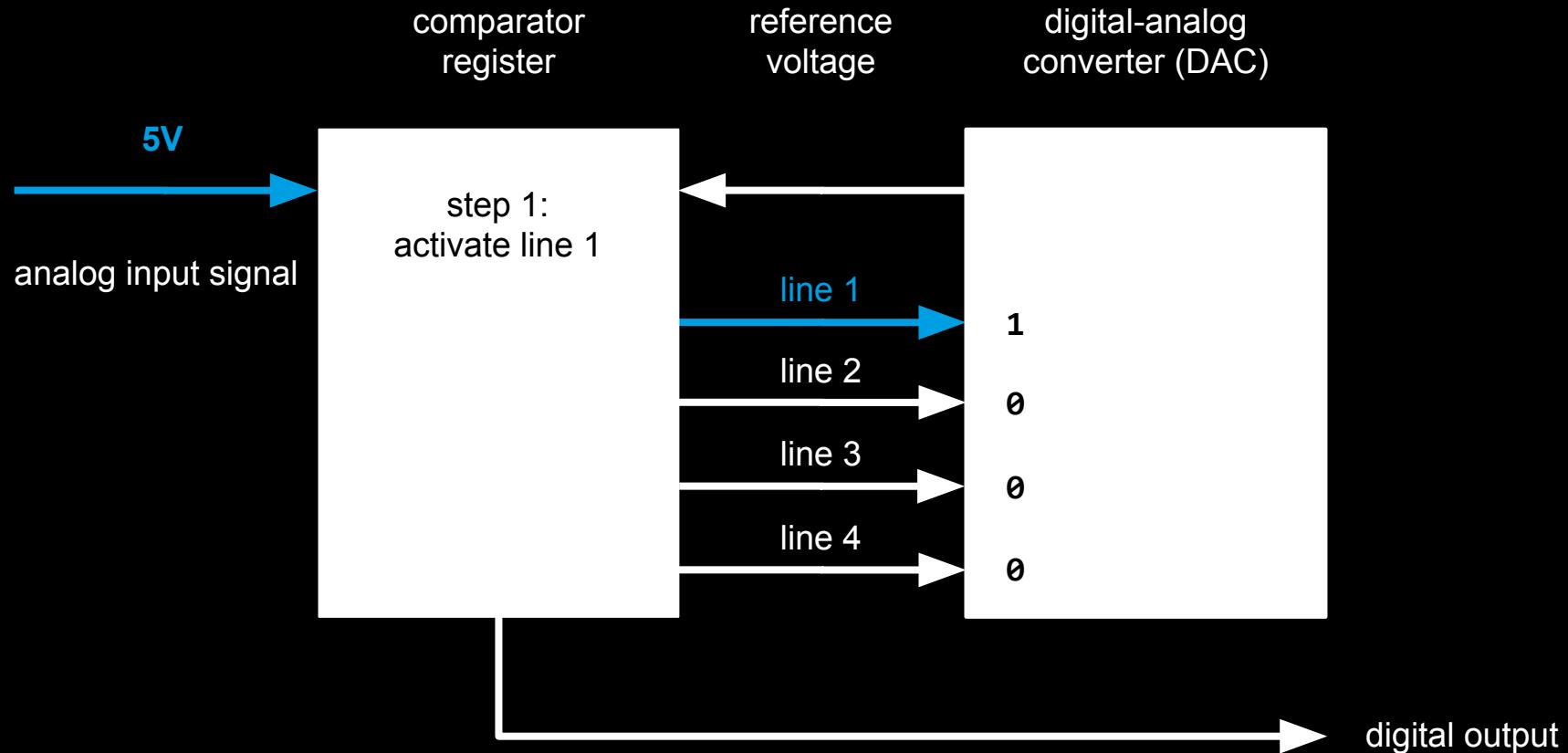
measuring of the
analog signal

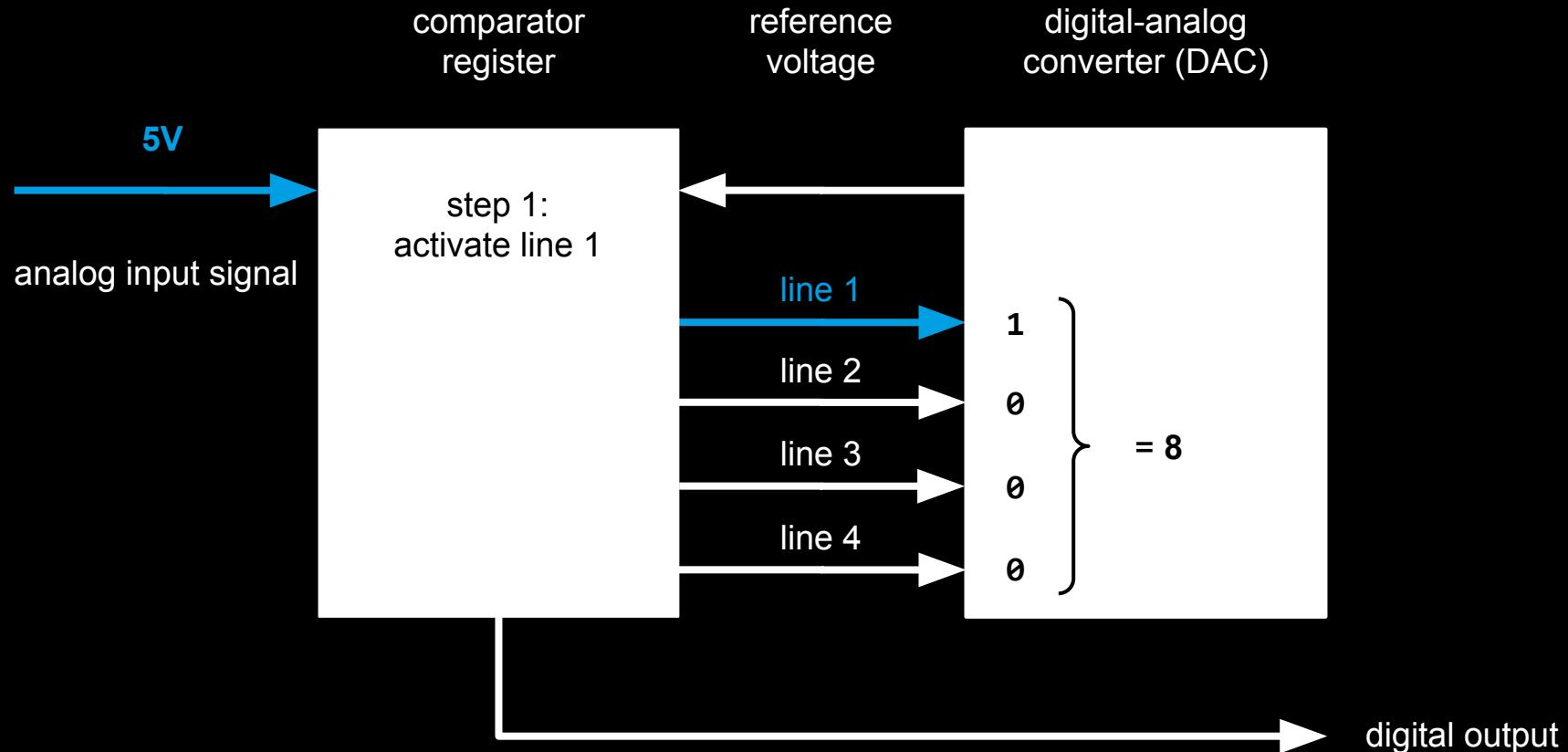
turning the
measurement into a
series of 1 and 0

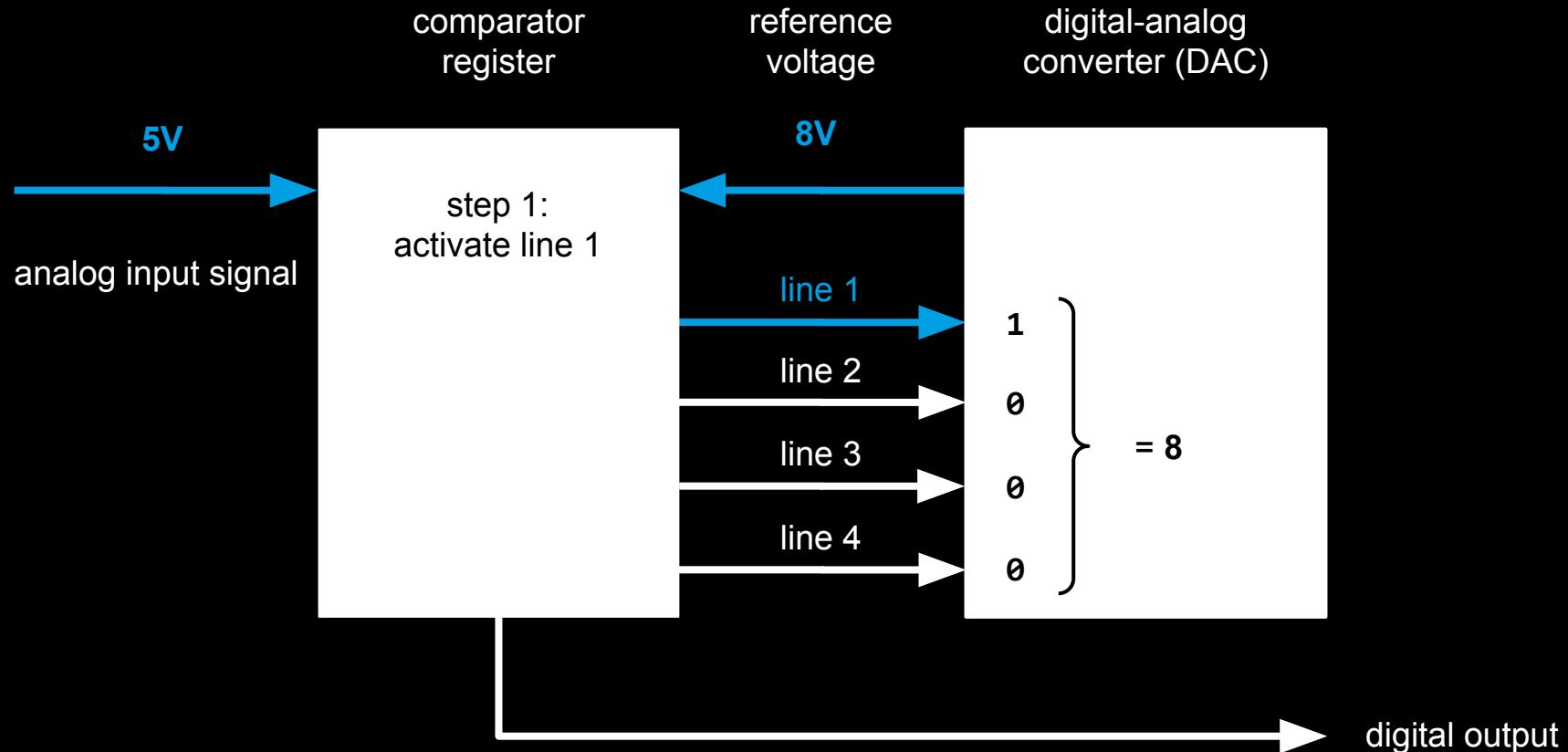
encoding the series of
numbers in binary
and assign meaning

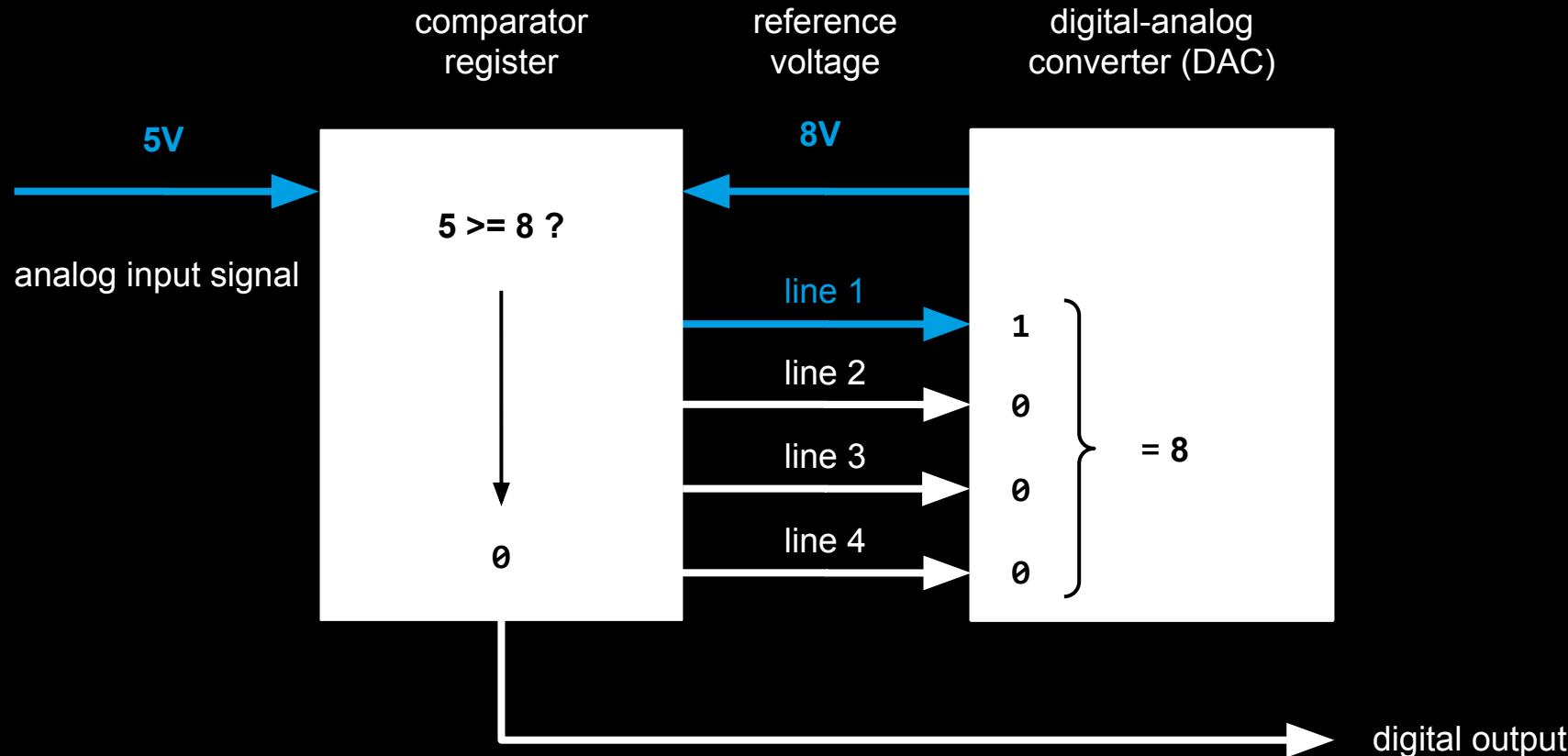


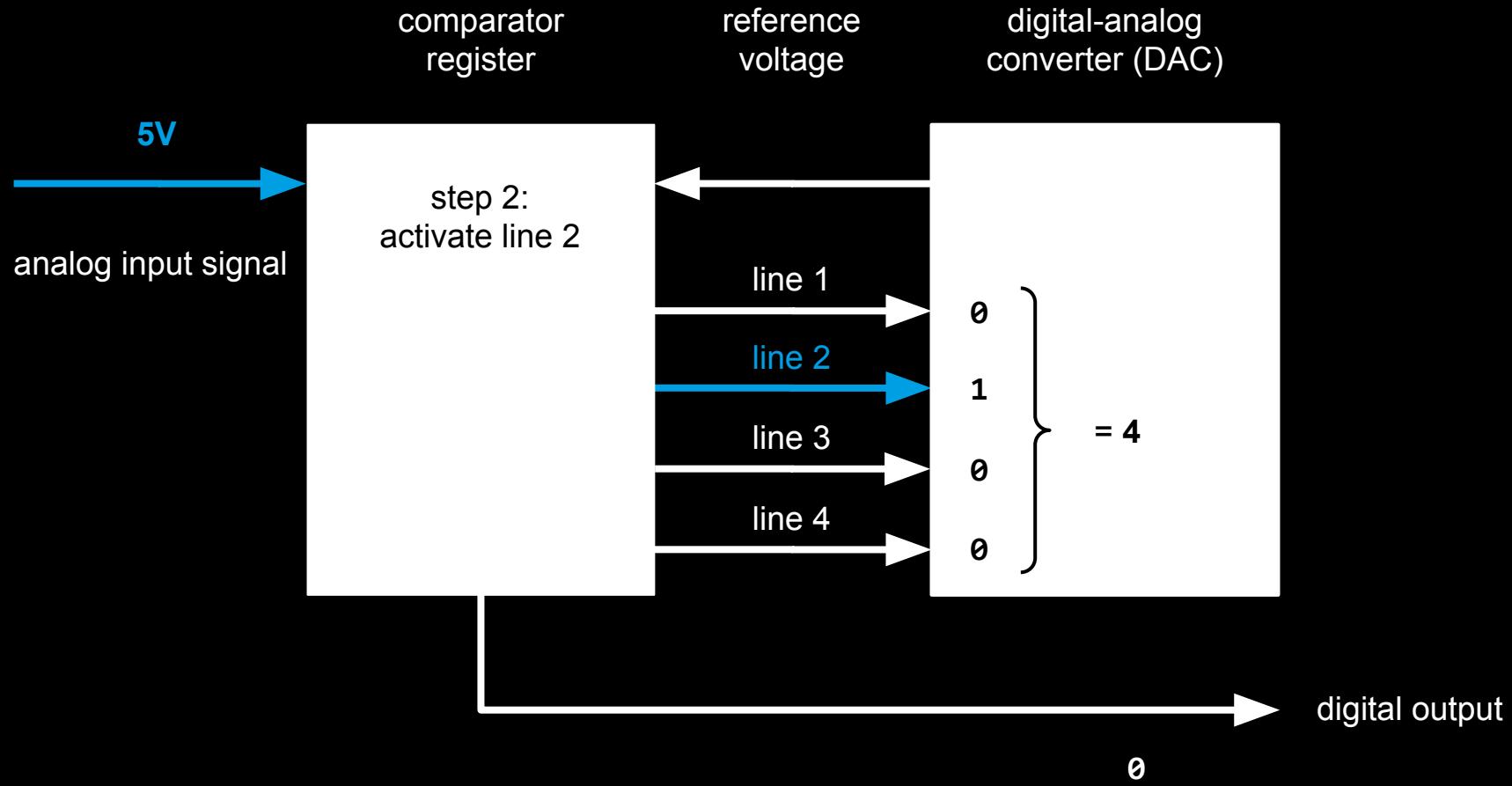


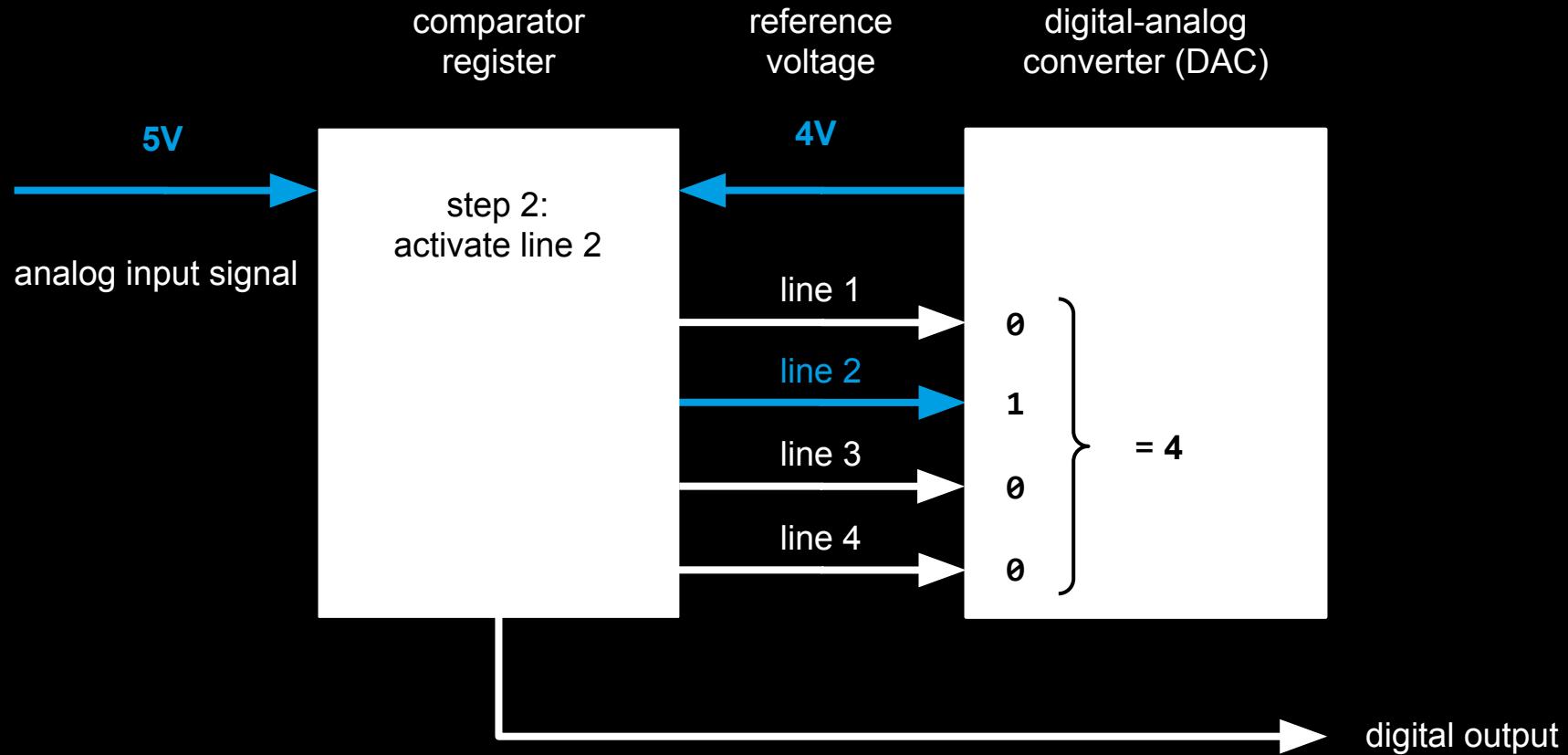


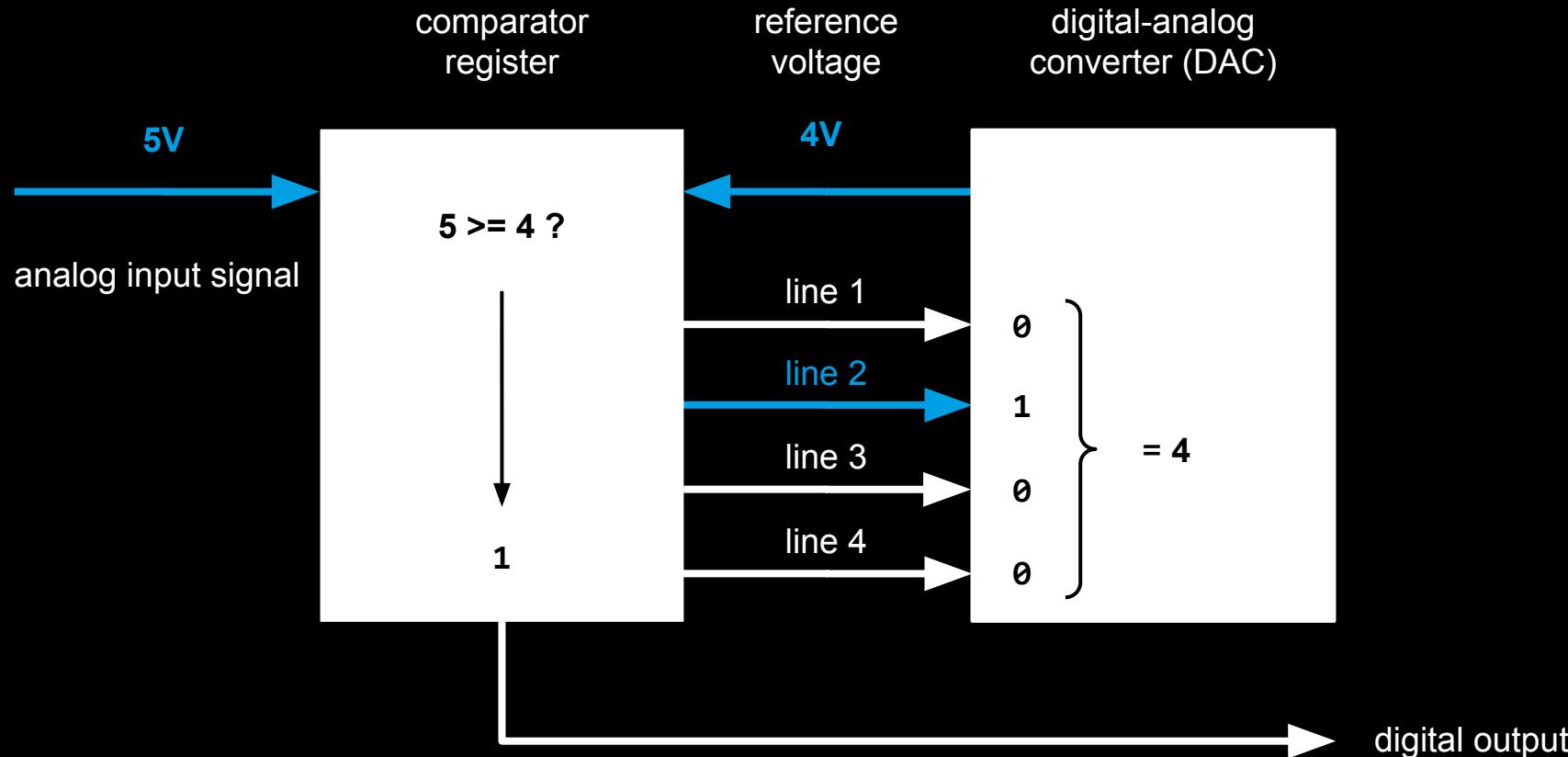


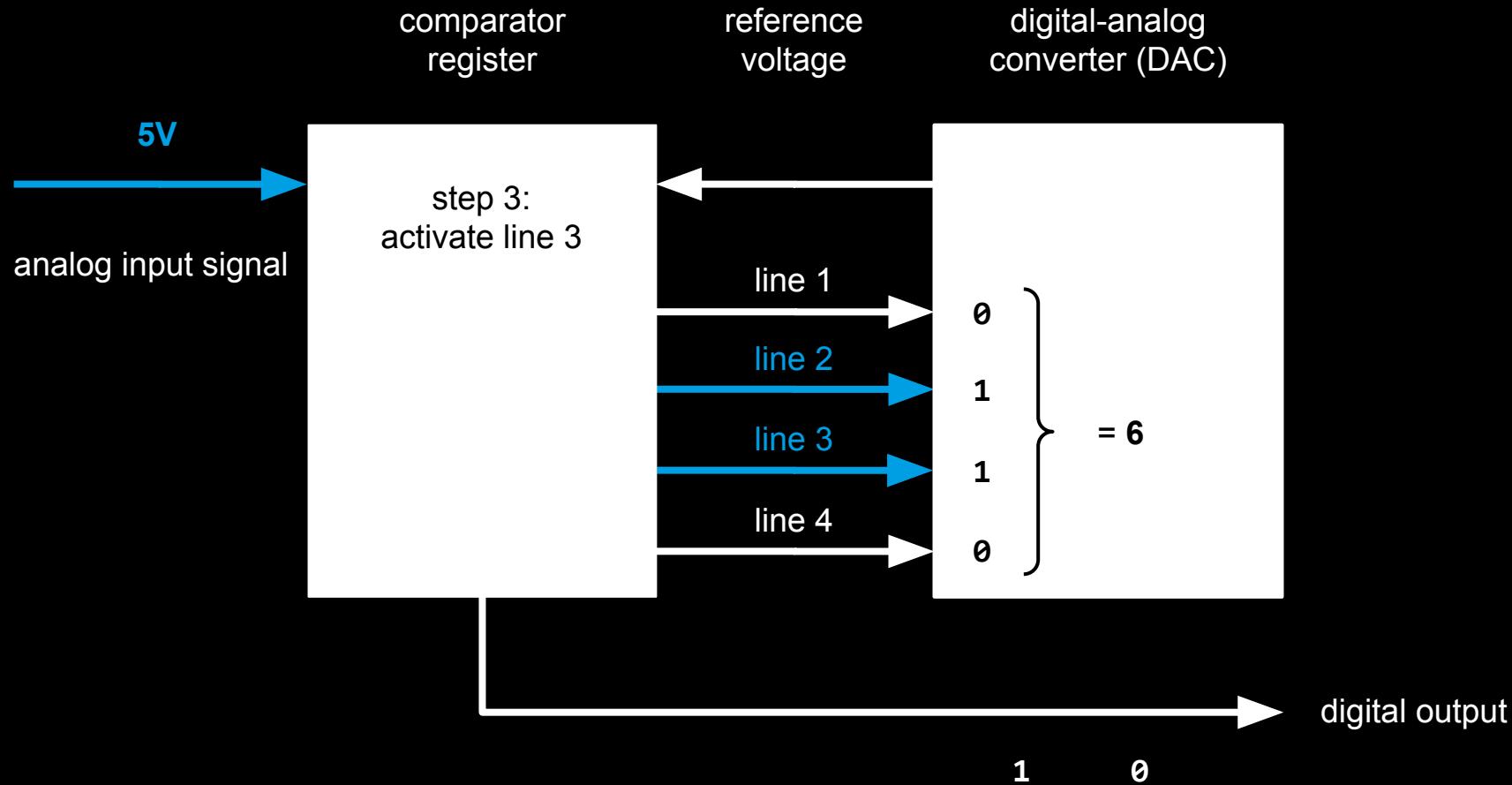


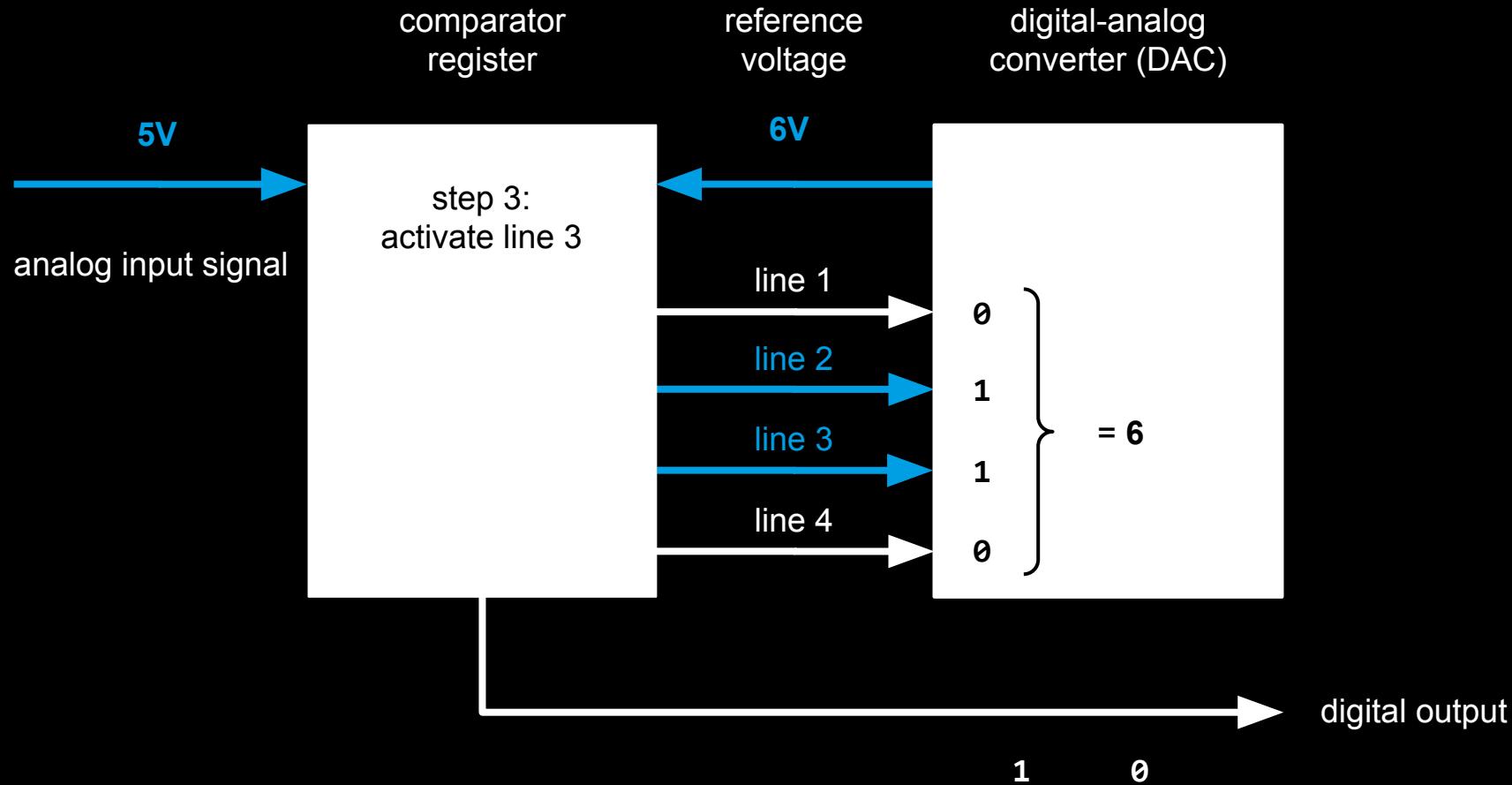


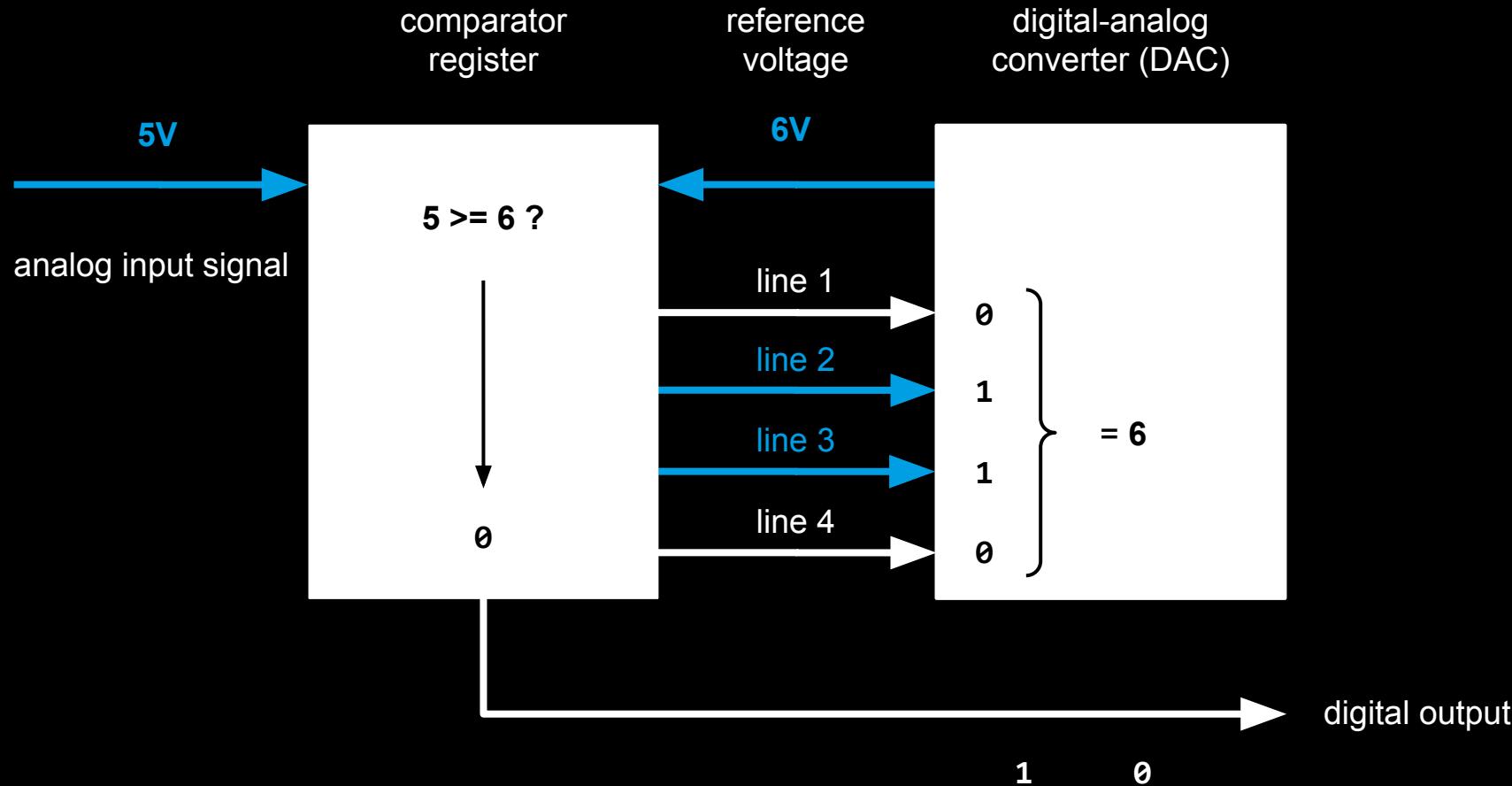


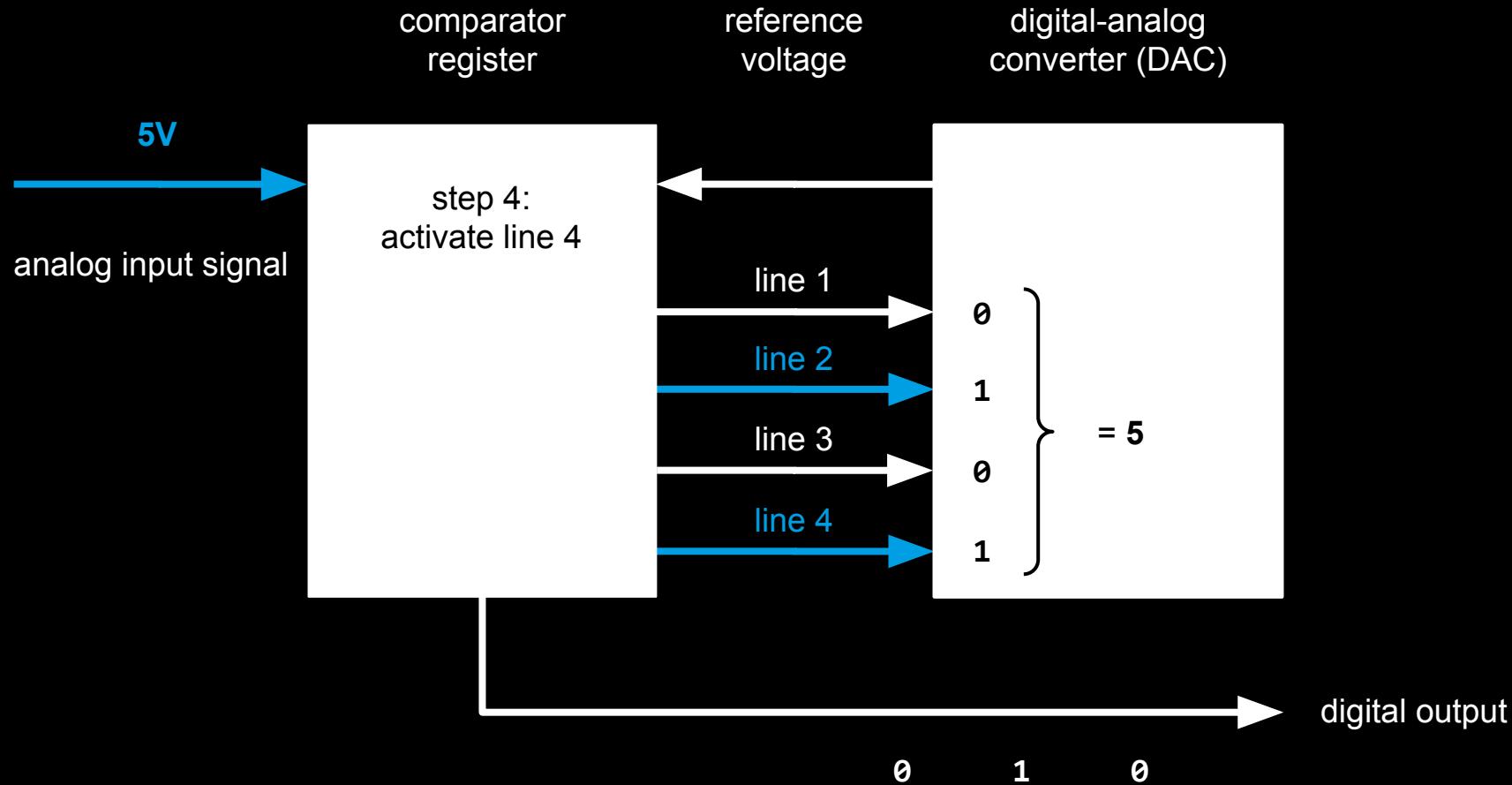


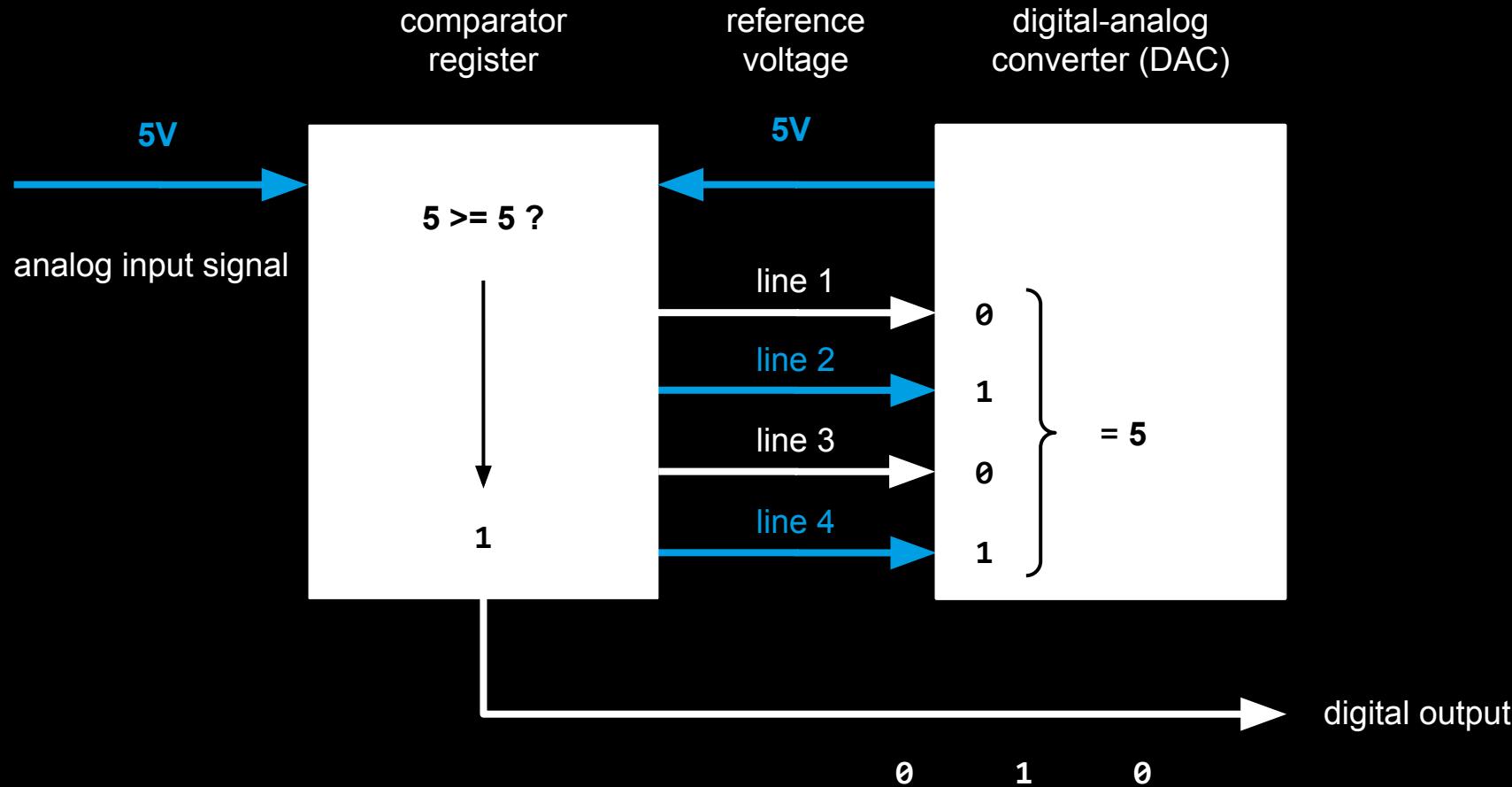


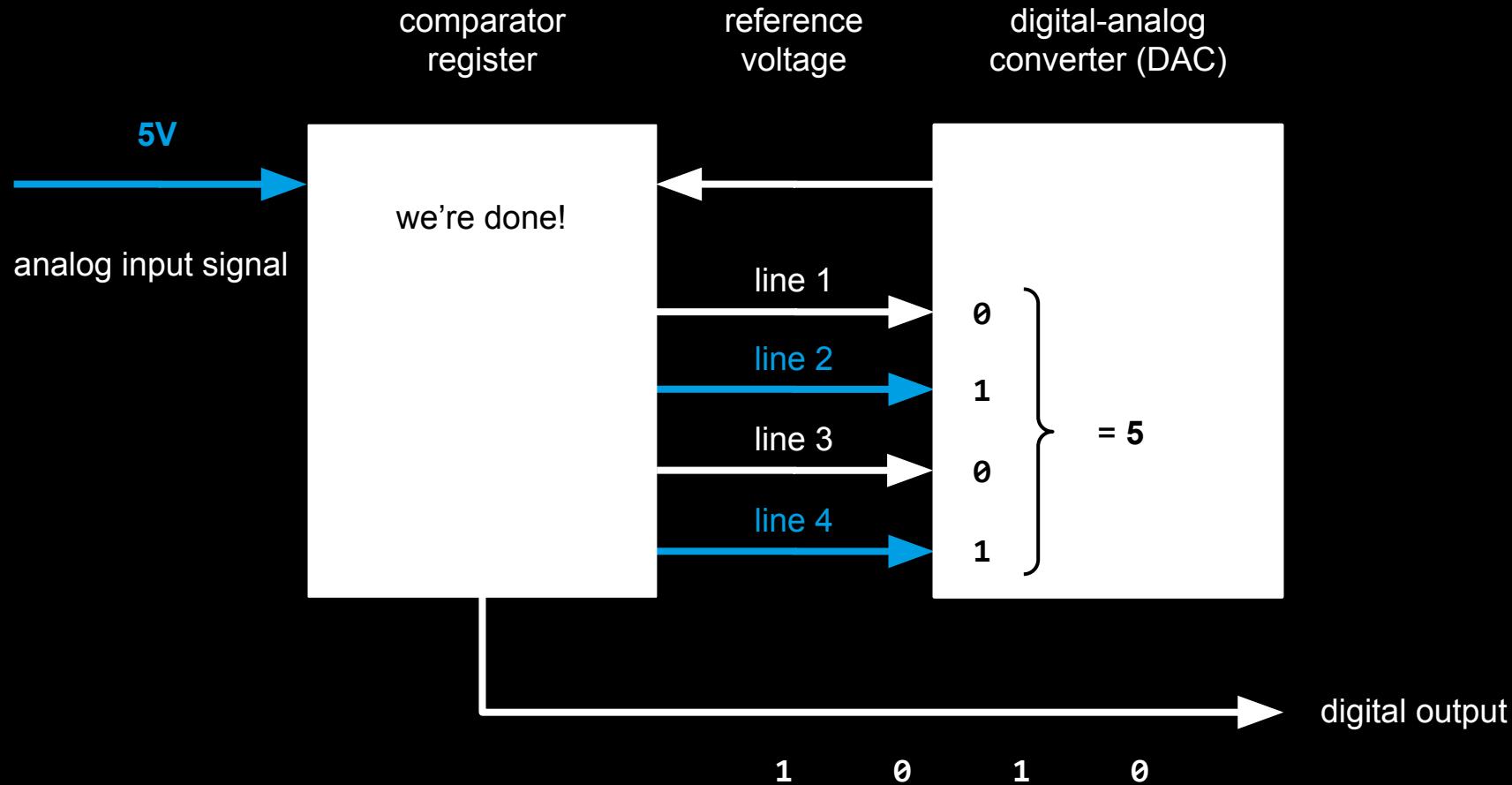




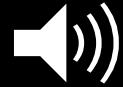


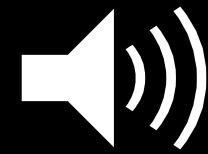






so, what analog signal is measured?





A **continuous** wave is transformed into an array of **discrete numbers** representing pressure.

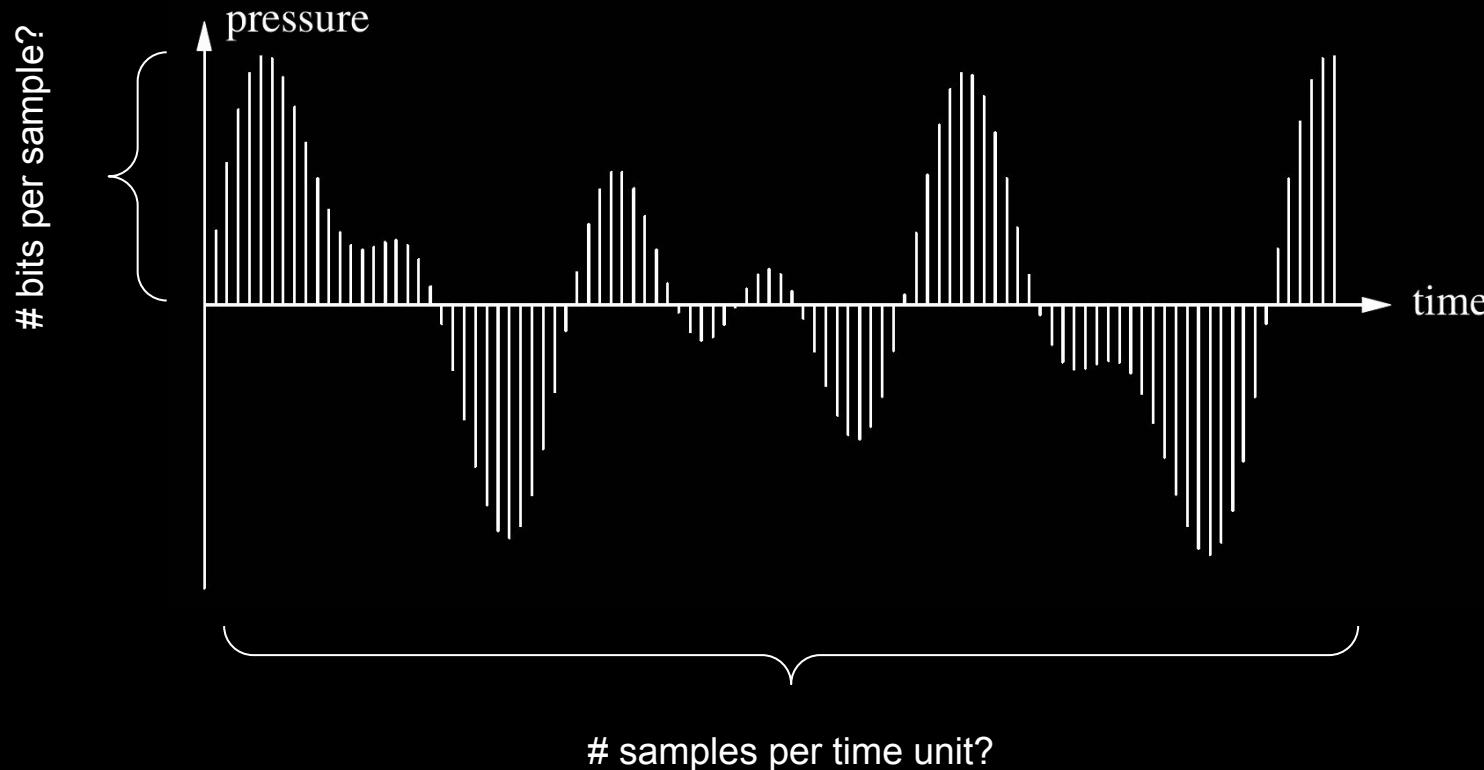




image sensor

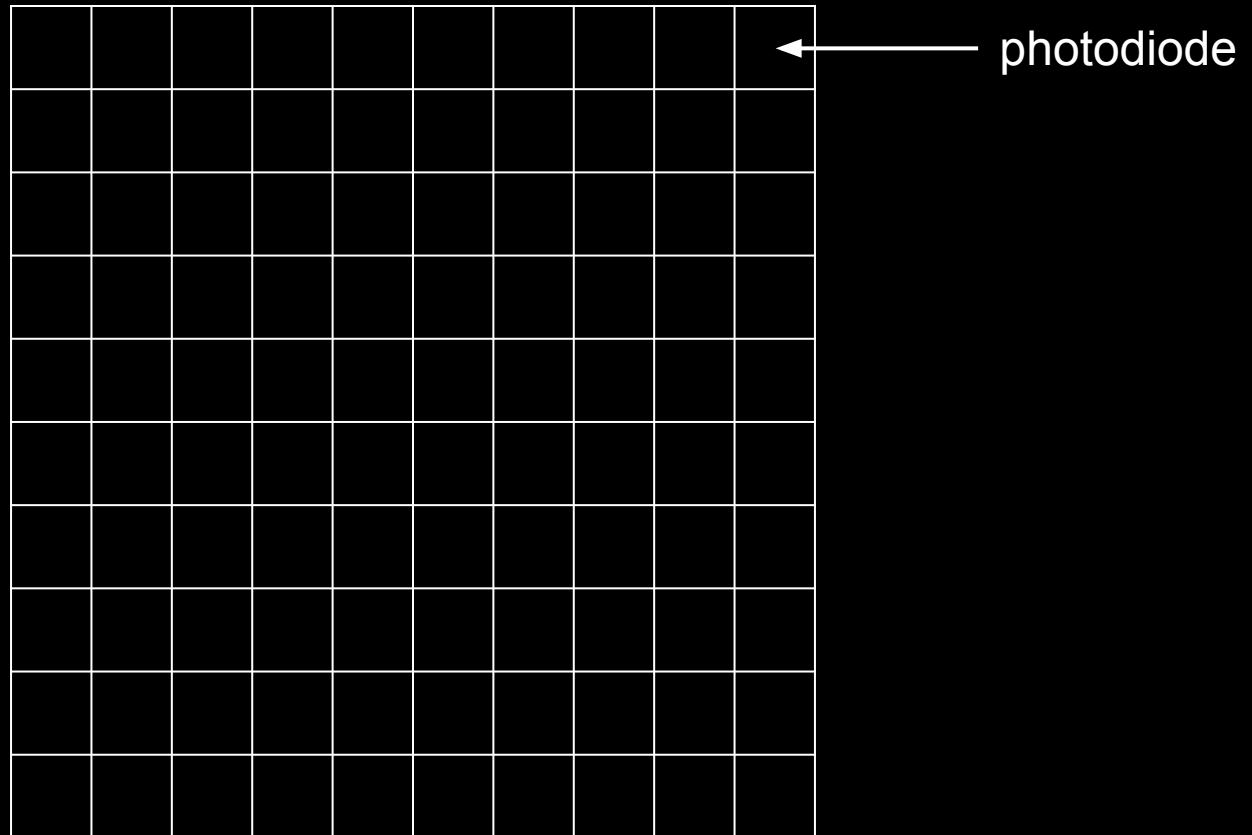
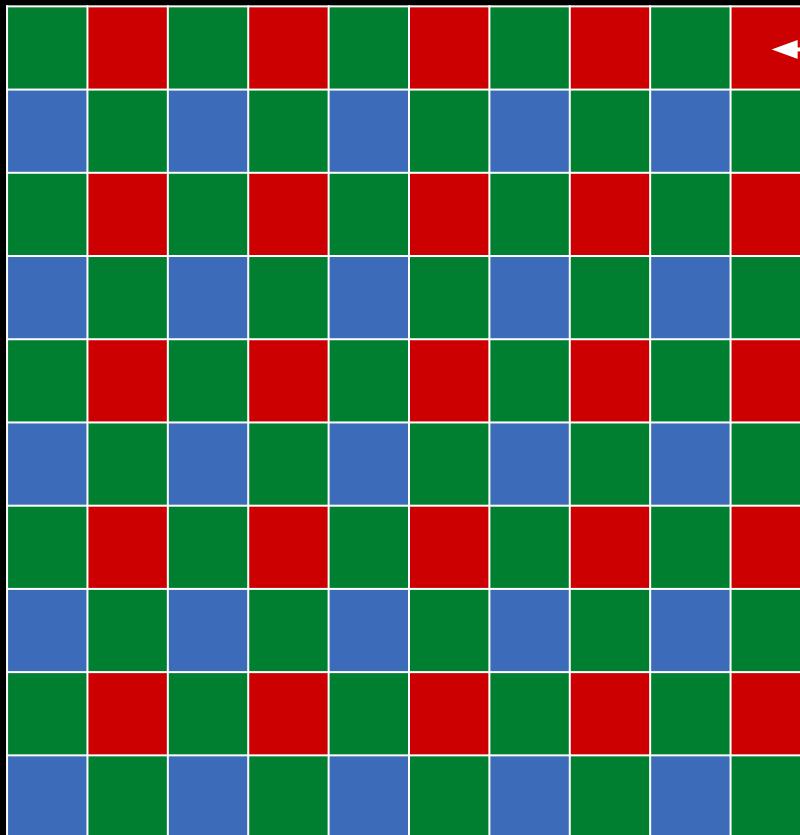
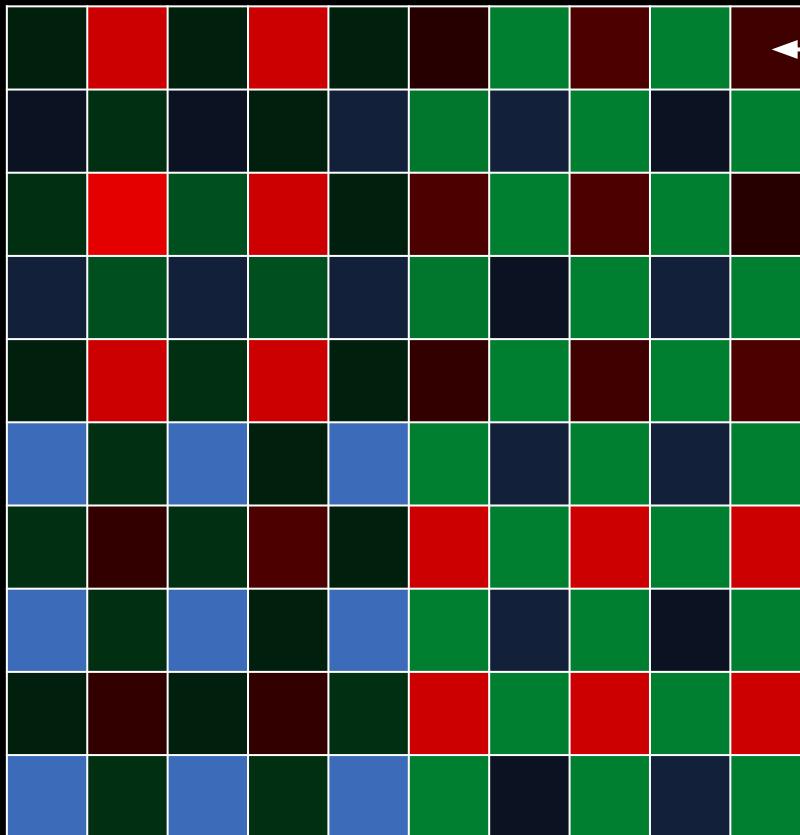


image sensor



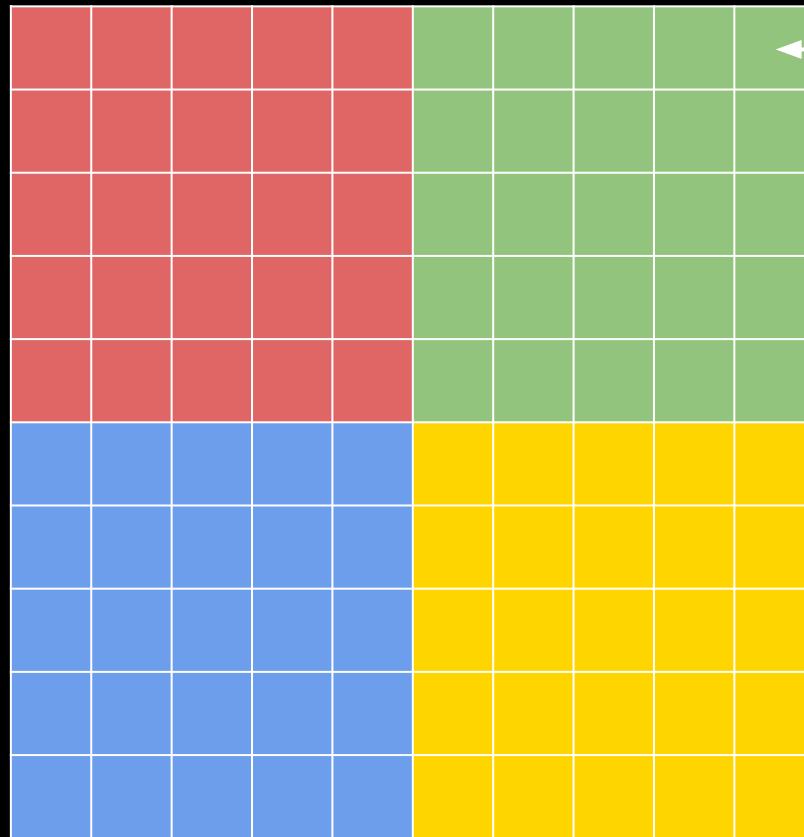
color filter on
top of photo-
diodes

image sensor



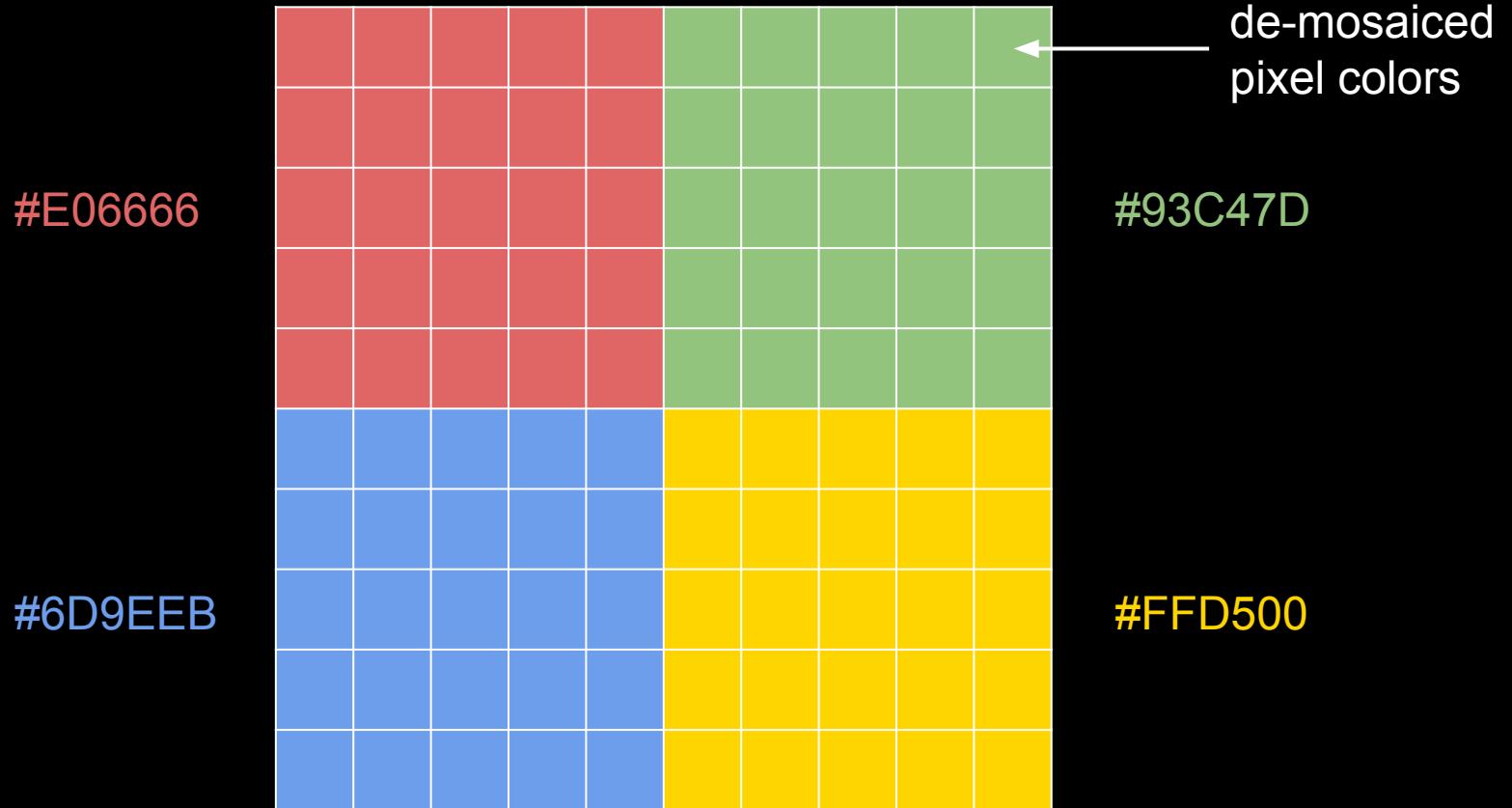
light passed
through filter

image sensor



de-mosaiced
pixel colors

image sensor

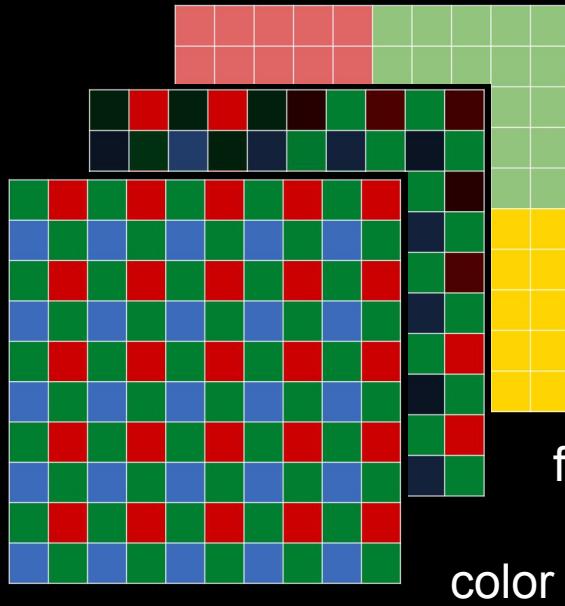


#FFD500

#6D9EEB

#93C47D

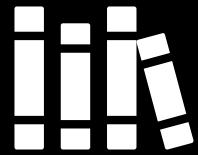
#E06666



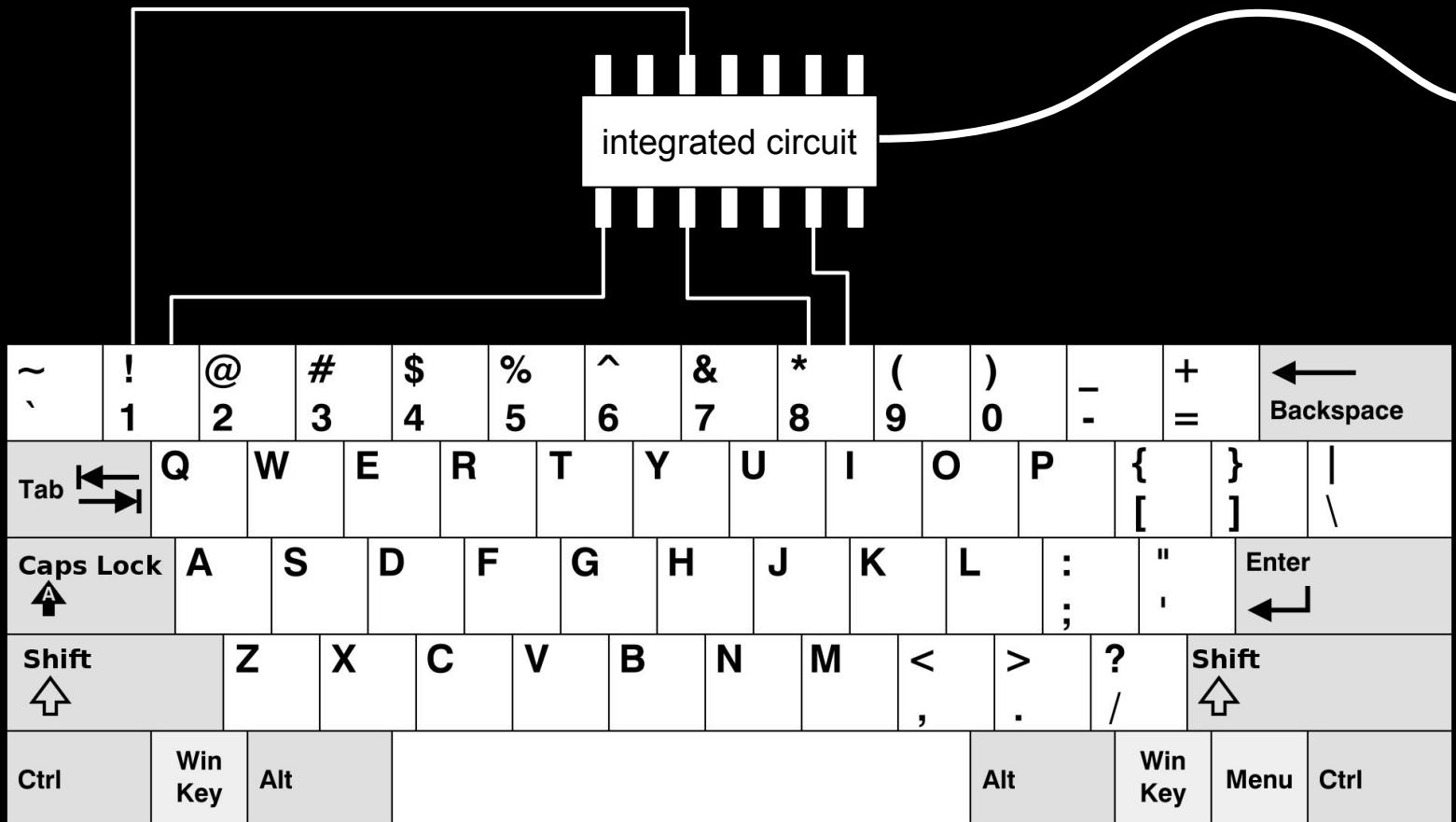
de-mosaiced pixel colors

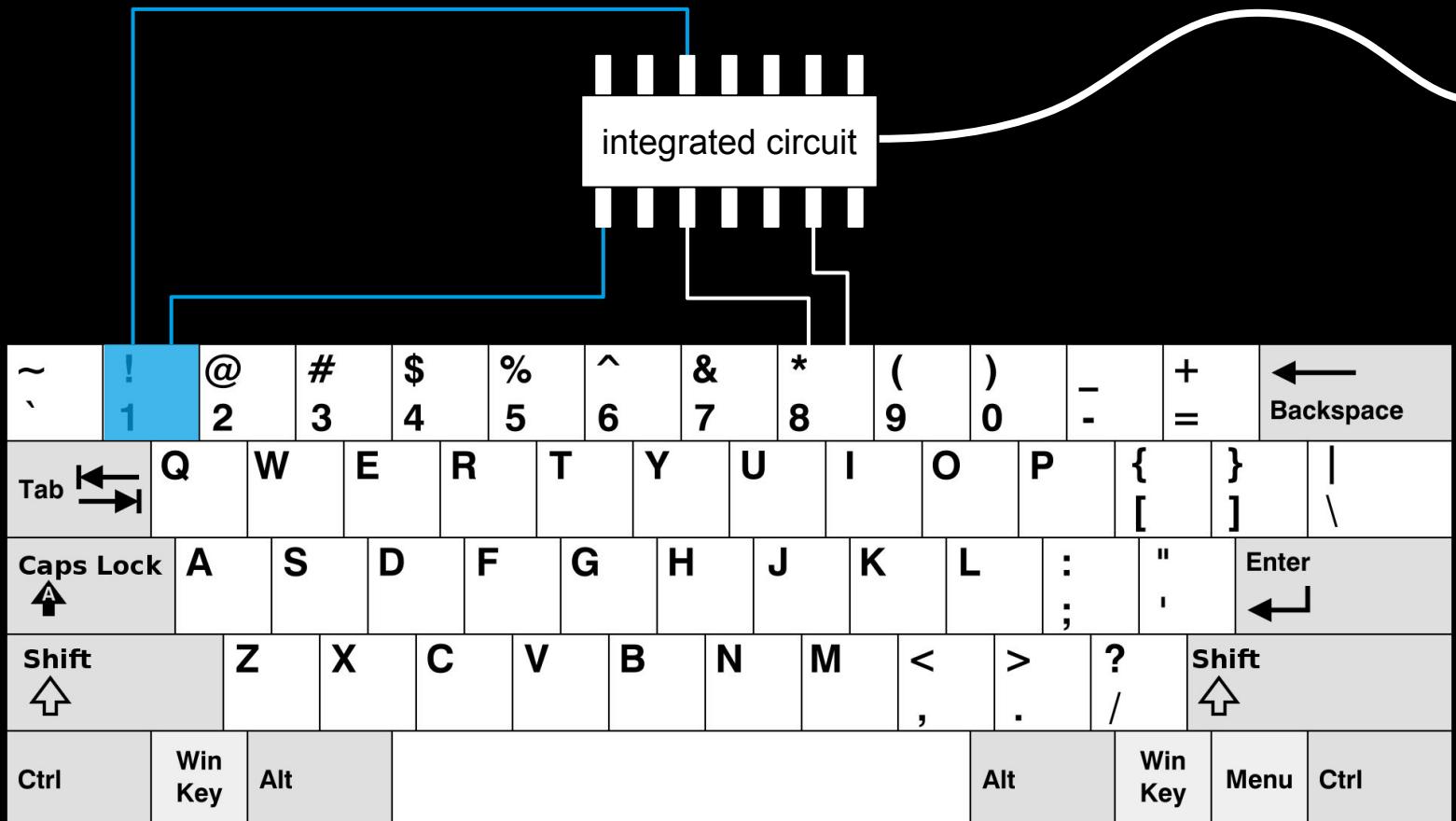
filtered light

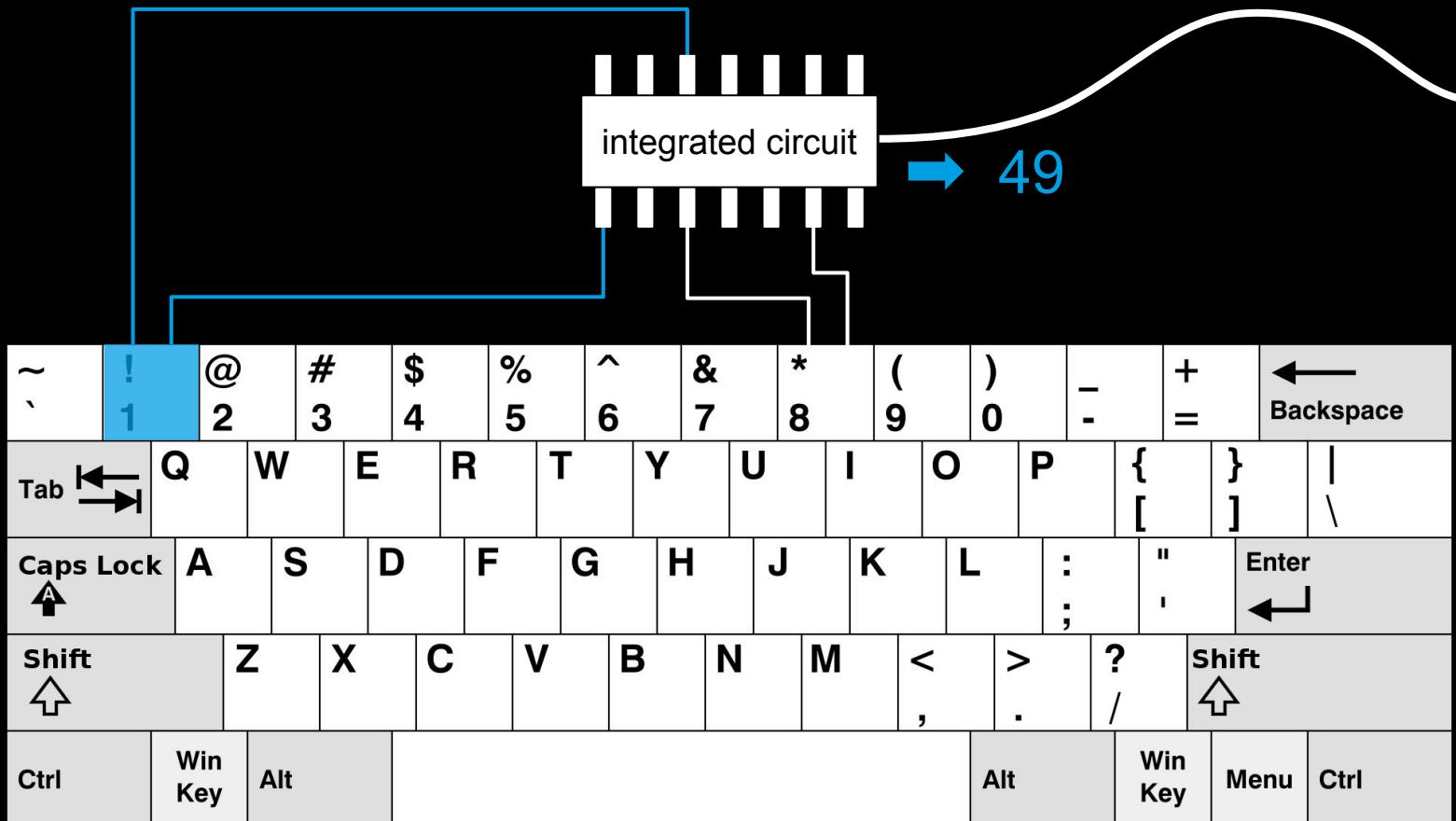
color filter

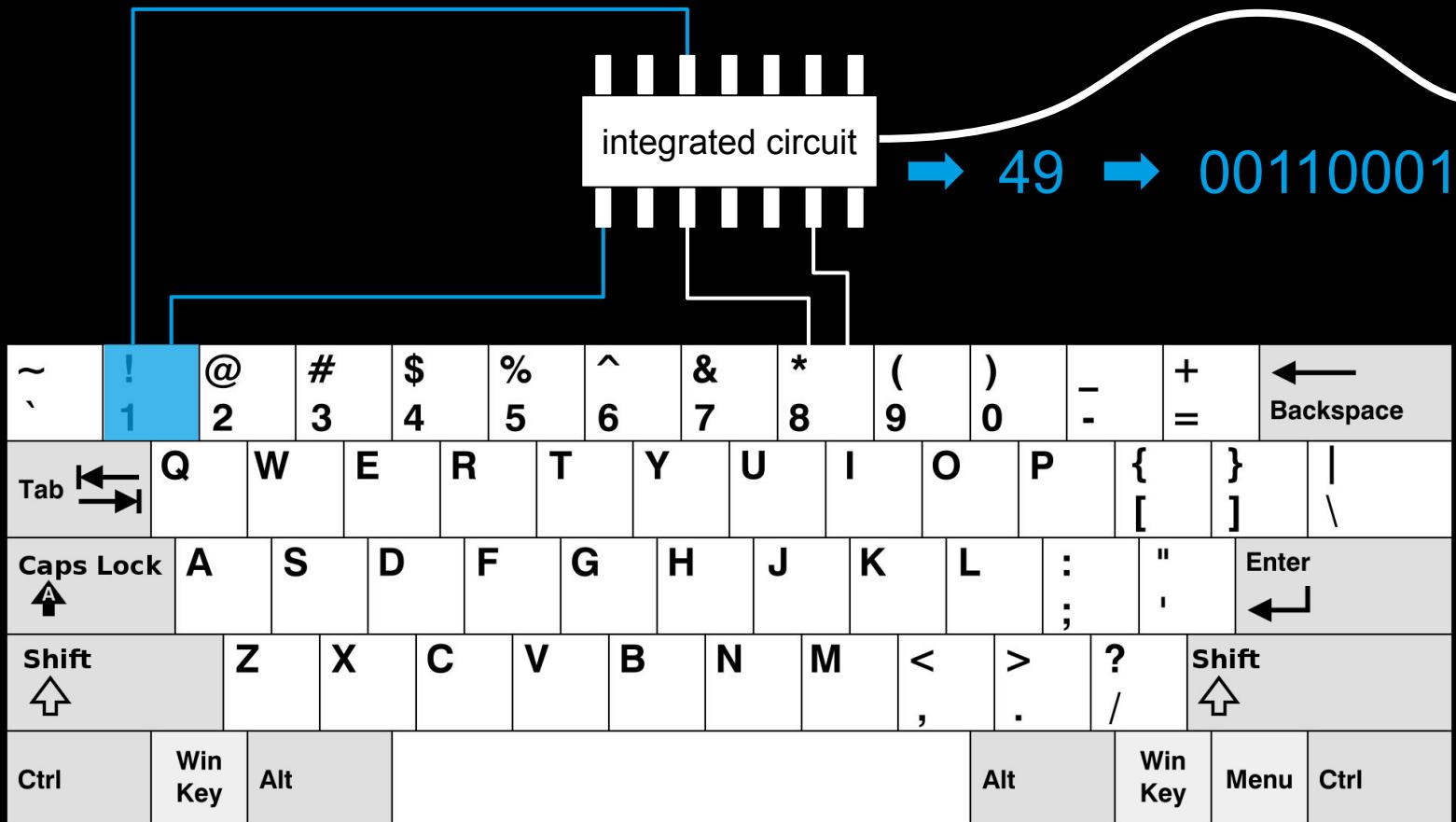


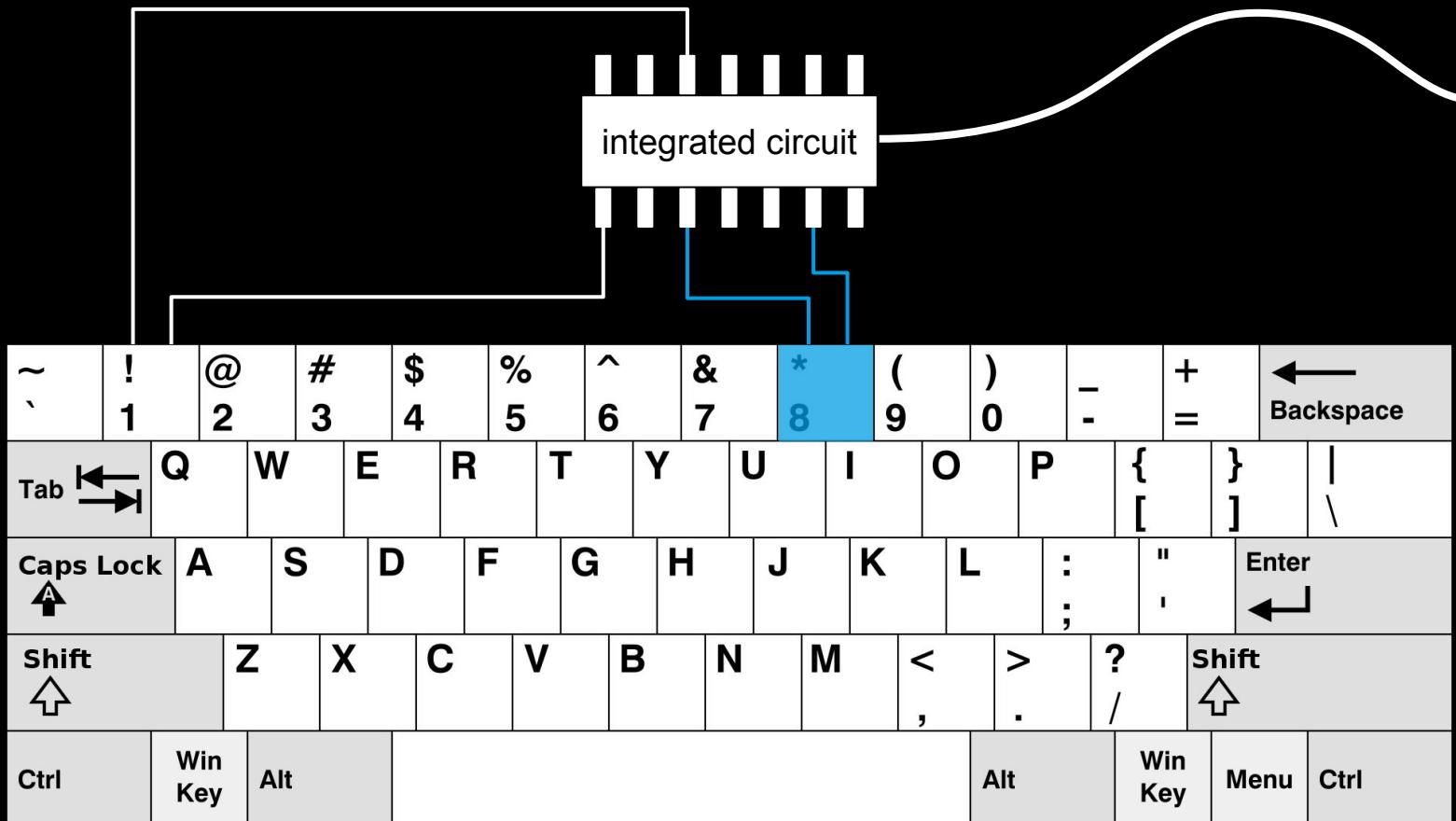
typing text

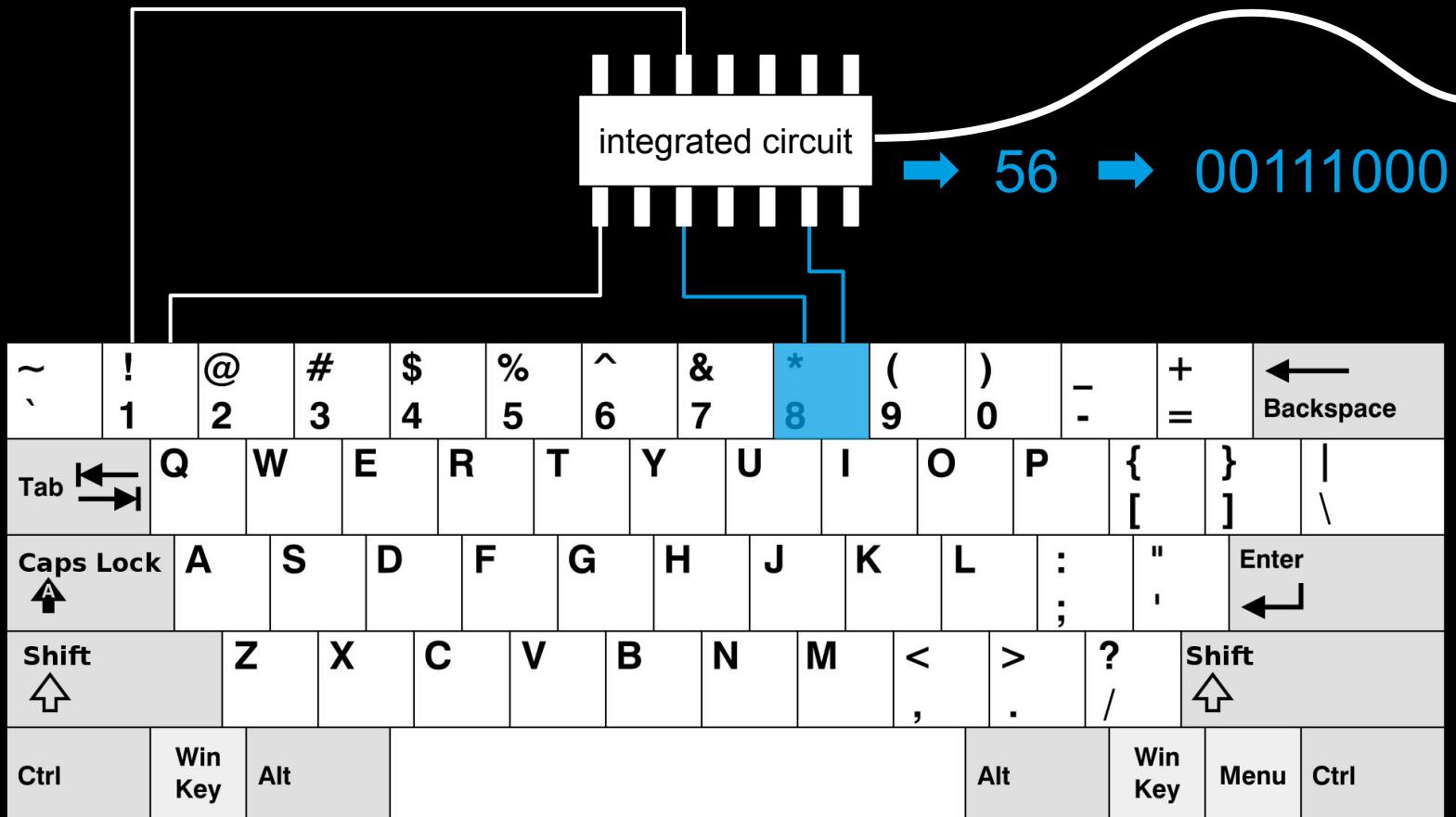










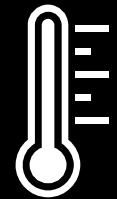


scanning text



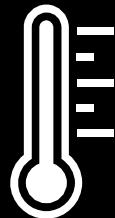
+

optical character recognition (OCR)



temperature sensor

A thermometer measures the temperature using a proxy, such as electrical resistance. The result is a change in voltage, which can be converted into a temperature.



1100 0101 1101 1110
1001 0110 1111 0100
0000 0111 1111 0001

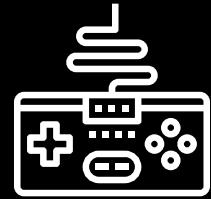




Image source: [Wikipedia](#)



Image source: [Wikipedia](#)

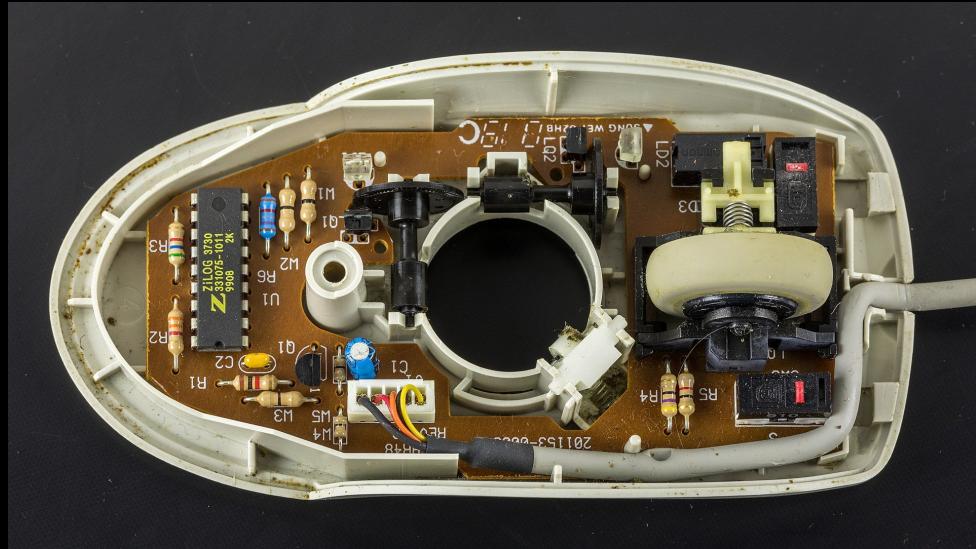


Image source: [Wikipedia](#)



Image source: [Wikipedia](#)

digital goods



Image source: [Wikipedia](#)



Image source: [Wikipedia](#)

computer processable

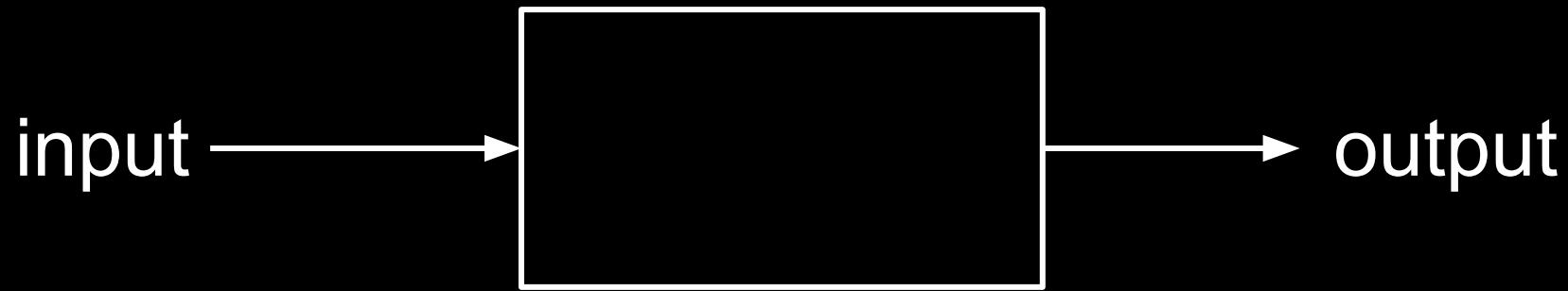
perfect reproduction

non-rival

near-zero costs of reproduction

ALGORITHMS

who solves the problem?





algorithm, program, process

"A finitely long rule consisting of individual instructions is called an **algorithm**."

Source: Vornberger, O., Algorithmen und Datenstrukturen, Lecture notes: <http://www-lehre.inf.uos.de/~ainf/2013/PDF/skript.pdf>

"A **program** is an algorithm expressed in a
programming language."

Source: Vornberger, O., Algorithmen und Datenstrukturen, Lecture notes: <http://www-lehre.inf.uos.de/~ainf/2013/PDF/skript.pdf>

"A process is a program that is currently executed by a computer."

Source: Vornberger, O., Algorithmen und Datenstrukturen, Lecture notes: <http://www-lehre.inf.uos.de/~ainf/2013/PDF/skript.pdf>



greatest common divisor

euclidean algorithm

Identify the larger number of a and b. If $a < b$, swap numbers so that $a > b$

Subtract b from a and replace a with the result

Repeat until one of the numbers becomes 0

Return the number that is not zero

Loop 1:

a = 18, b = 48 → swap

Loop 1:

a = 18, b = 48 → swap → a = 48, b = 18
a = 48 - 18 = 30

Loop 1:

a = 18, b = 48 → swap → a = 48, b = 18
a = 48 - 18 = 30

Loop 2:

a = 30, b = 18 → no swap
a = 30 - 18 = 12

Loop 1:

a = 18, b = 48 → swap → a = 48, b = 18
a = 48 - 18 = 30

Loop 2:

a = 30, b = 18 → no swap
a = 30 - 18 = 12

Loop 3:

a = 12, b = 18 → swap → a = 18, b = 12
a = 18 - 12 = 6

Loop 1:

a = 18, b = 48 → swap → a = 48, b = 18
a = 48 - 18 = 30

Loop 2:

a = 30, b = 18 → no swap
a = 30 - 18 = 12

Loop 3:

a = 12, b = 18 → swap → a = 18, b = 12
a = 18 - 12 = 6

Loop 4:

a = 6, b = 12 → swap → a = 12, b = 6
a = 12 - 6 = 6

Loop 1:

a = 18, b = 48 → swap → a = 48, b = 18
a = 48 - 18 = 30

Loop 2:

a = 30, b = 18 → no swap
a = 30 - 18 = 12

Loop 3:

a = 12, b = 18 → swap → a = 18, b = 12
a = 18 - 12 = 6

Loop 4:

a = 6, b = 12 → swap → a = 12, b = 6
a = 12 - 6 = 6

Loop 5:

a = 6, b = 6 → no swap
a = 6 - 6 = 0

Loop 1:

a = 18, b = 48 → swap → a = 48, b = 18
a = 48 - 18 = 30

Loop 2:

a = 30, b = 18 → no swap
a = 30 - 18 = 12

Loop 3:

a = 12, b = 18 → swap → a = 18, b = 12
a = 18 - 12 = 6

Loop 4:

a = 6, b = 12 → swap → a = 12, b = 6
a = 12 - 6 = 6

Loop 5:

a = 6, b = 6 → no swap
a = 6 - 6 = 0

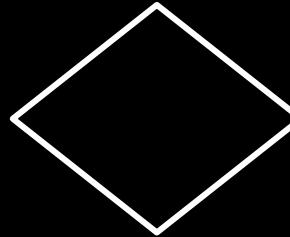
return b = 6

flow diagrams

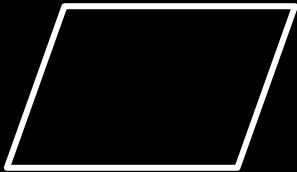


BEGIN / END

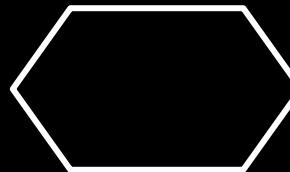
start / end of
algorithm



decision



input / output



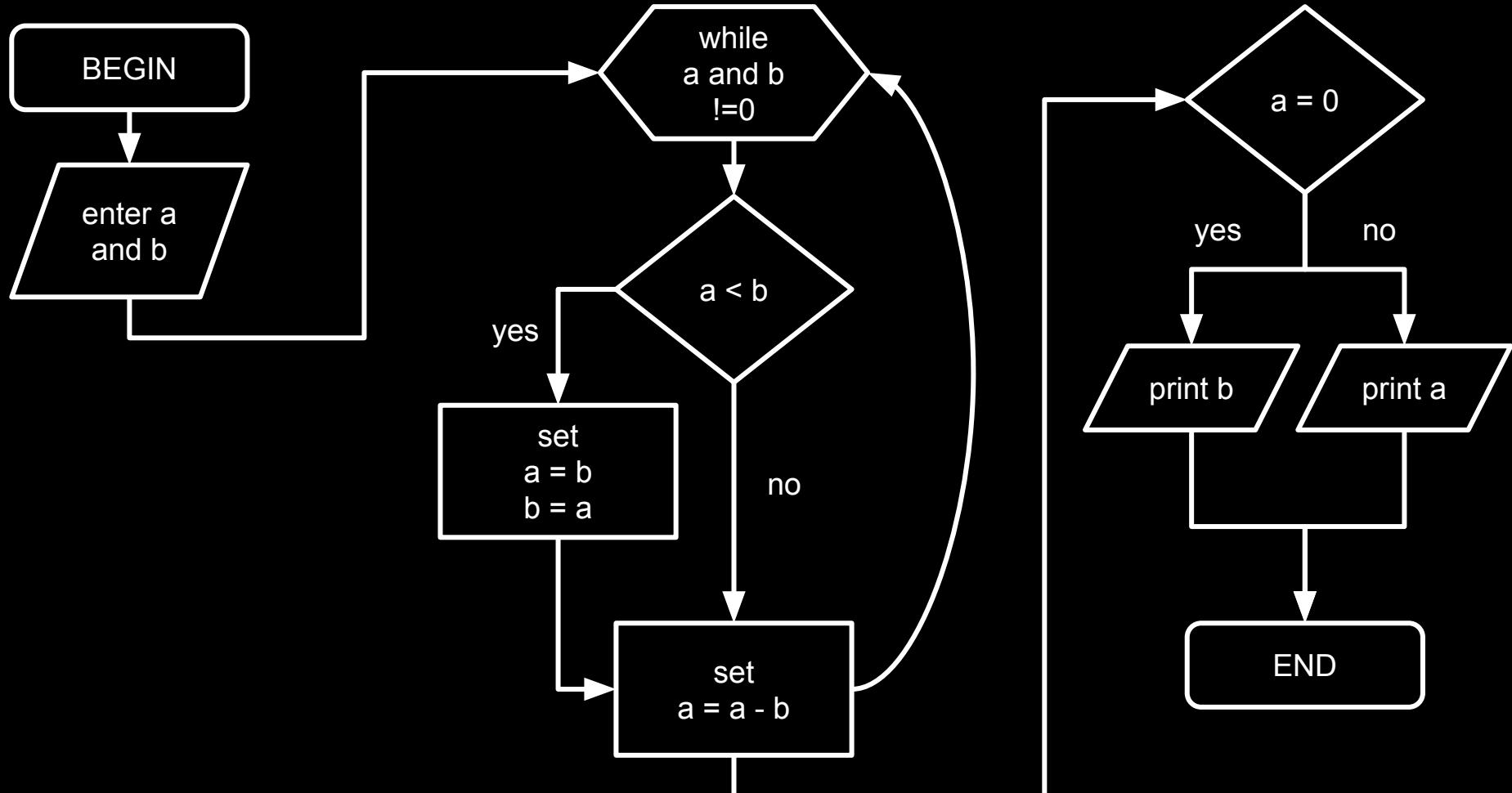
repetition



command /
assignment



external routine





square roots

babylonian method

calculate square root of
 $x = 16$

A = 1

B = X / A = 16

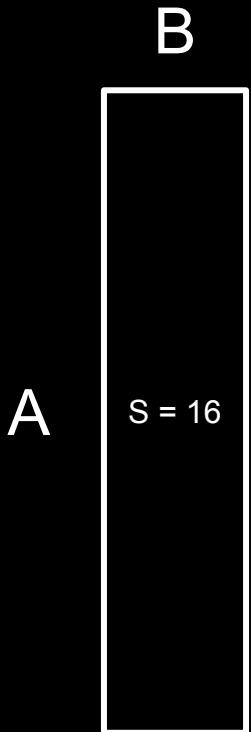
B

A

S = 16

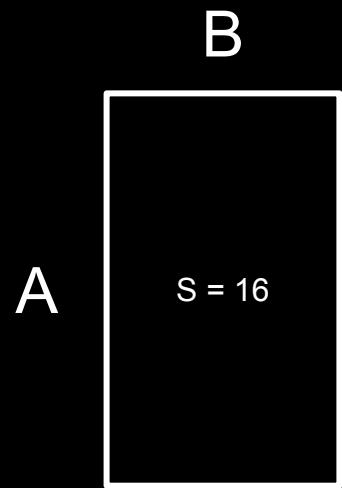
$$A = (A + B) / 2 = 17 / 2 = 8.5$$

$$B = X / A = 16 / 8.5 \approx 1.88$$



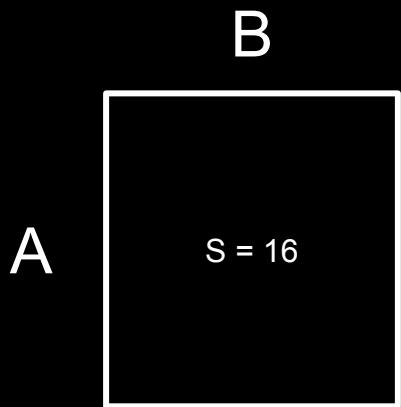
$$A = (A + B) / 2 \approx 10.38 / 2 \approx 5.19$$

$$B = X / A \approx 16 / 5.19 \approx 3.08$$



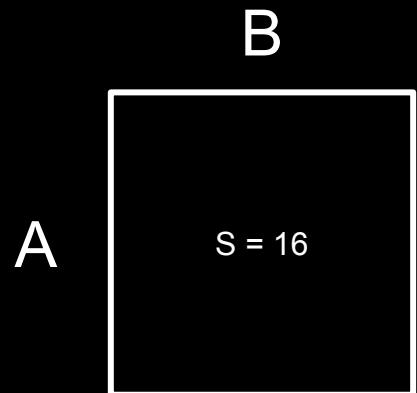
$$A = (A + B) / 2 \approx 8.27 / 2 \approx 4.14$$

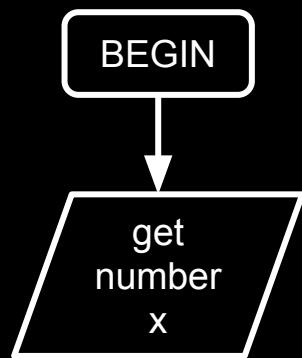
$$B = X / A \approx 16 / 4.14 \approx 3.86$$

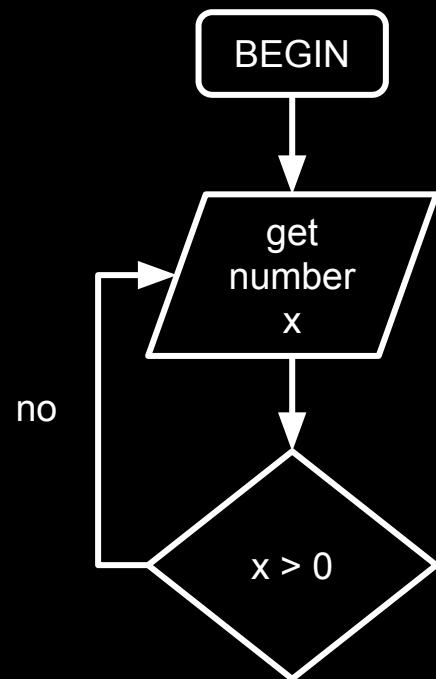


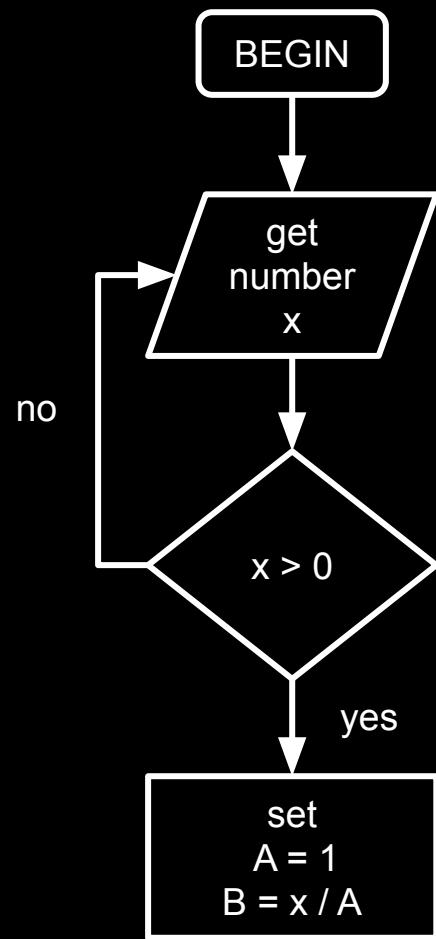
$$A = (A + B) / 2 = 8 / 2 = 4$$

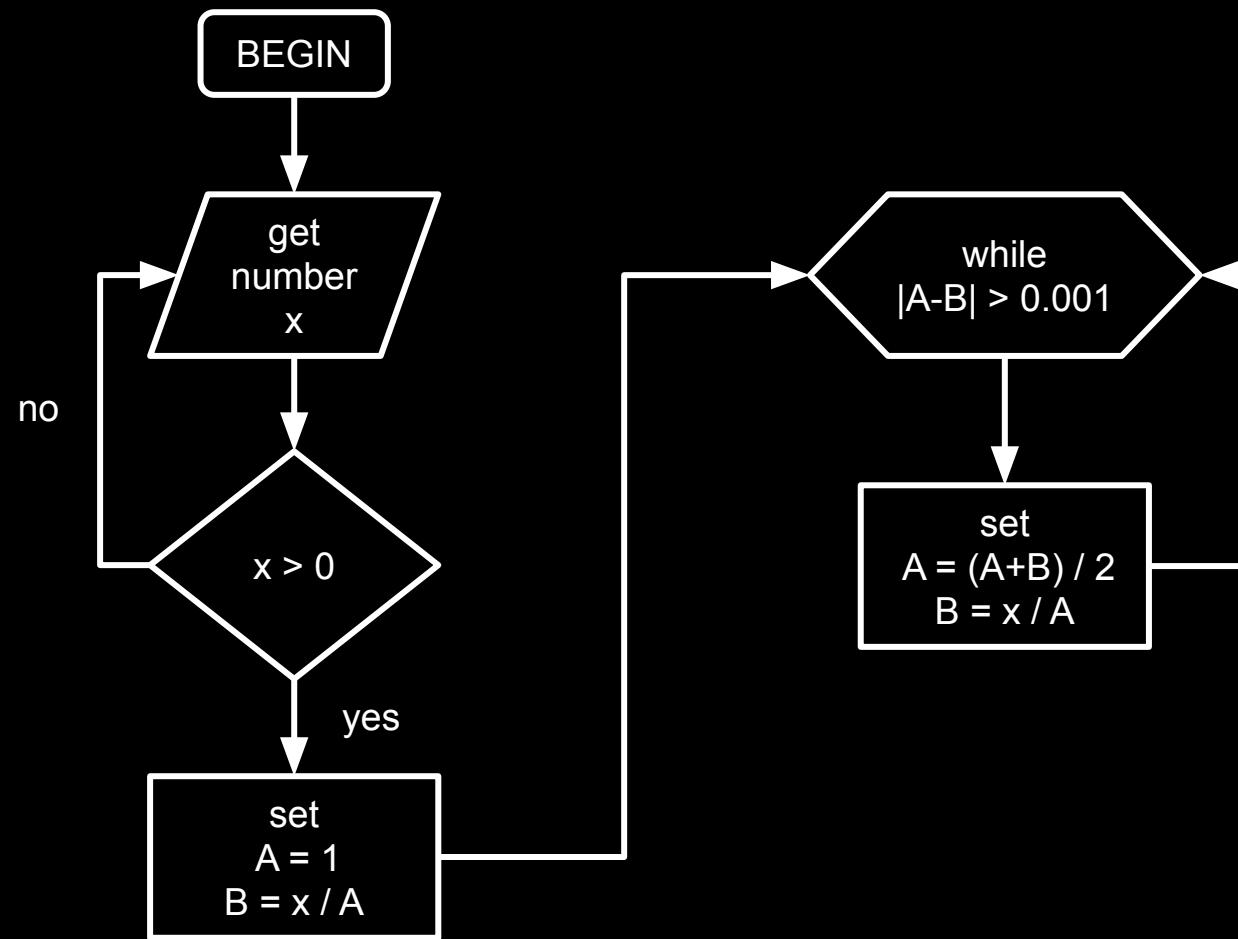
$$B = X / A = 16 / 4 = 4$$

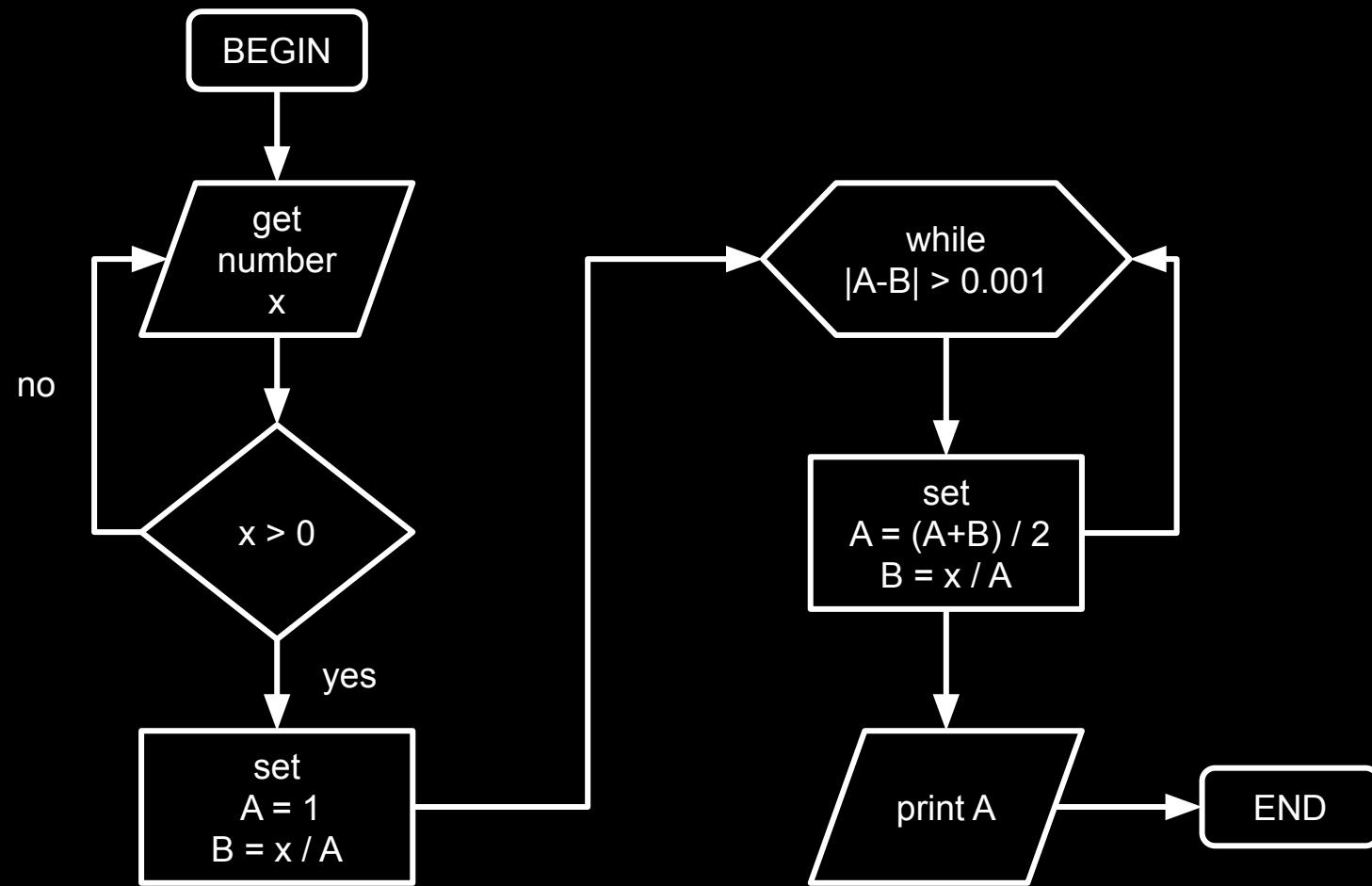






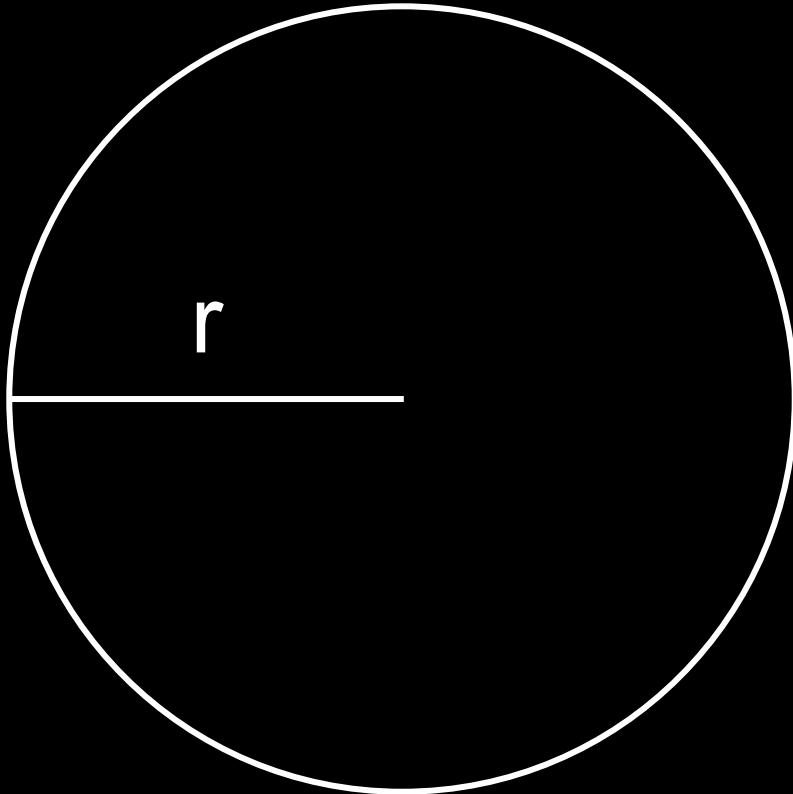


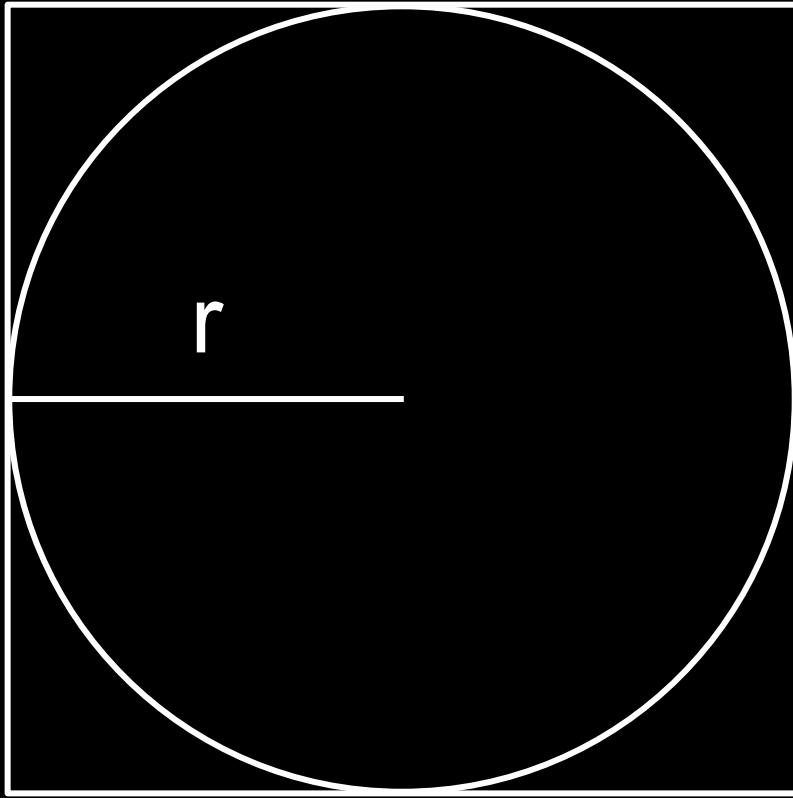


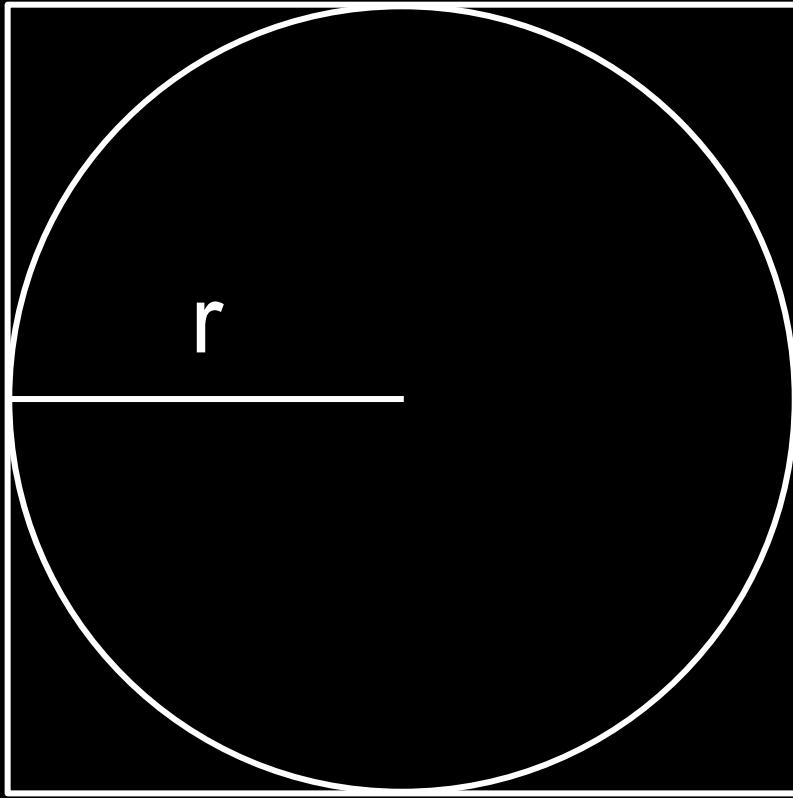


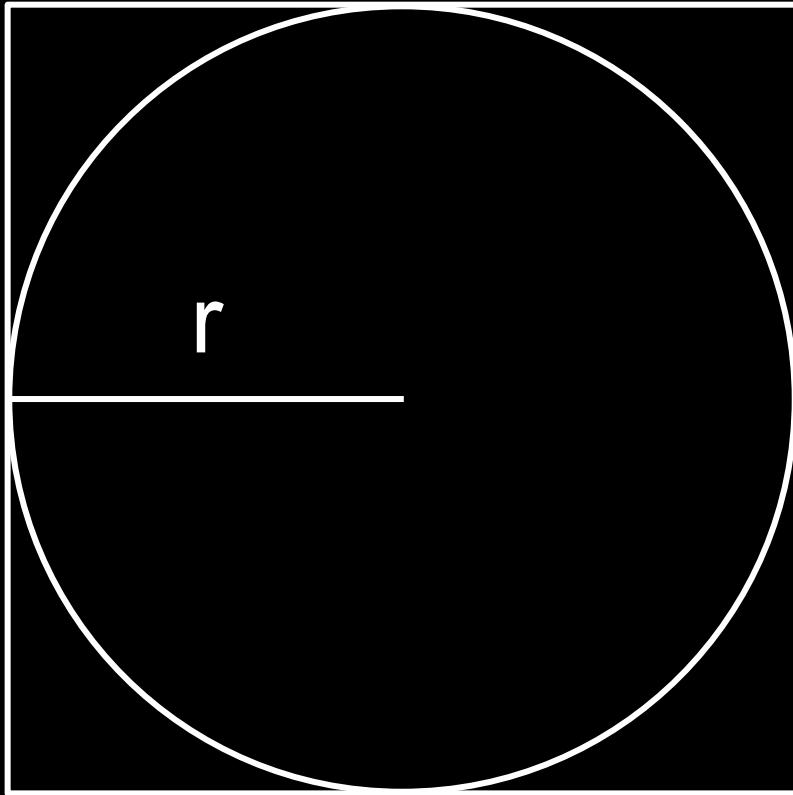


estimating π



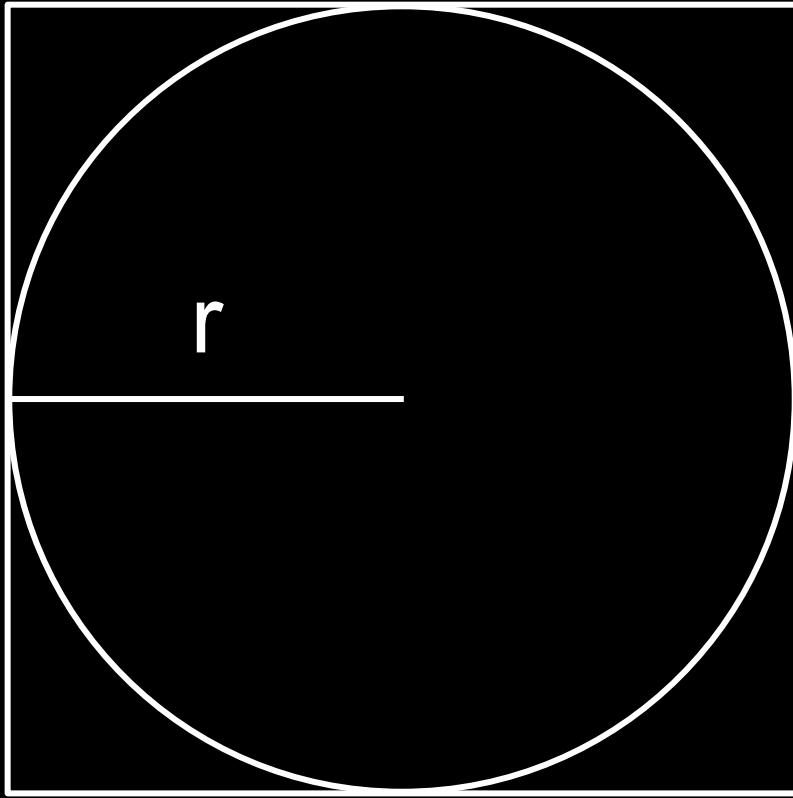


 $2r$ $2r$



$$\frac{\textcircled{1}}{\square} = \frac{\pi r^2}{4r^2}$$

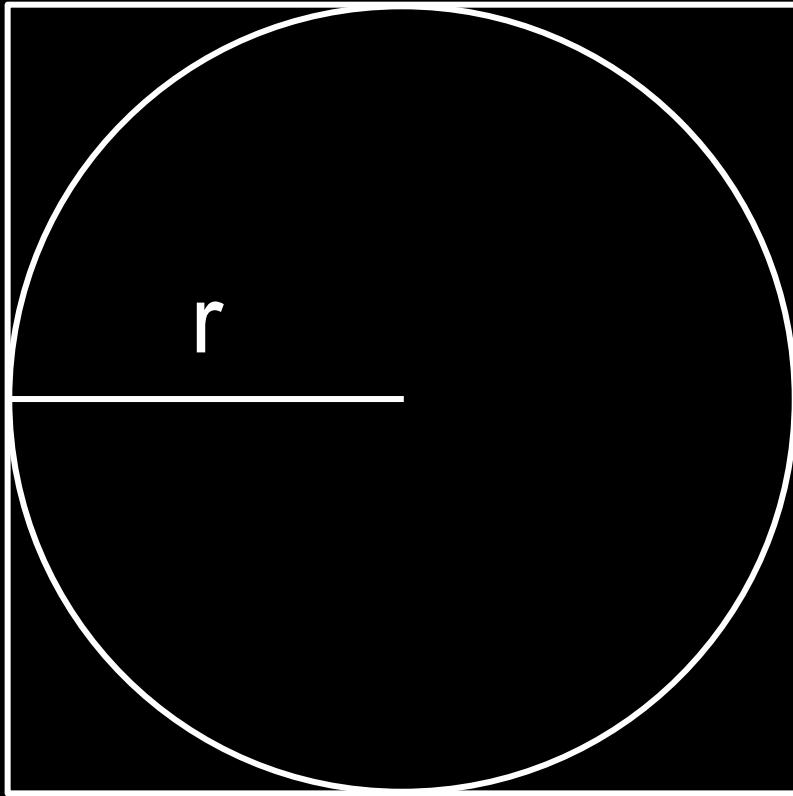
2r



$2r$

$$\frac{\textcircled{1}}{\square} = \frac{\pi r^2}{4r^2}$$

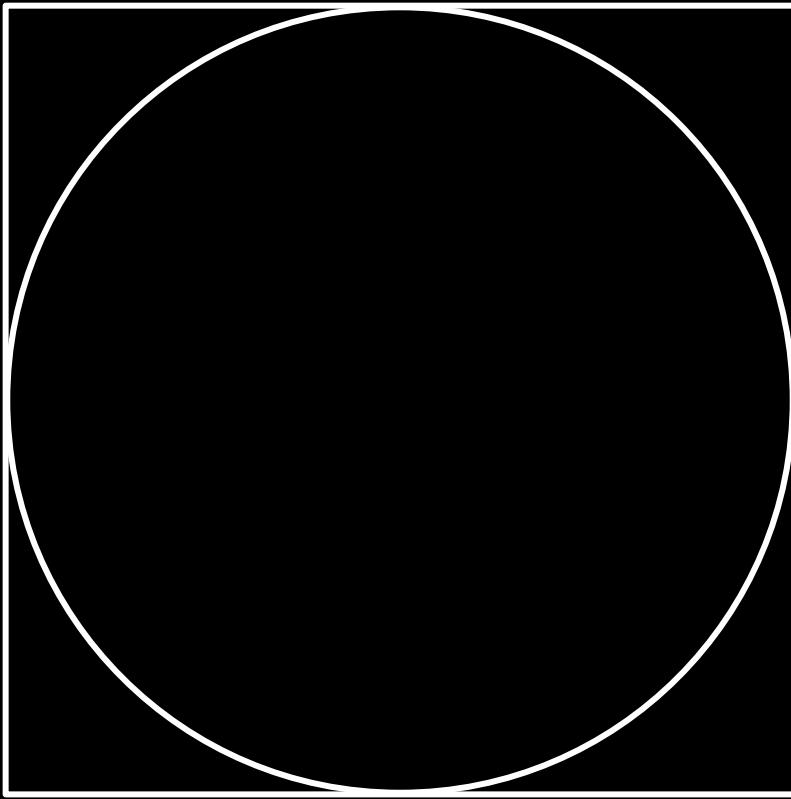
$2r$



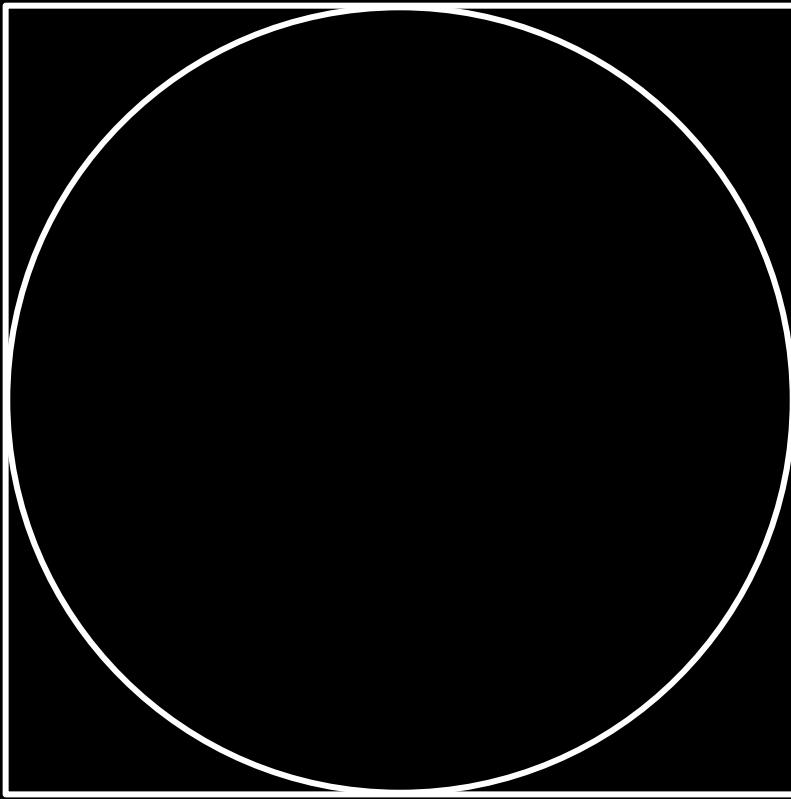
$$\frac{\textcircled{1}}{\square} = \frac{\pi}{4}$$

2r

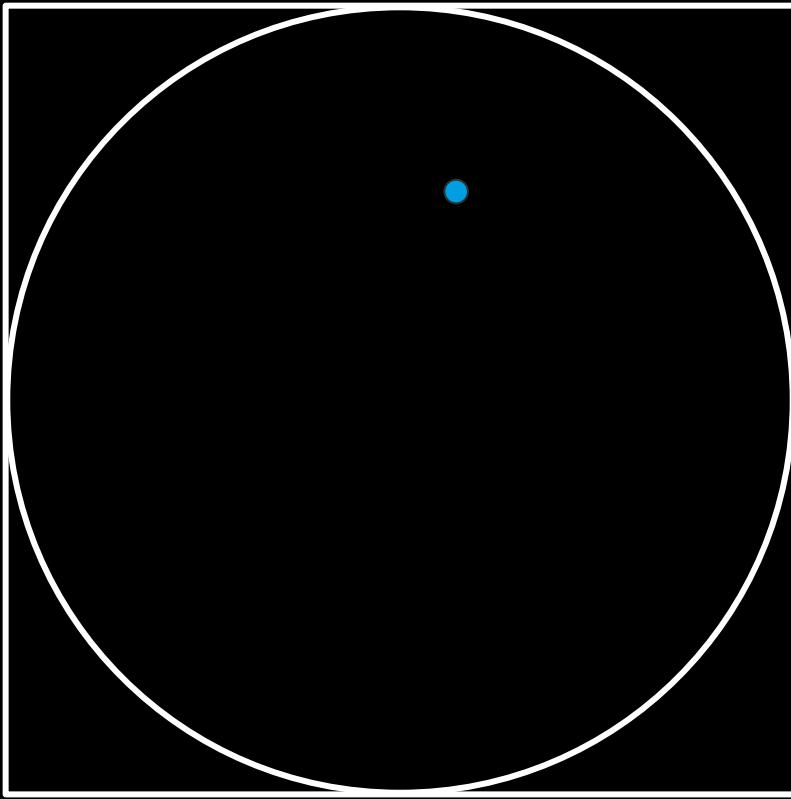
monte carlo simulation



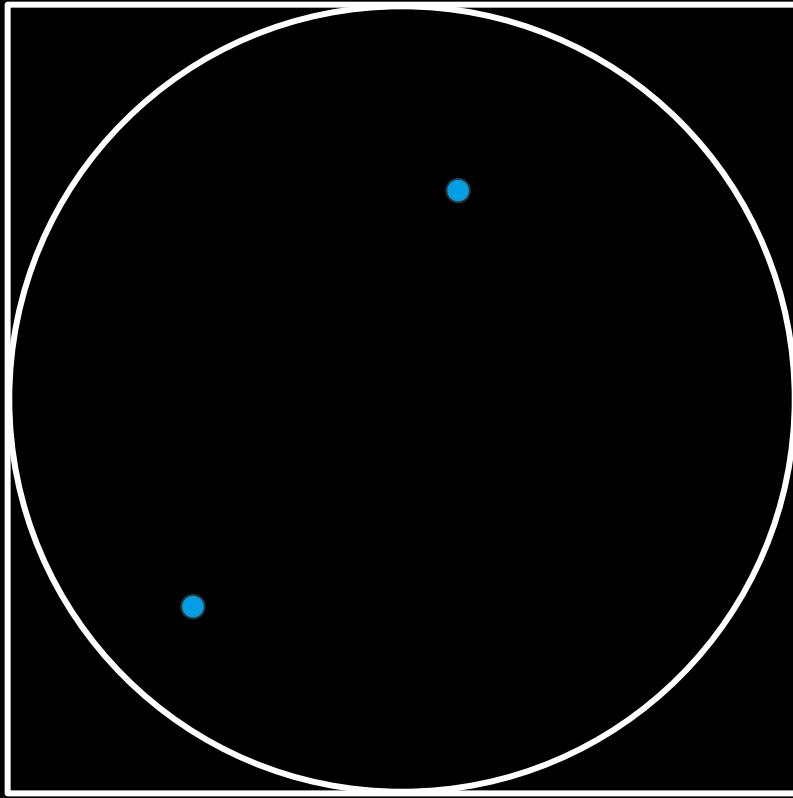
$$\frac{\textcircled{1}}{\square} = \frac{\pi}{4}$$



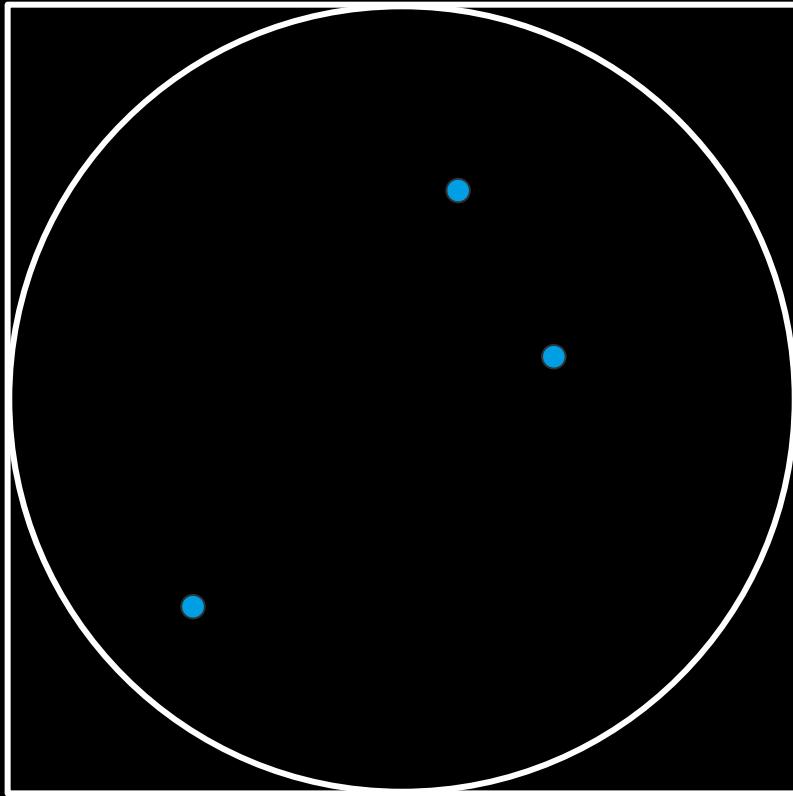
$$4 \frac{\circ}{\square} = \pi$$



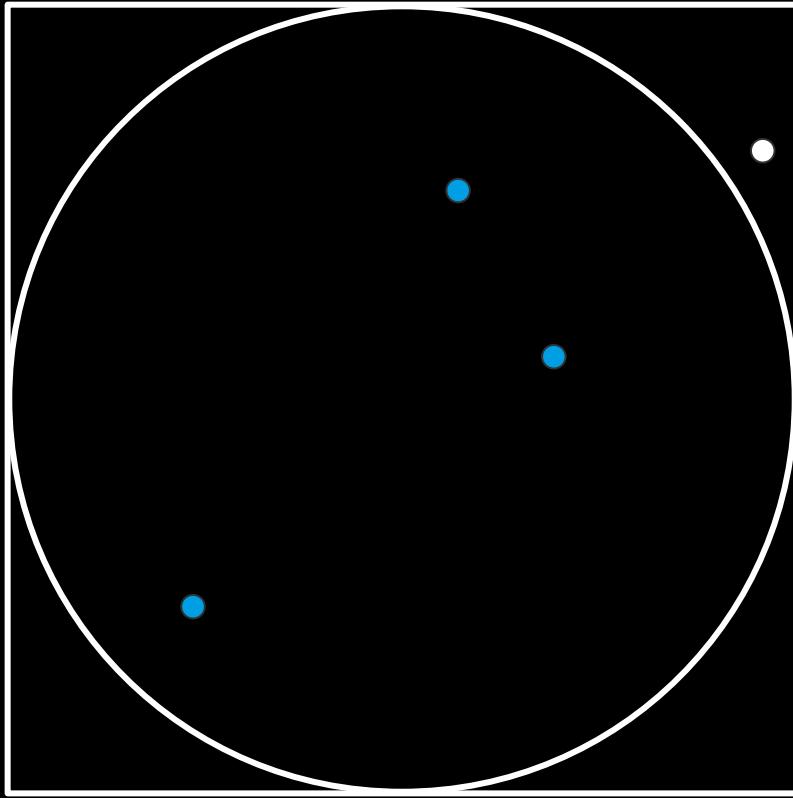
$$4 \frac{\bigcirc}{\square} = \pi$$
$$= 4$$



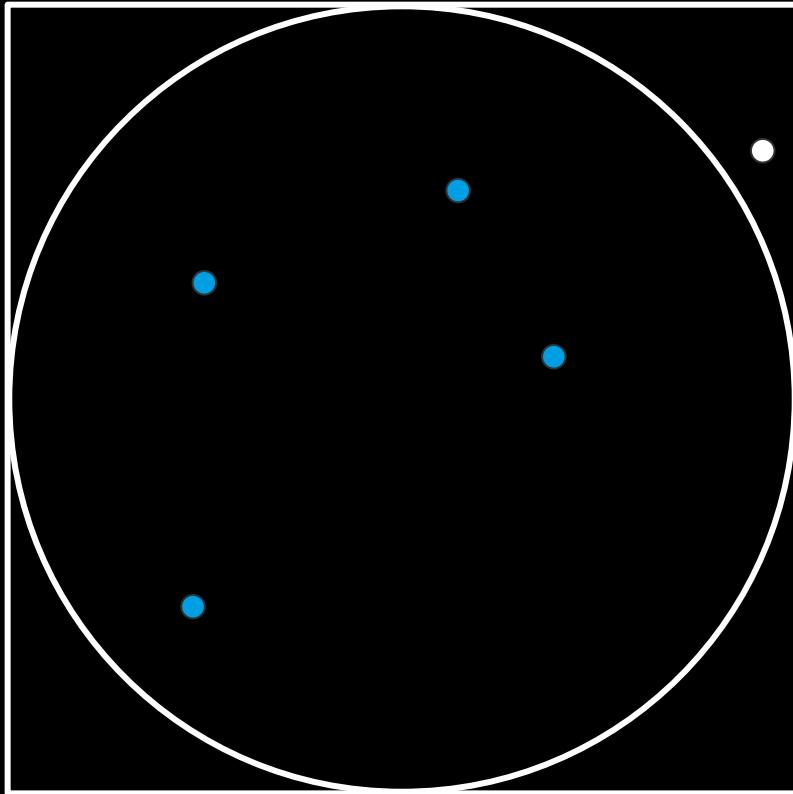
$$4 \frac{\bigcirc}{\square} = \pi$$
$$= 4$$



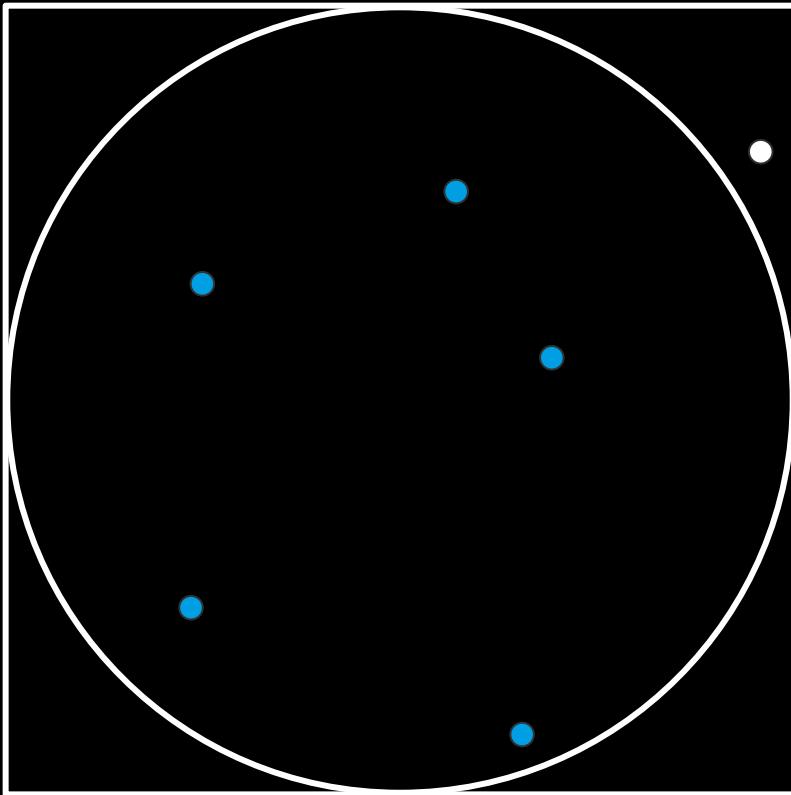
$$4 \frac{\bigcirc}{\square} = \pi$$
$$= 4$$



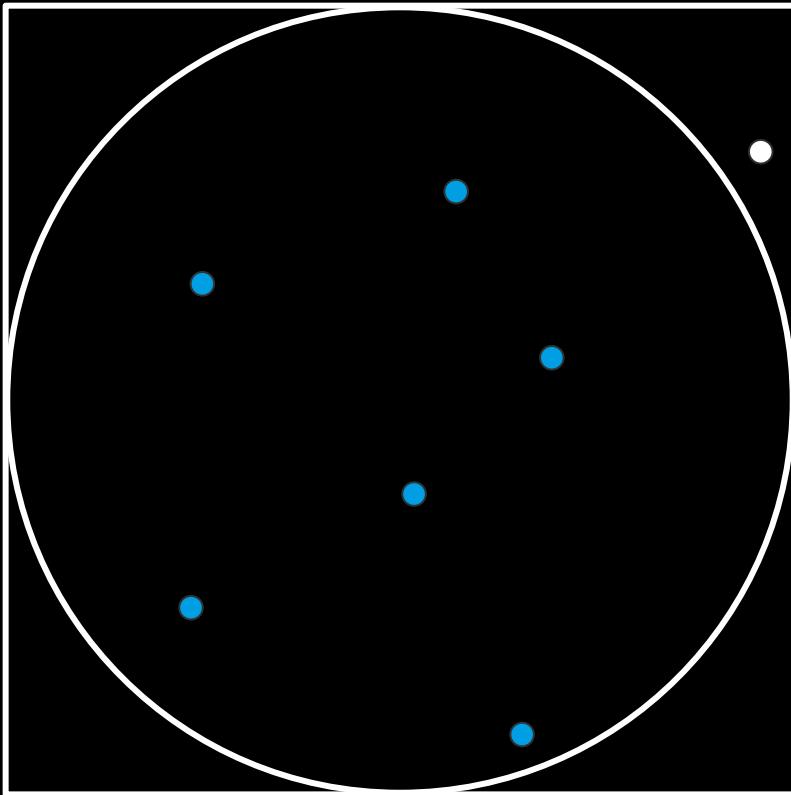
$$4 \frac{\bigcirc}{\square} = \pi$$
$$= 3$$



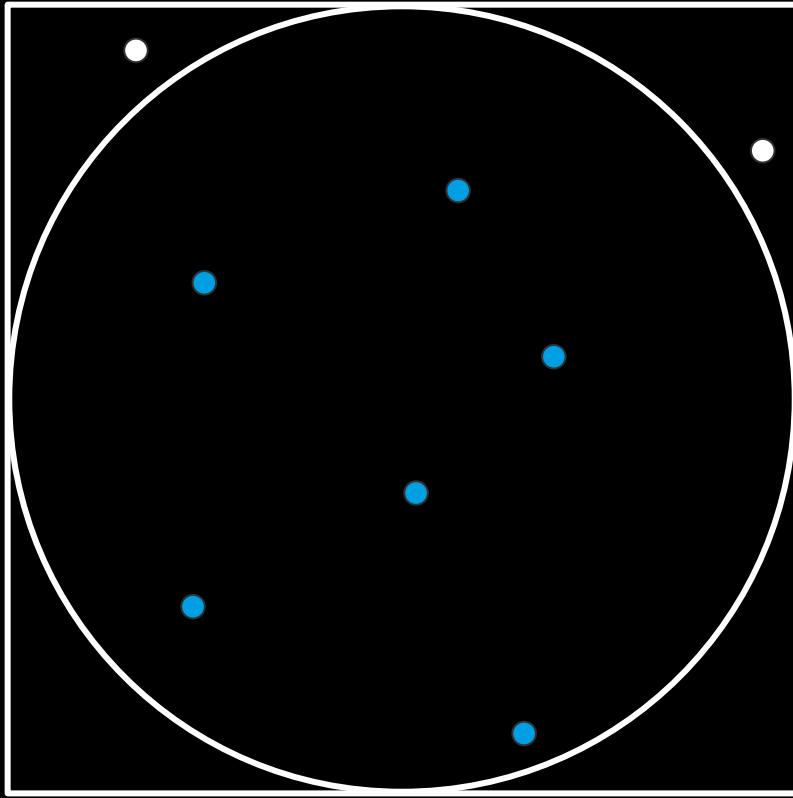
$$4 \frac{\bigcirc}{\square} = \pi$$
$$= 3,2$$



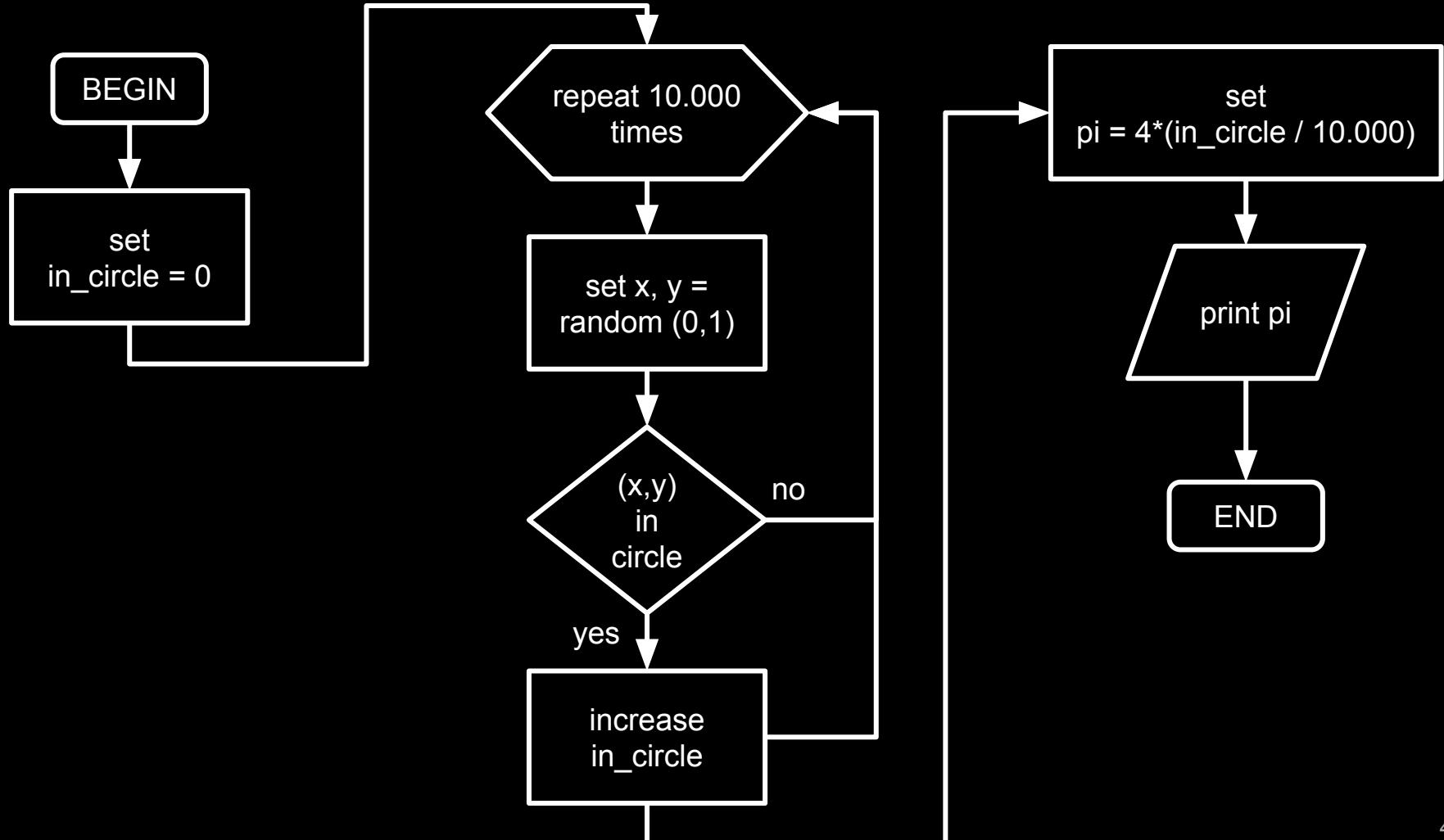
$$4 \frac{\bigcirc}{\square} = \pi$$
$$= 3,33$$



$$4 \frac{\bigcirc}{\square} = \pi$$
$$= 3,43$$



$$4 \frac{\textcircled{1}}{\textsquare} = \pi$$
$$= 3$$



gregory-leibniz series

$$\pi = 4 \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \dots\right)$$



sorting

[9, 5, 2, 1, 4, 7]

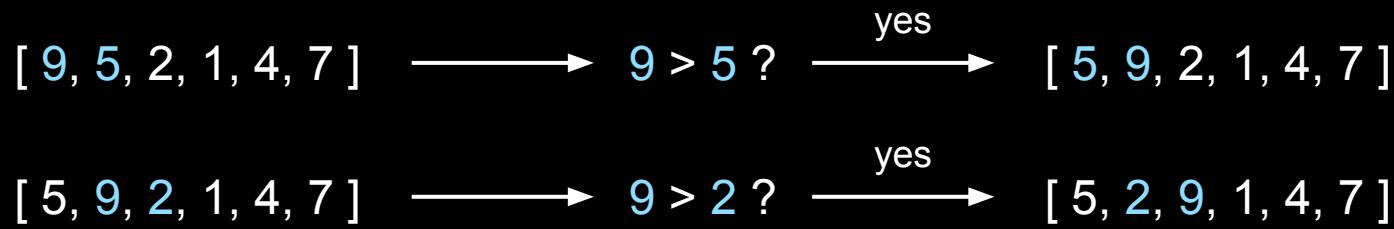
bubble sort

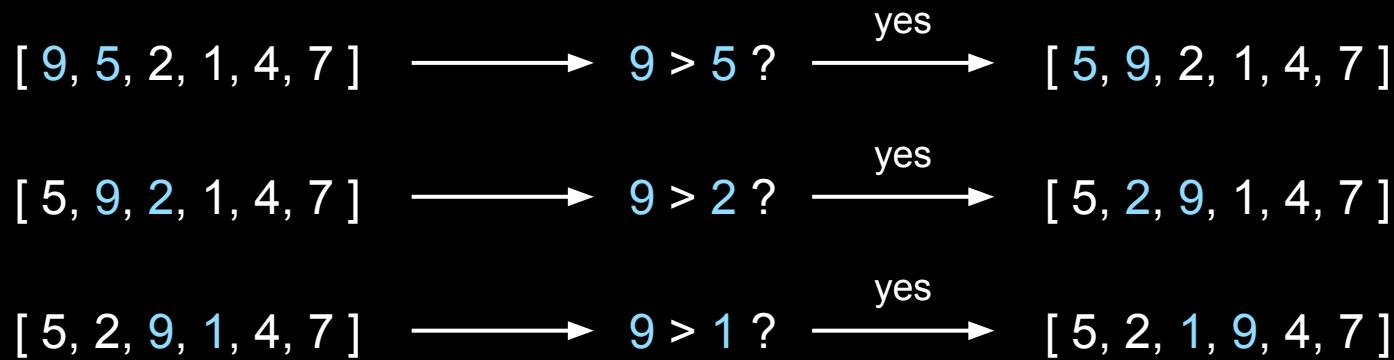
repeatedly compare and swap elements until done.

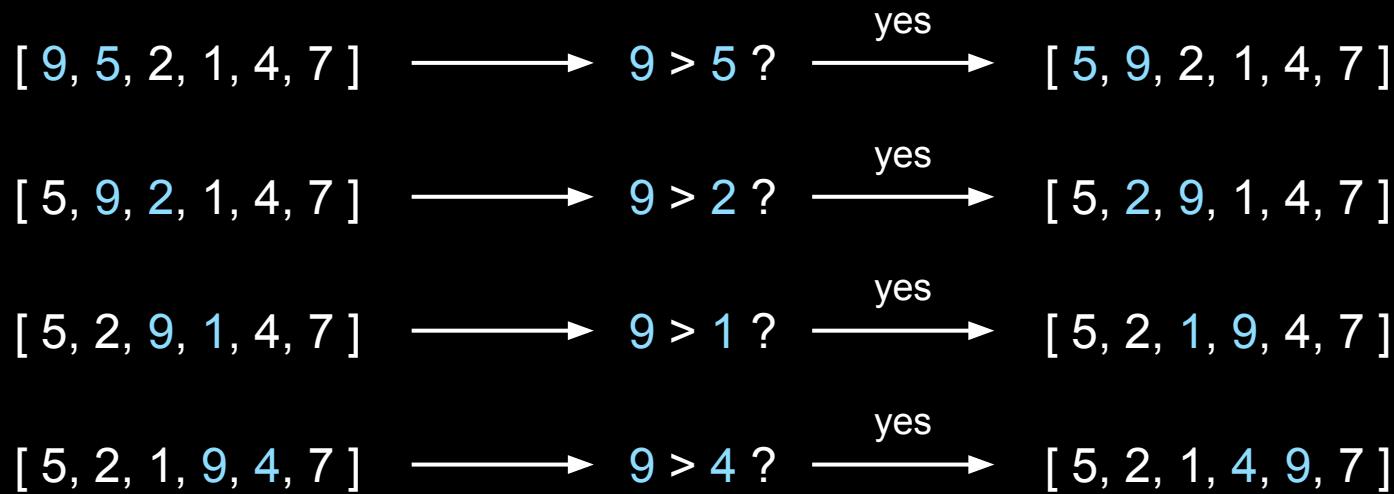
[9, 5, 2, 1, 4, 7]

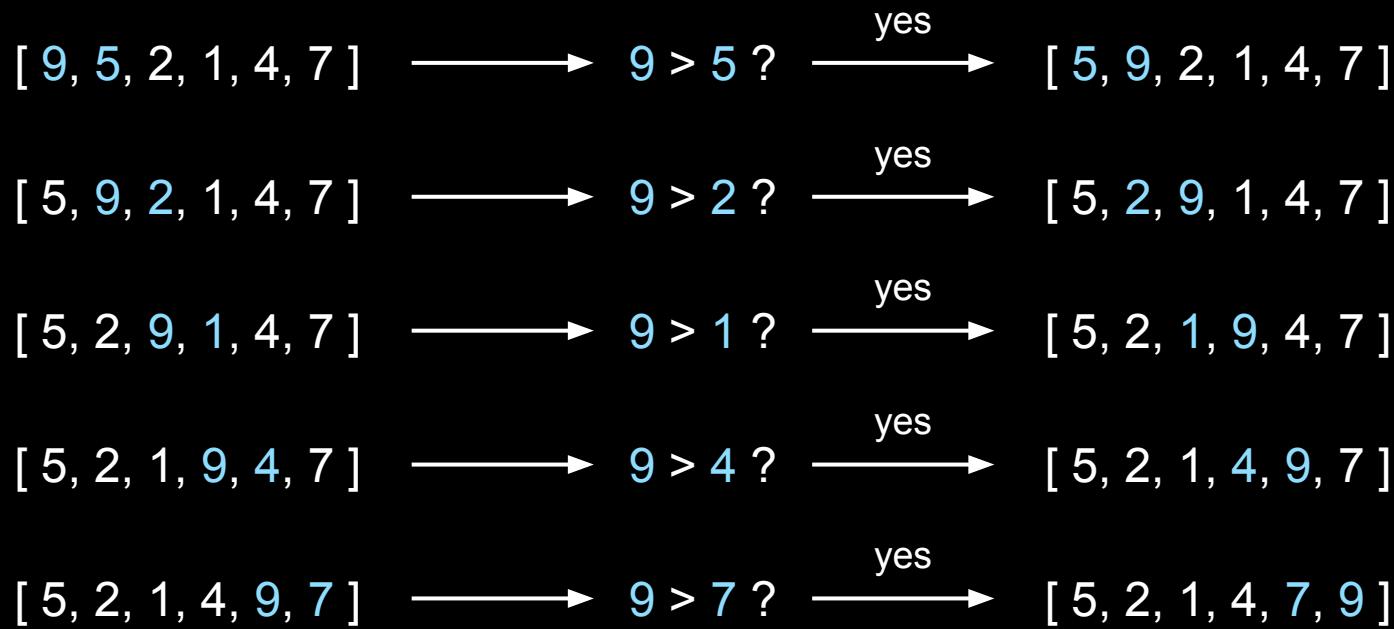
[9, 5, 2, 1, 4, 7] \longrightarrow $9 > 5 ?$

[9, 5, 2, 1, 4, 7] → 9 > 5 ? → yes [5, 9, 2, 1, 4, 7]

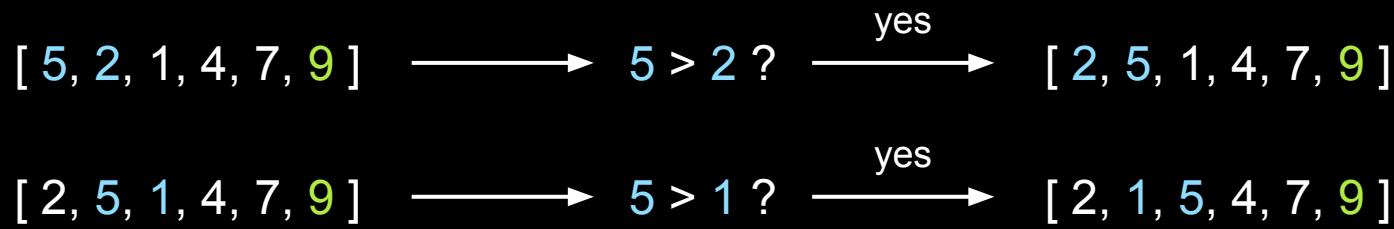


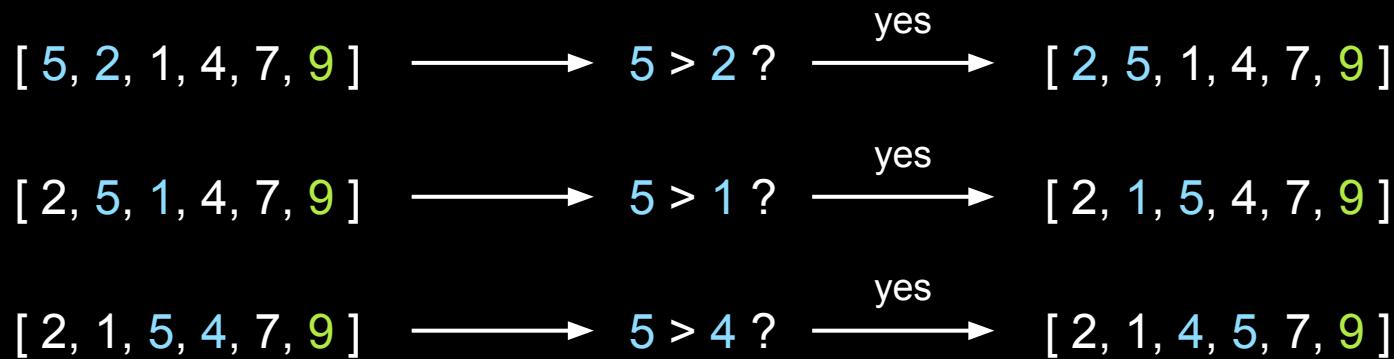


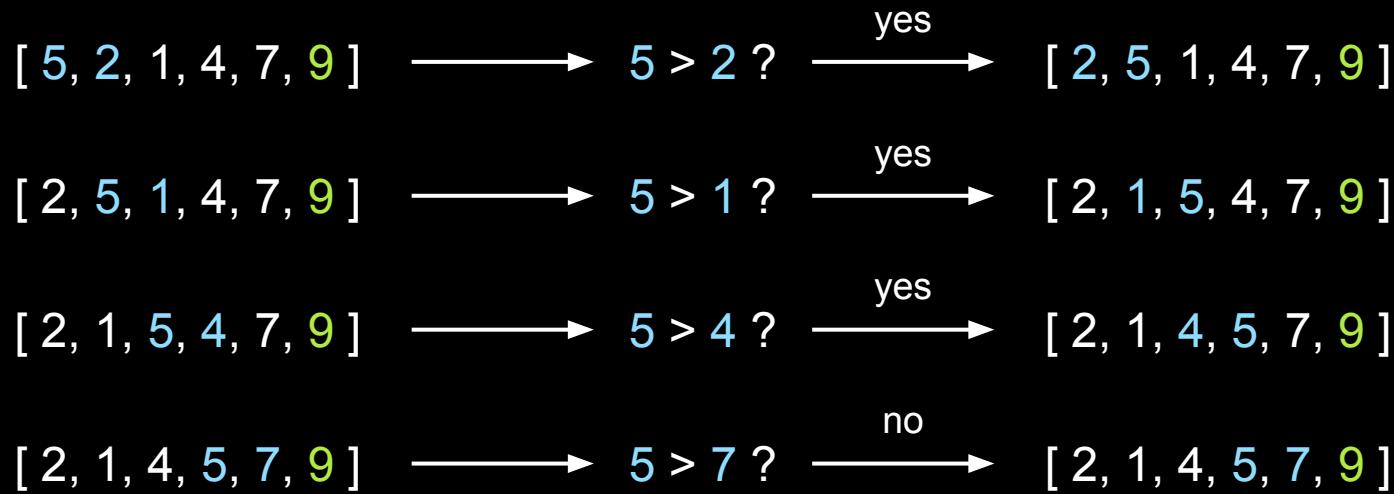




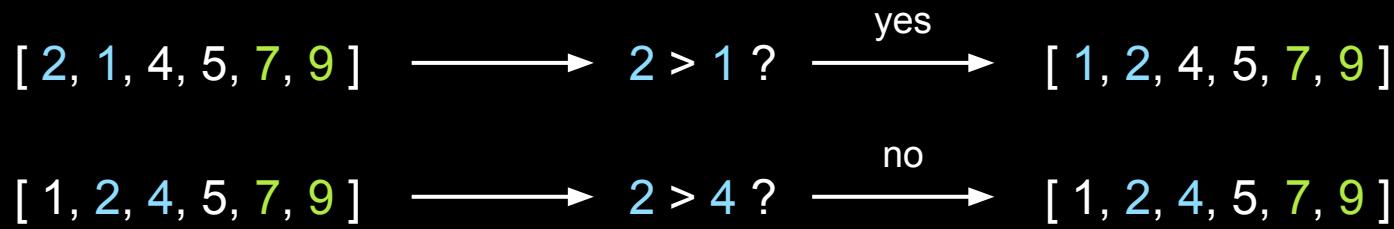
[5, 2, 1, 4, 7, 9] → 5 > 2 ? → yes [2, 5, 1, 4, 7, 9]

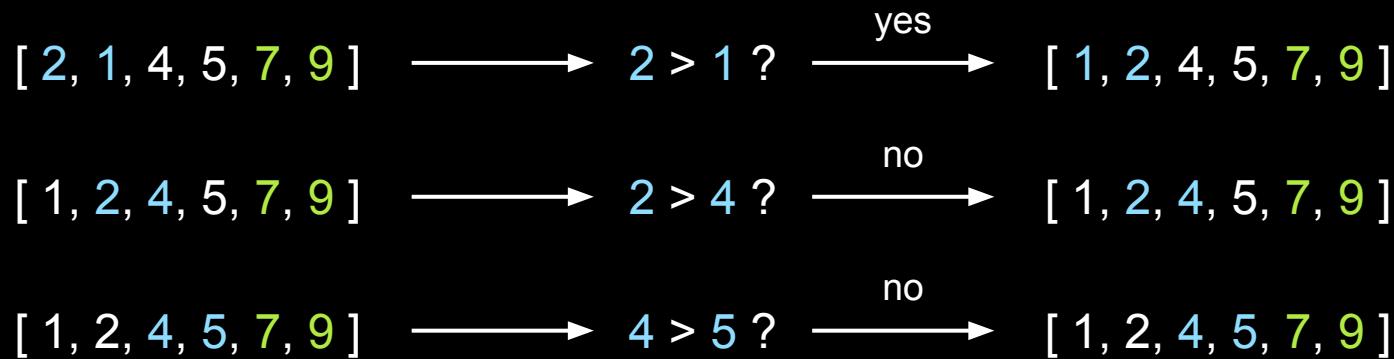




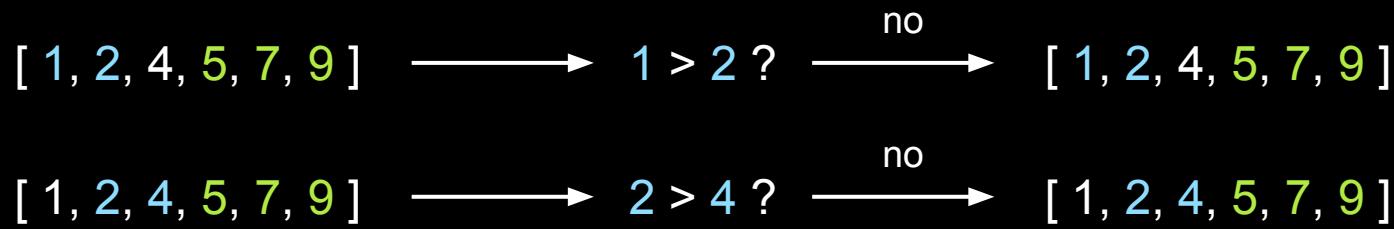


[2, 1, 4, 5, 7, 9] → 2 > 1 ? → yes [1, 2, 4, 5, 7, 9]





[1, 2, 4, 5, 7, 9] → 1 > 2 ? →
no [1, 2, 4, 5, 7, 9]



[1, 2, 4, 5, 7, 9] → 1 > 2 ? →
no [1, 2, 4, 5, 7, 9]

[1, 2, 4, 5, 7, 9] → 1 > 2 ? →
no [1, 2, 4, 5, 7, 9]

[1, 2, 4, 5, 7, 9] DONE!

selection sort

find the smallest element and move it to front. repeat for the rest of the elements.

move to front

[9, 5, 2, 1, 4, 7] → [1, 5, 9, 2, 4, 7]

[9, 5, 2, 1, 4, 7] move to front [1, 5, 9, 2, 4, 7]

[1, 5, 9, 2, 4, 7] move to front [1, 2, 5, 9, 4, 7]

move to front

[9, 5, 2, 1, 4, 7] → [1, 5, 9, 2, 4, 7]

[1, 5, 9, 2, 4, 7] → [1, 2, 5, 9, 4, 7]

[1, 2, 5, 9, 4, 7] → [1, 2, 4, 5, 9, 7]

move to front

[9, 5, 2, 1, 4, 7] → [1, 5, 9, 2, 4, 7]

[1, 5, 9, 2, 4, 7] → [1, 2, 5, 9, 4, 7]

[1, 2, 5, 9, 4, 7] → [1, 2, 4, 5, 9, 7]

[1, 2, 4, 5, 9, 7] → [1, 2, 4, 5, 9, 7]

[9, 5, 2, 1, 4, 7] move to front [1, 5, 9, 2, 4, 7]

[1, 5, 9, 2, 4, 7] move to front [1, 2, 5, 9, 4, 7]

[1, 2, 5, 9, 4, 7] move to front [1, 2, 4, 5, 9, 7]

[1, 2, 4, 5, 9, 7] move to front [1, 2, 4, 5, 9, 7]

[1, 2, 4, 5, 9, 7] move to front [1, 2, 4, 5, 7, 9]

- move to front
- [9, 5, 2, 1, 4, 7] → [1, 5, 9, 2, 4, 7]
- [1, 5, 9, 2, 4, 7] → [1, 2, 5, 9, 4, 7]
- [1, 2, 5, 9, 4, 7] → [1, 2, 4, 5, 9, 7]
- [1, 2, 4, 5, 9, 7] → [1, 2, 4, 5, 9, 7]
- [1, 2, 4, 5, 9, 7] → [1, 2, 4, 5, 7, 9]
- [1, 2, 4, 5, 7, 9] → [1, 2, 4, 5, 7, 9]

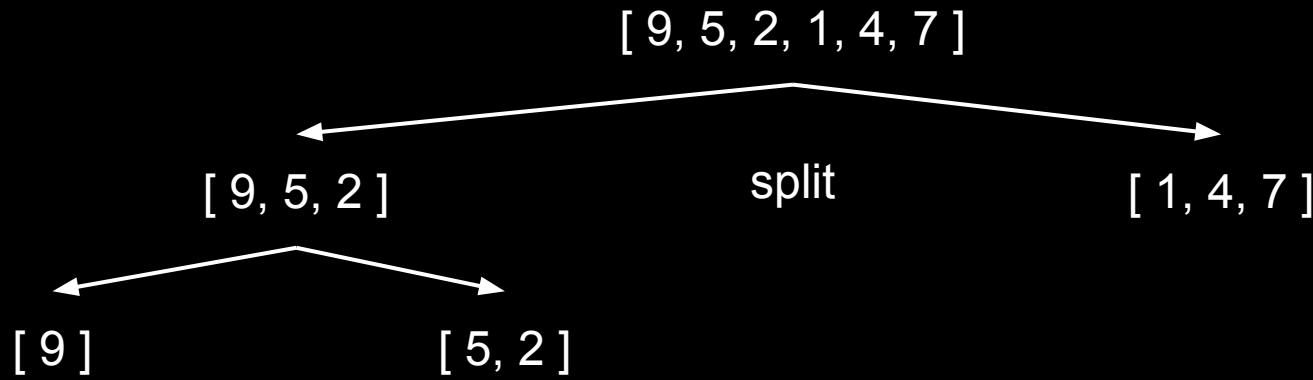
merge sort

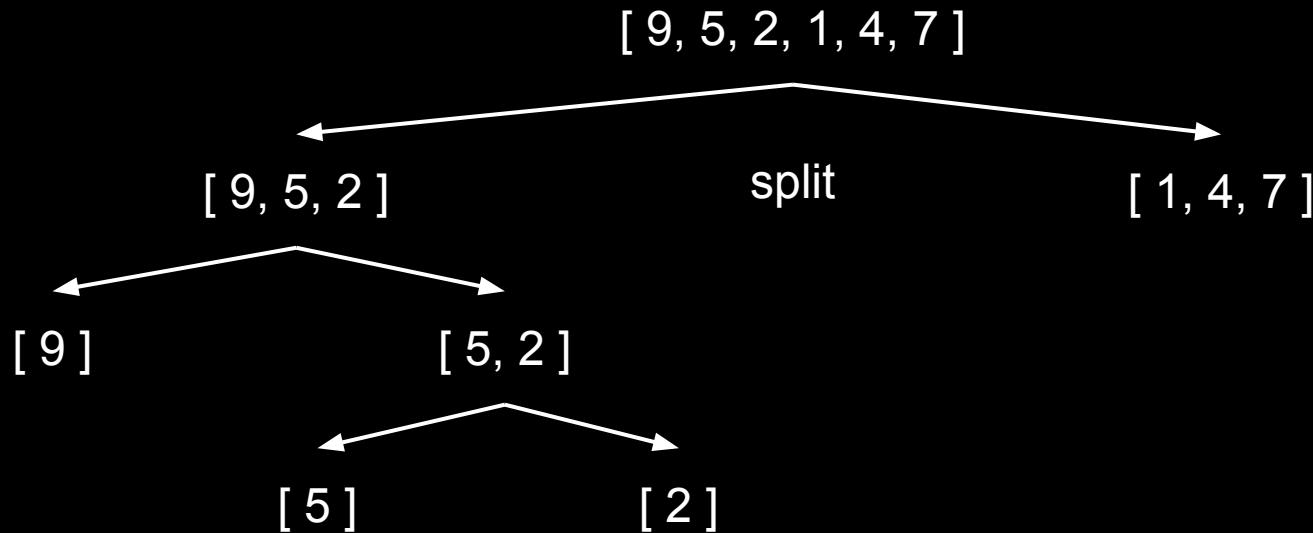
divide the elements recursively in two halves until only one element is left. then merge the sorted halves back together.

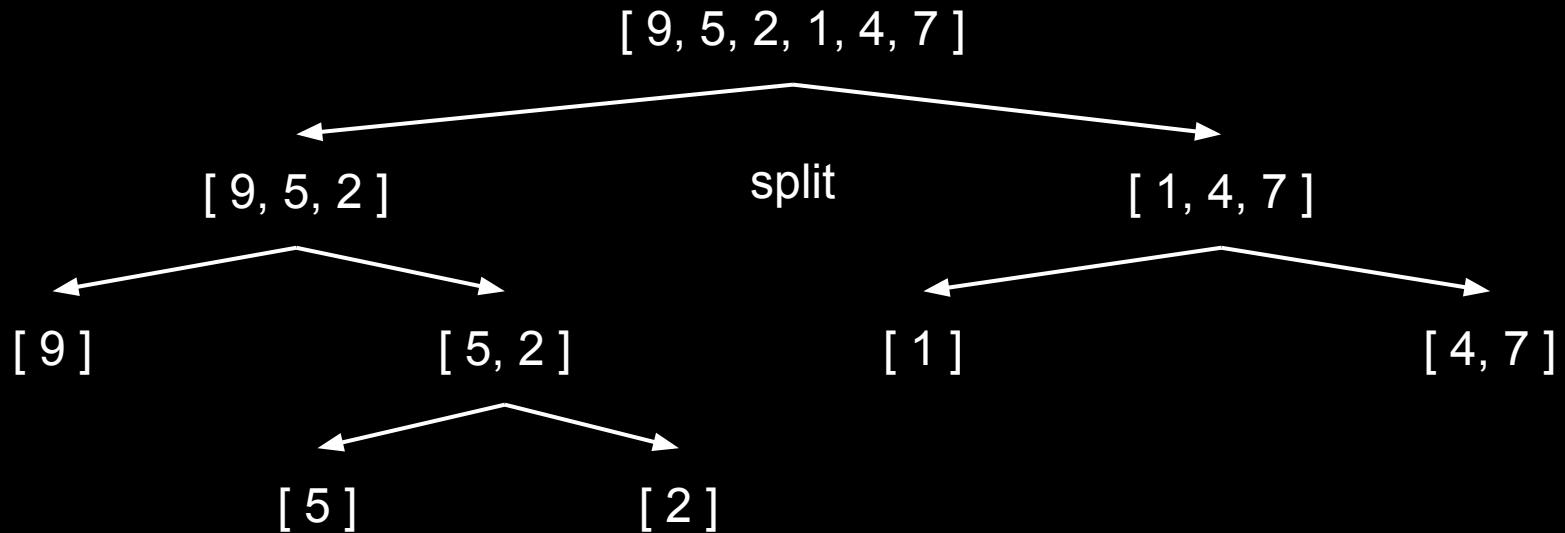
divide the elements **recursively** in two halves until only one element is left.
then merge the sorted halves back together.

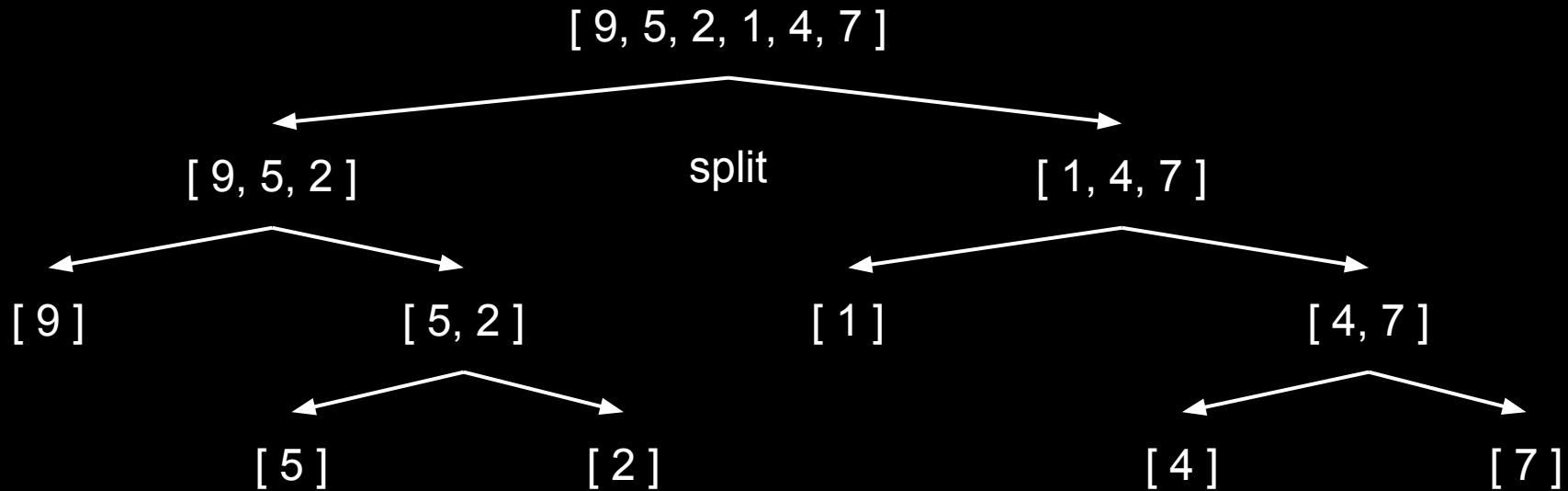
[9, 5, 2, 1, 4, 7]

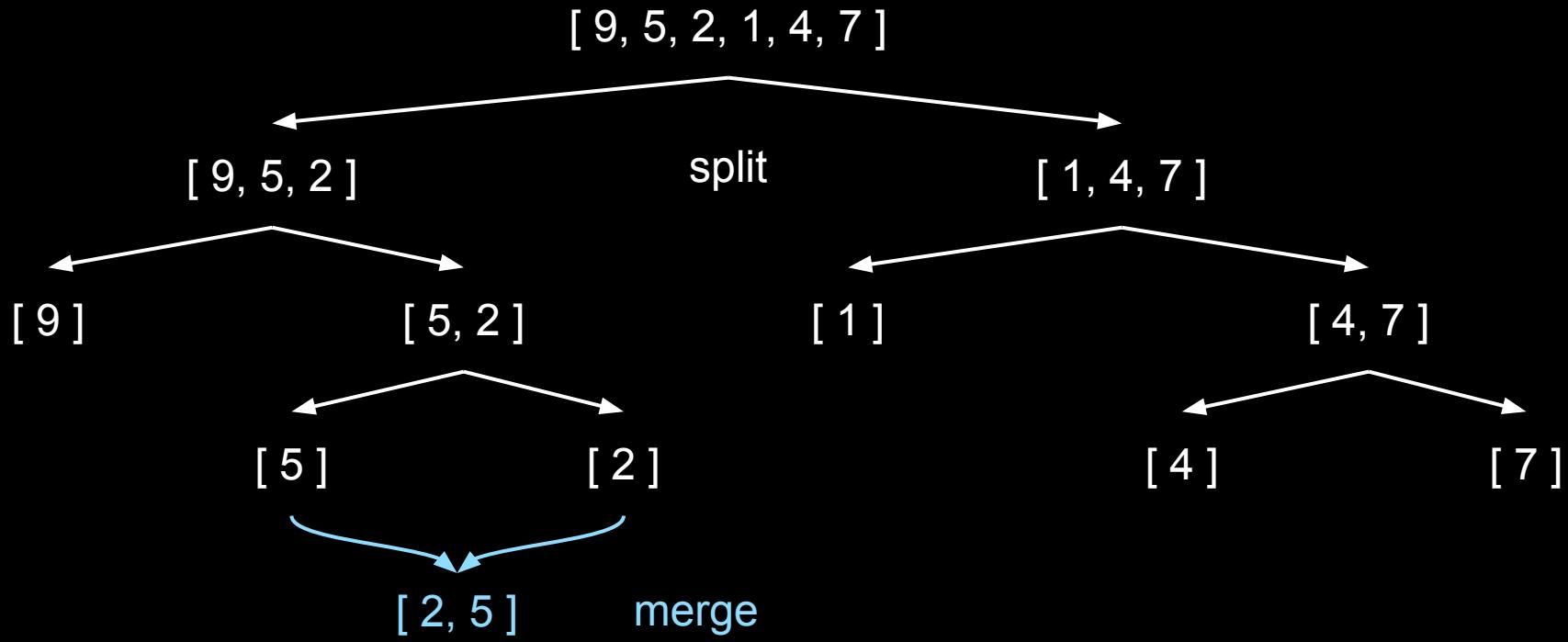


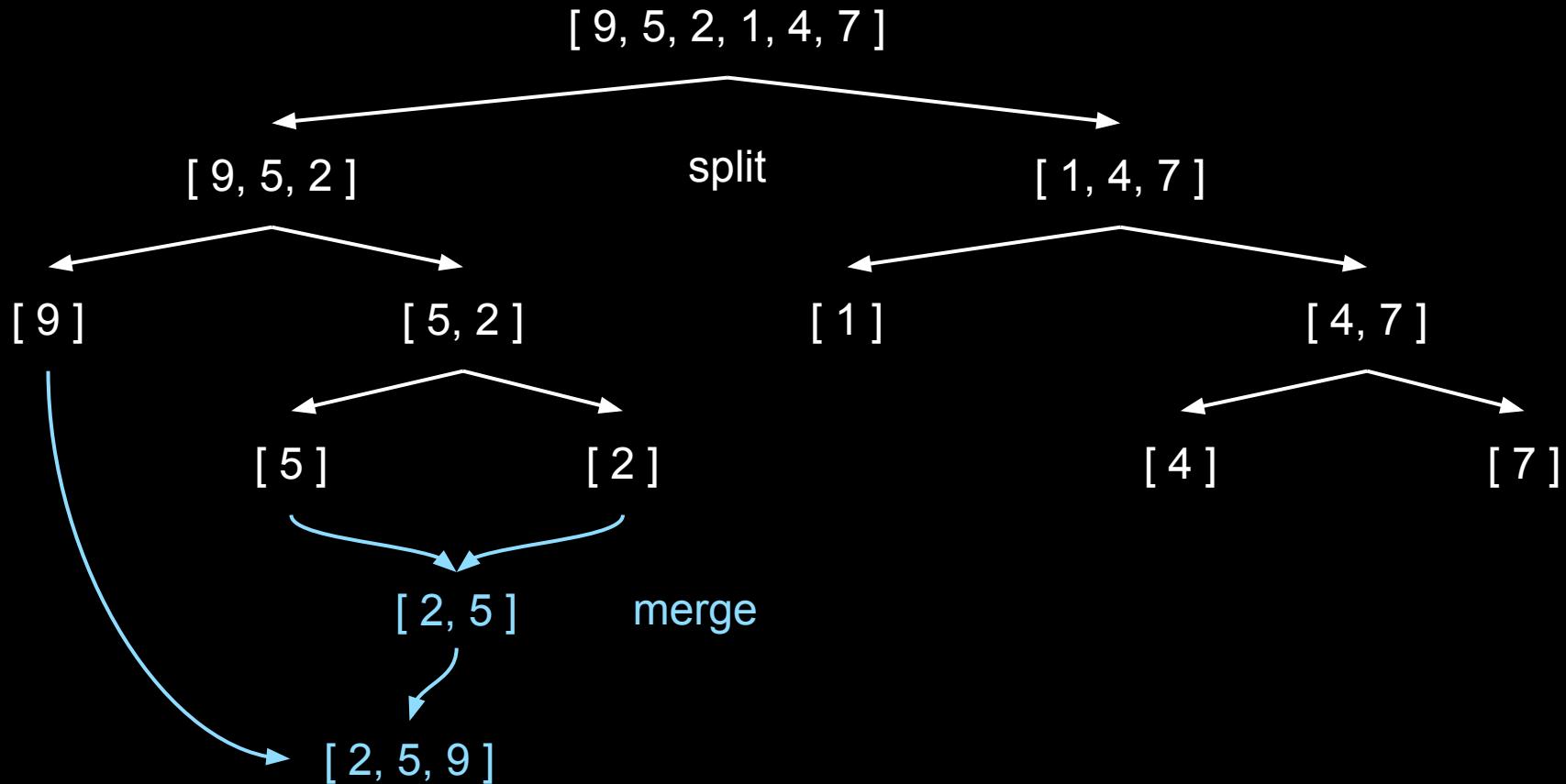


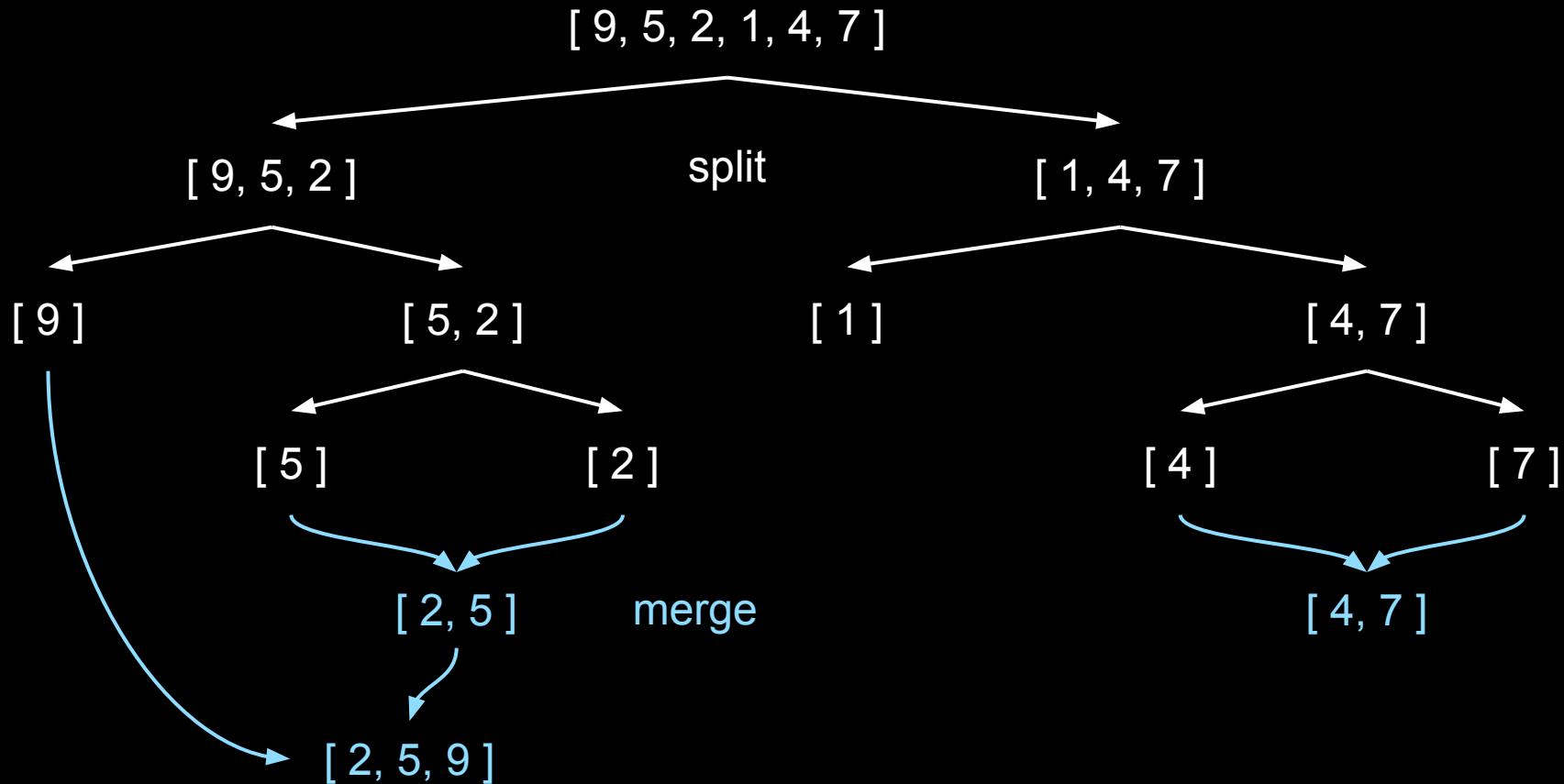


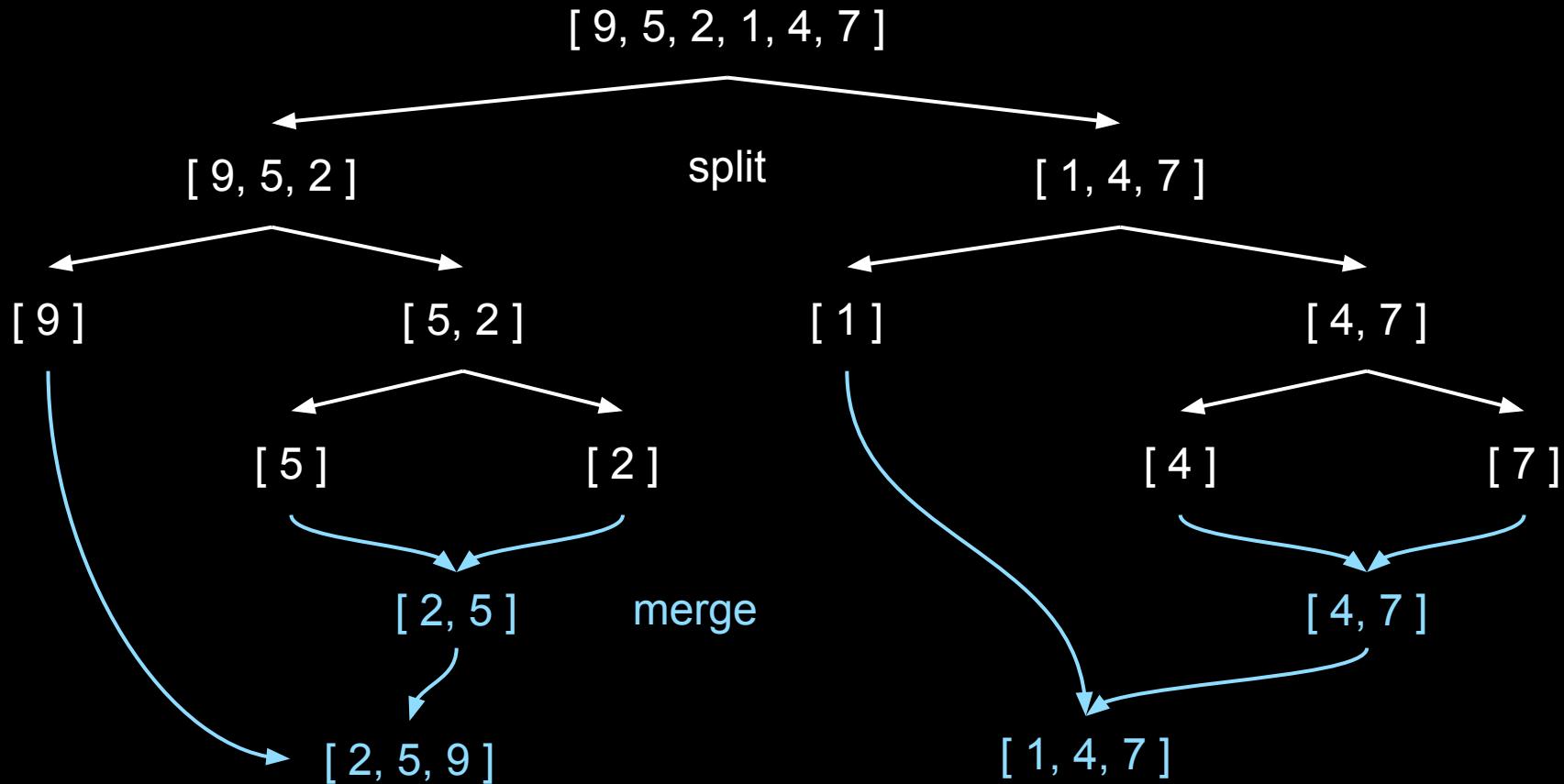


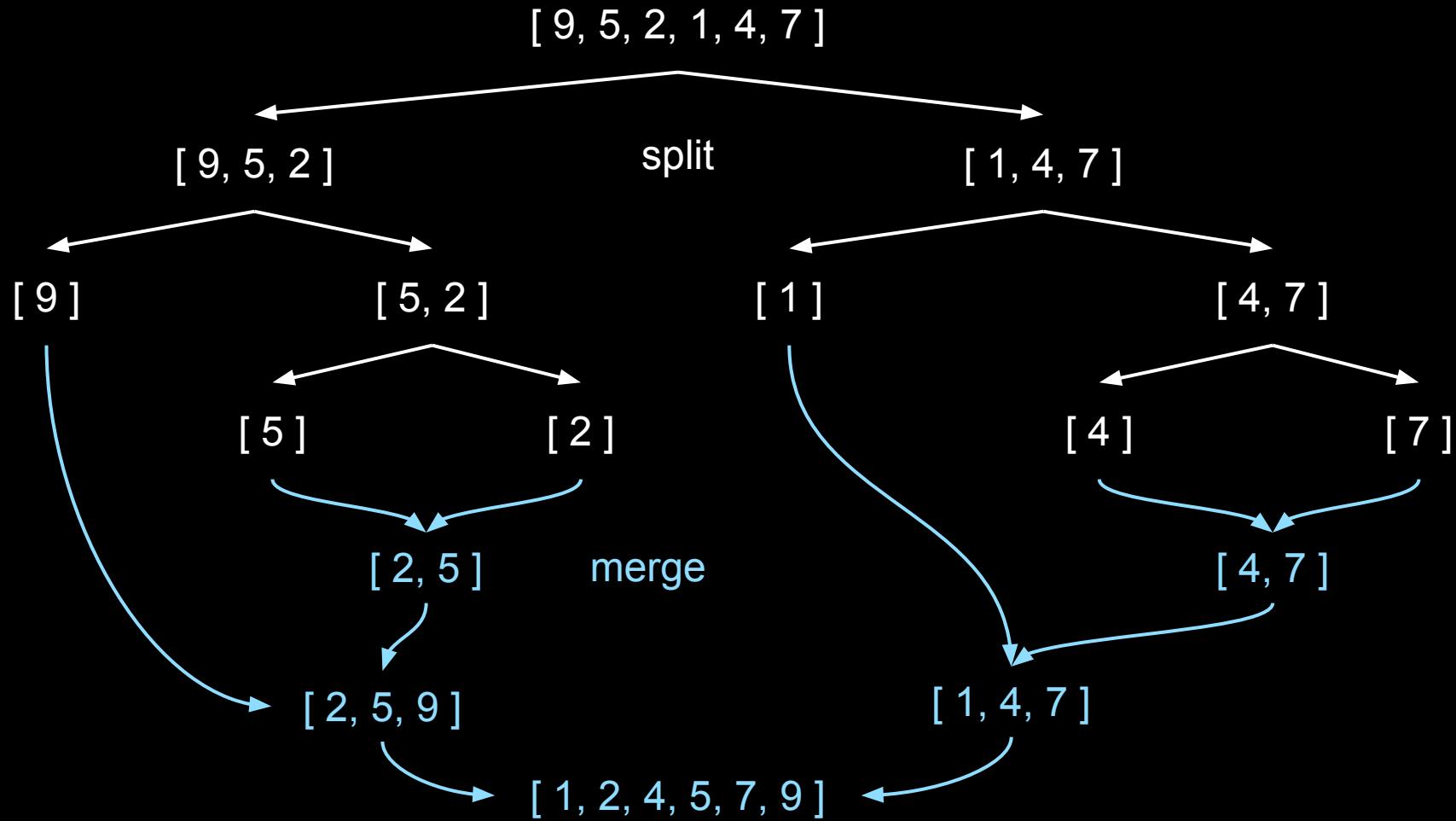




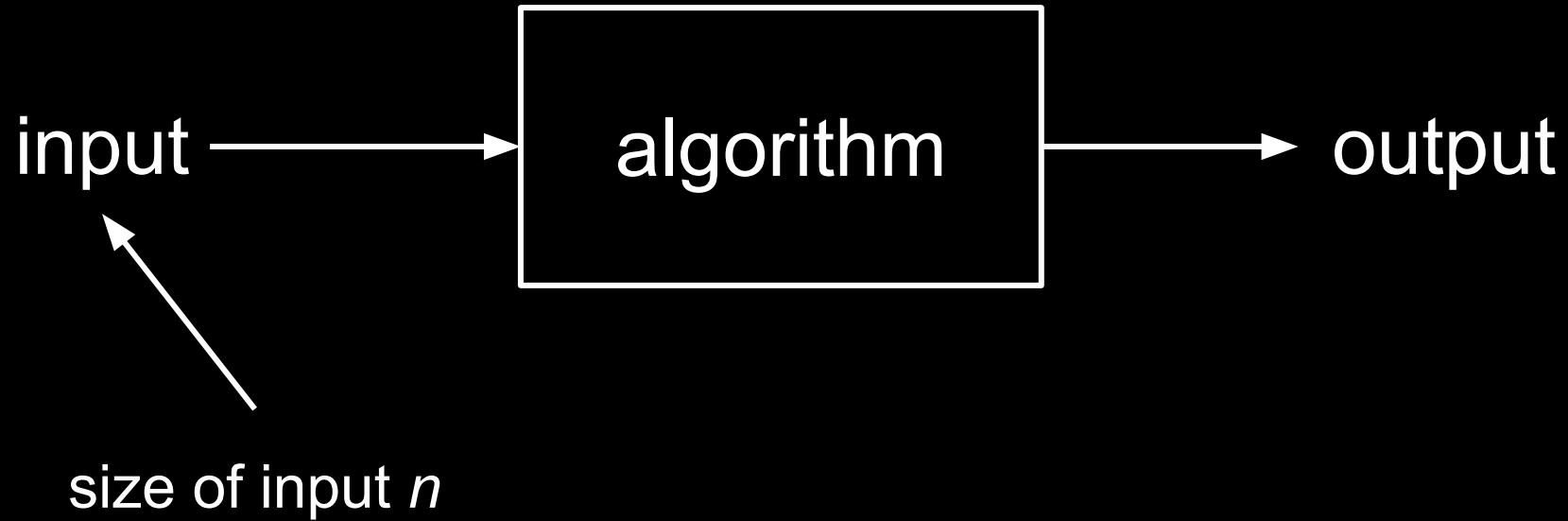


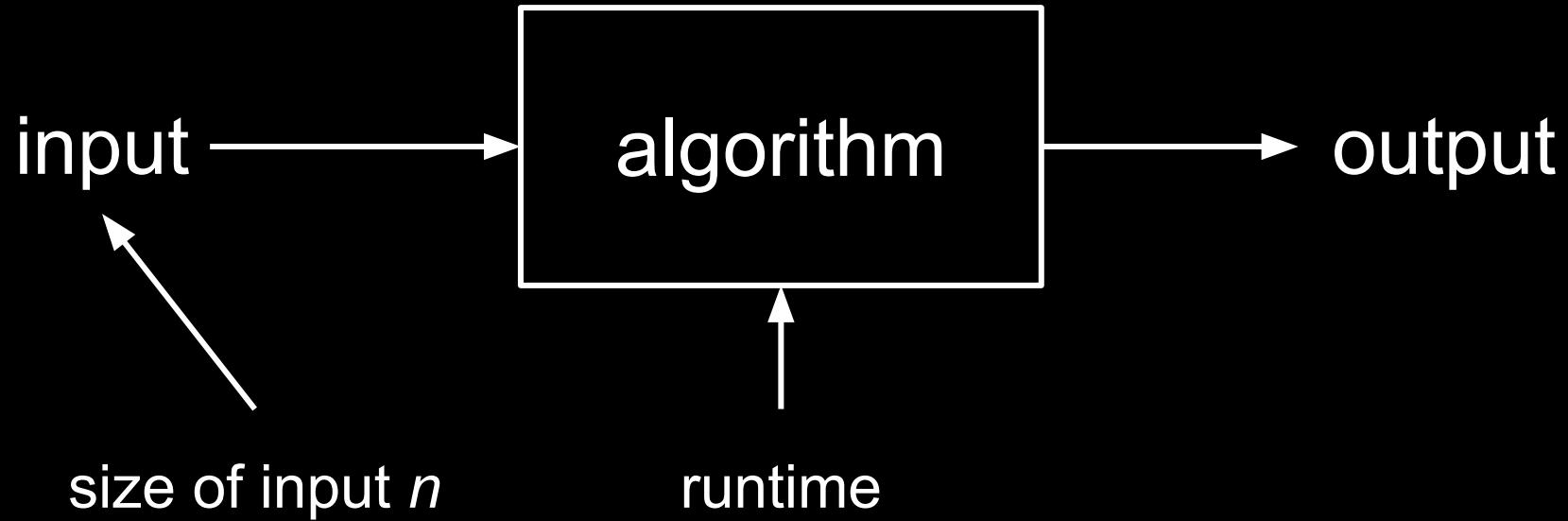




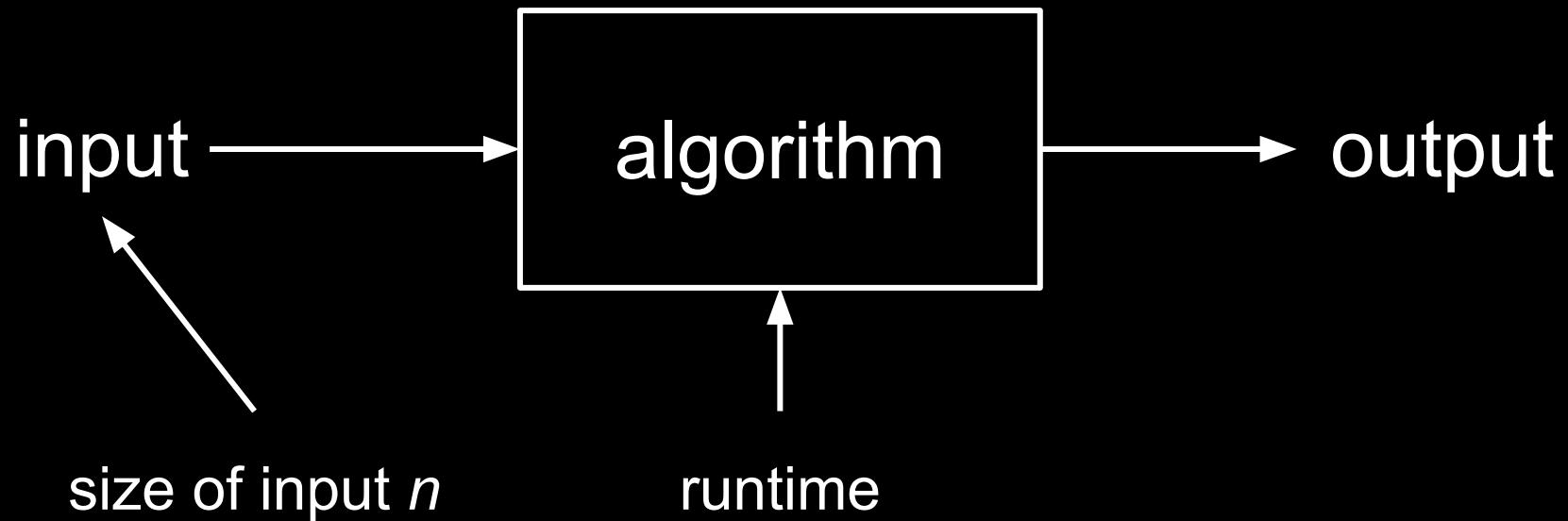


complexity





$O(n)$



| | |
|---------------|--|
| $O(1)$ | runtime is constant and independent of problem size |
| $O(\log_2 n)$ | runtime is determined by the logarithm of problem size |
| $O(n)$ | runtime is linear to problem size |
| $O(n^2)$ | runtime grows quadratically with the size of the problem |
| $O(n^3)$ | runtime grows cubically with the size of the problem |
| $O(2^n)$ | runtime grows exponentially with the size of the problem |
| $O(n!)$ | runtime grows factorially with the size of the problem |

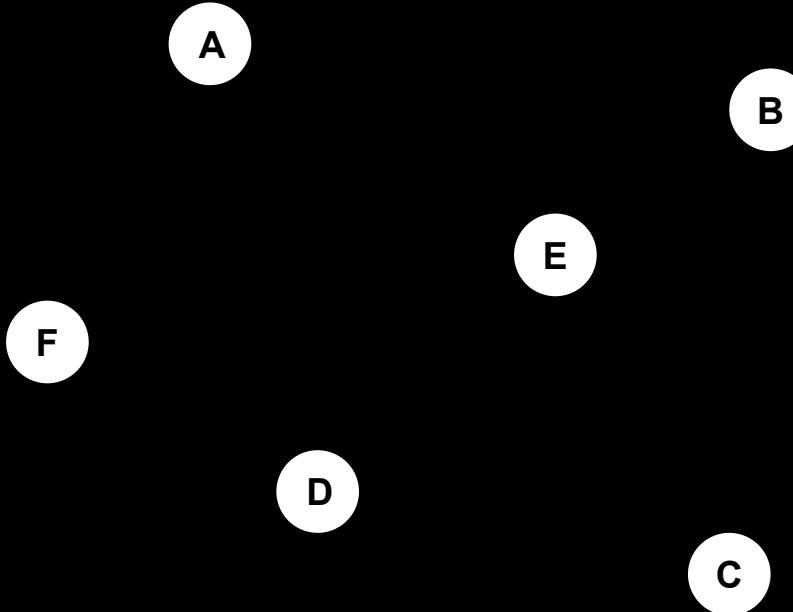


optimization

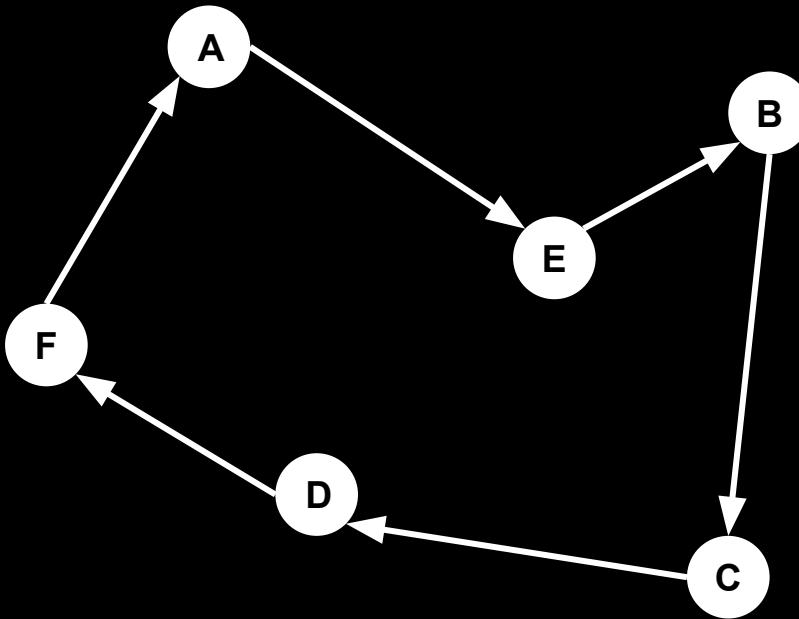


traveling salesmen

shortest tour?



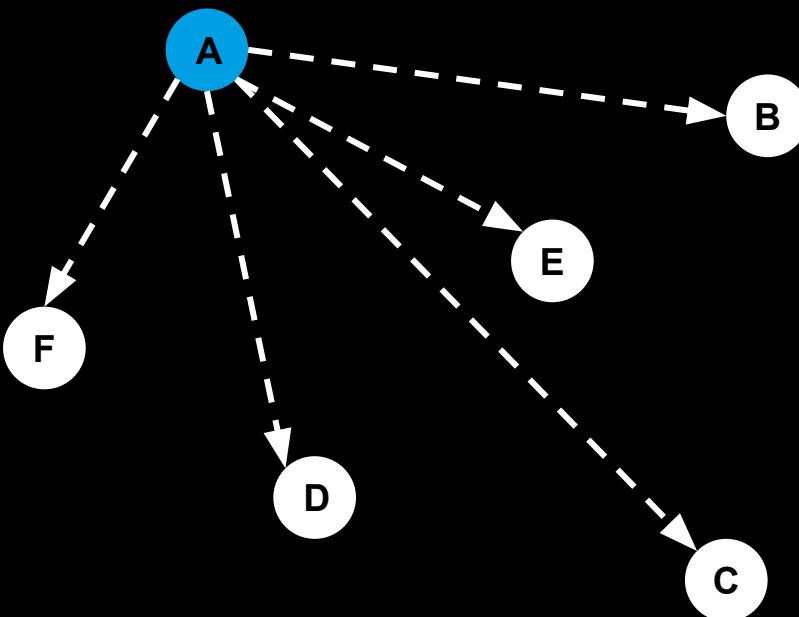
shortest tour?



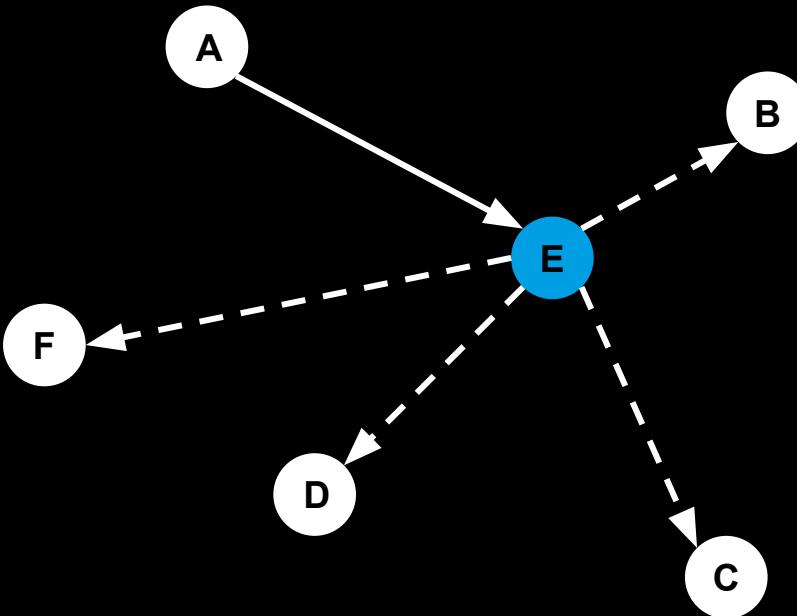
brute force

$$O(n) = n!$$

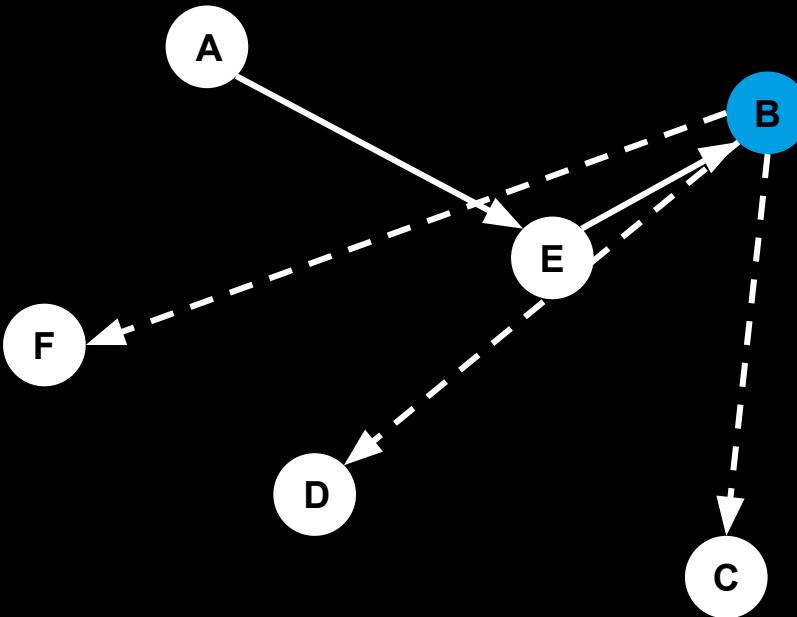
5 possible cities



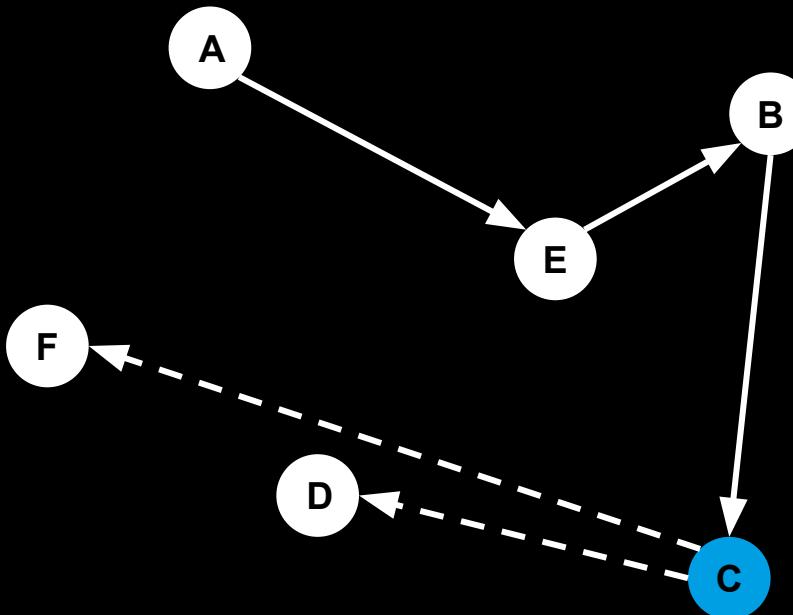
5 4 possible cities



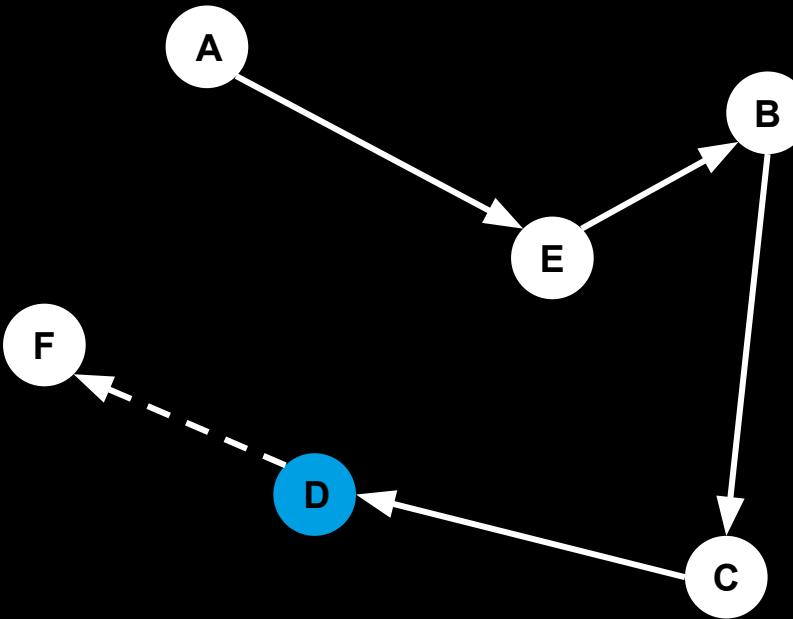
5 4 3 possible cities



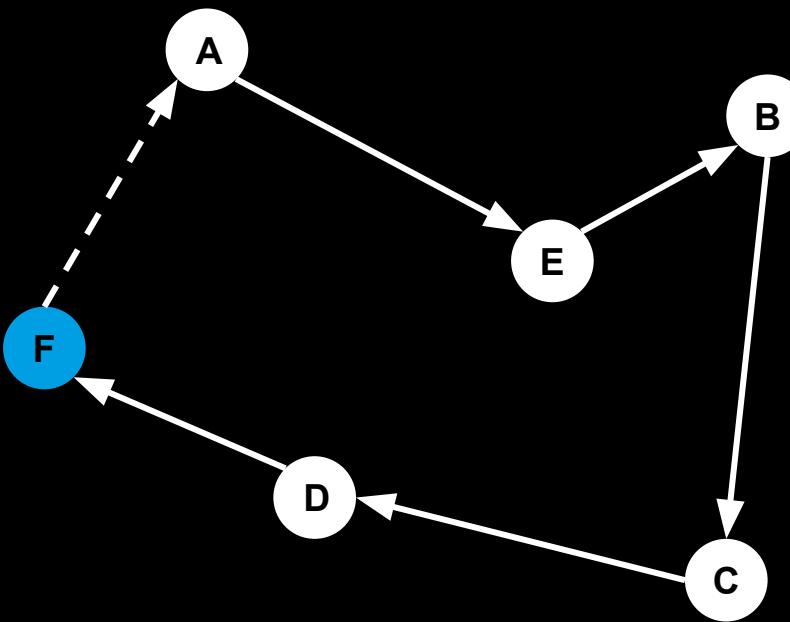
5 4 3 2 possible cities



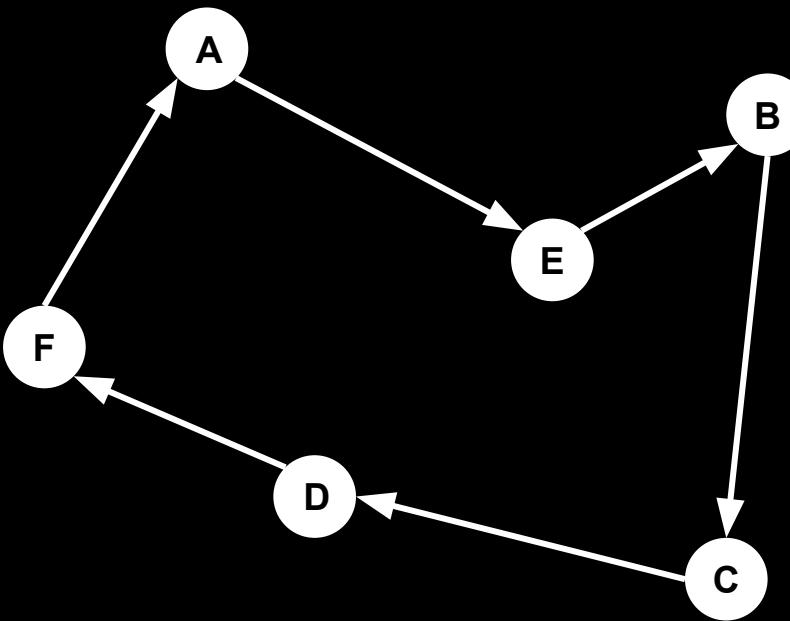
5 4 3 2 1 possible city



5 4 3 2 1 return nome



5 4 3 2 4 return nome



$$O(n) = n!$$

$$n = 5$$

$$O(n) = n!$$

$$n = 5$$

$$N = 5 * 4 * 3 * 2 * 1$$

$$O(n) = n!$$

$$n = 5$$

$$N = 5 * 4 * 3 * 2 * 1$$

$$= 120$$

$n = 10$ $n = 20$ $n = 30$

$$n = 25$$

brute force takes longer than the universe is old

$$n = 60$$

more possible routes than atoms in the universe

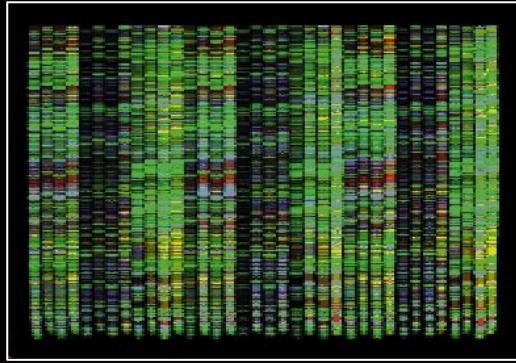


Image source: [IEEE](#)



Image source: [VDI Nachrichten](#)



Image source: [Wikimedia](#)



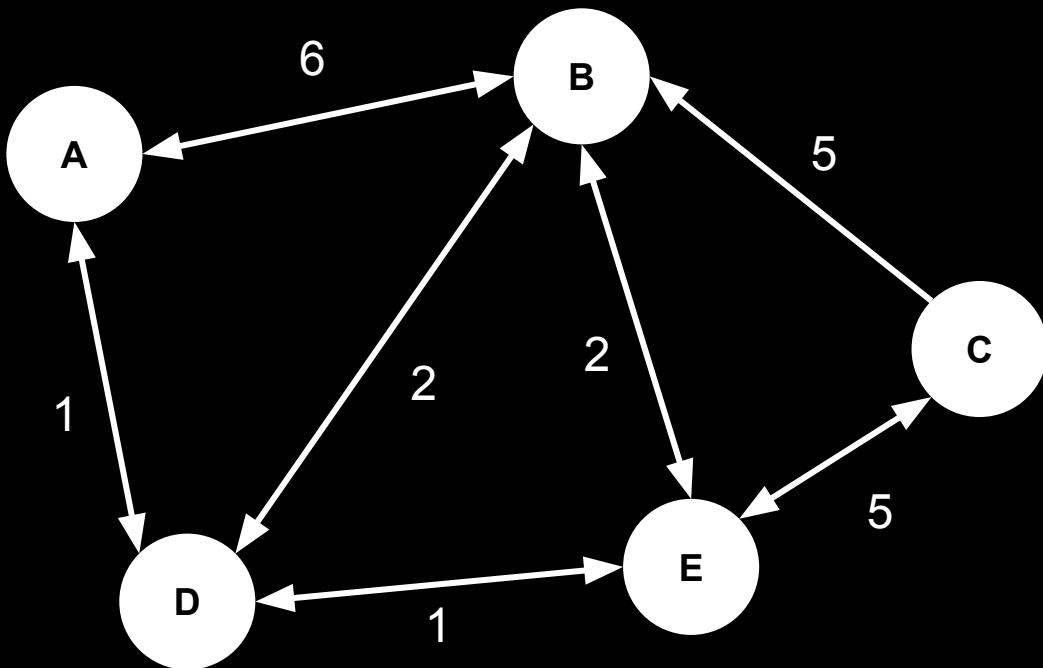
Image source: <https://github.com/meiyi1986/tutorials/blob/master/notebooks/img/pcb-drilling.jpeg>



Image source: [IAS Observatory](#)



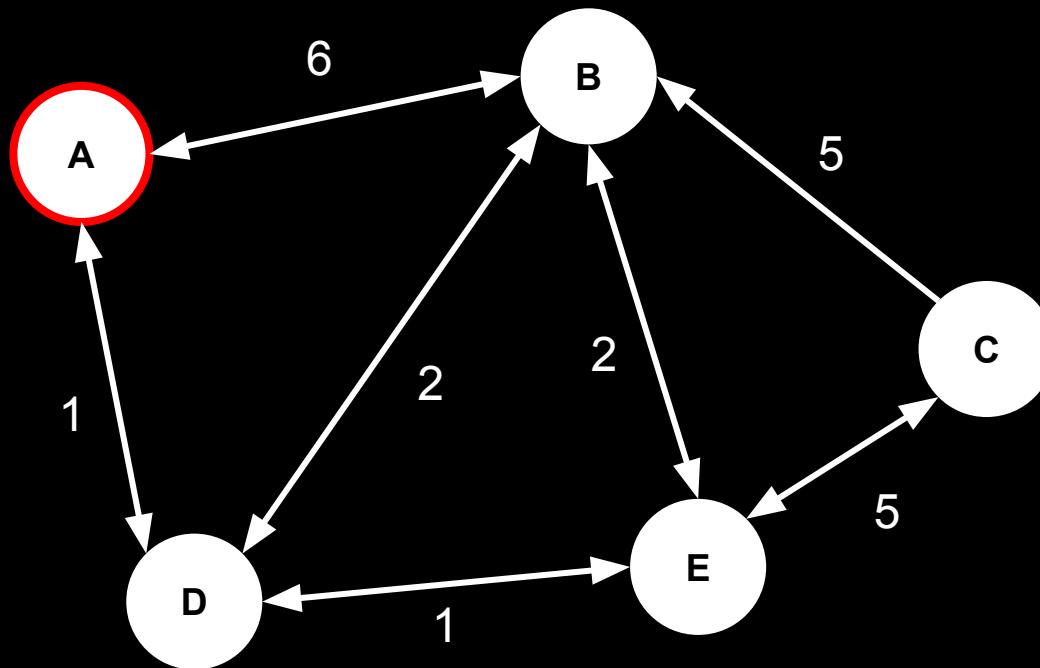
shortest paths



dijkstra's algorithm

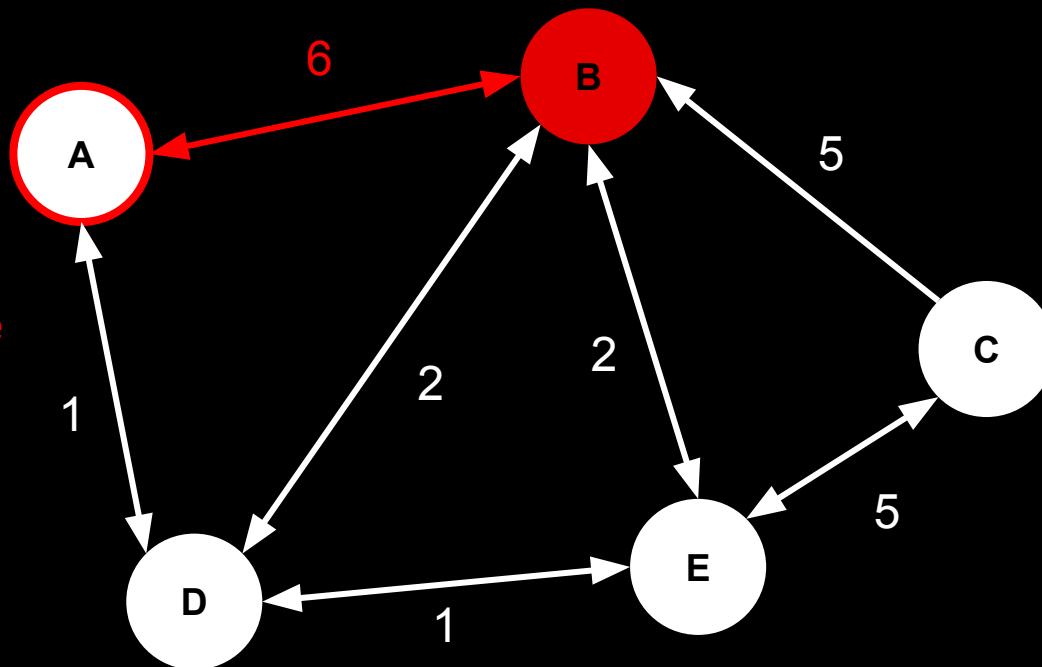
Distances: A = 0, B = ∞ , C = ∞ , D = ∞ , E = ∞

Set A as first location



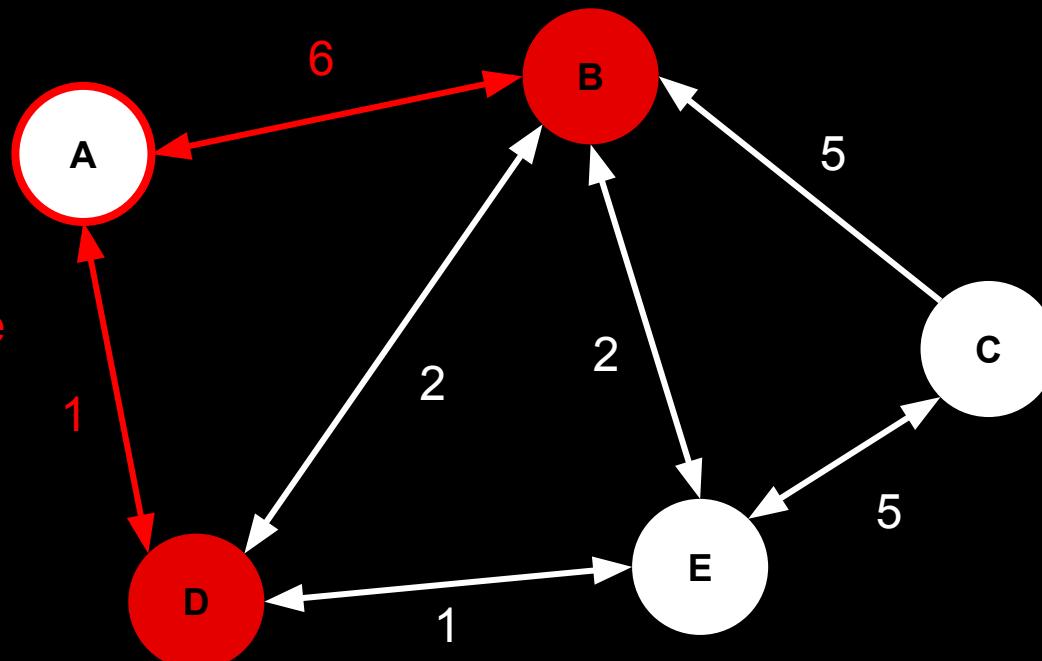
Distances: A = 0, B = 6, C = ∞ , D = ∞ , E = ∞

Update distance
to B to 6



Distances: A = 0, B = 6, C = ∞ , D = 1, E = ∞

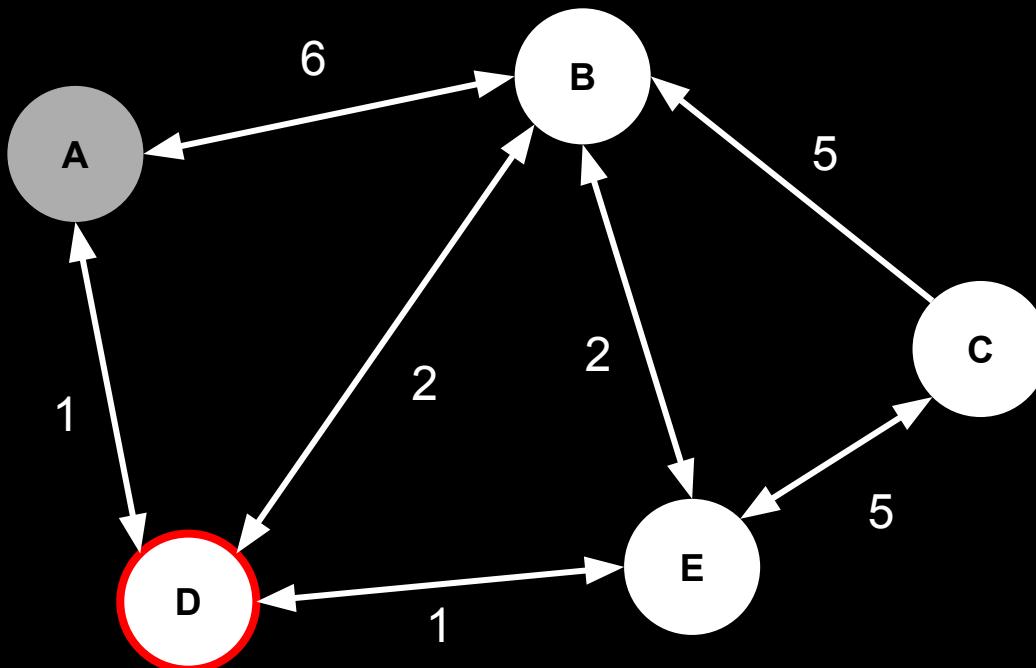
Update distance
to D to 1



Distances: A = 0, B = 6, C = ∞ , D = 1, E = ∞

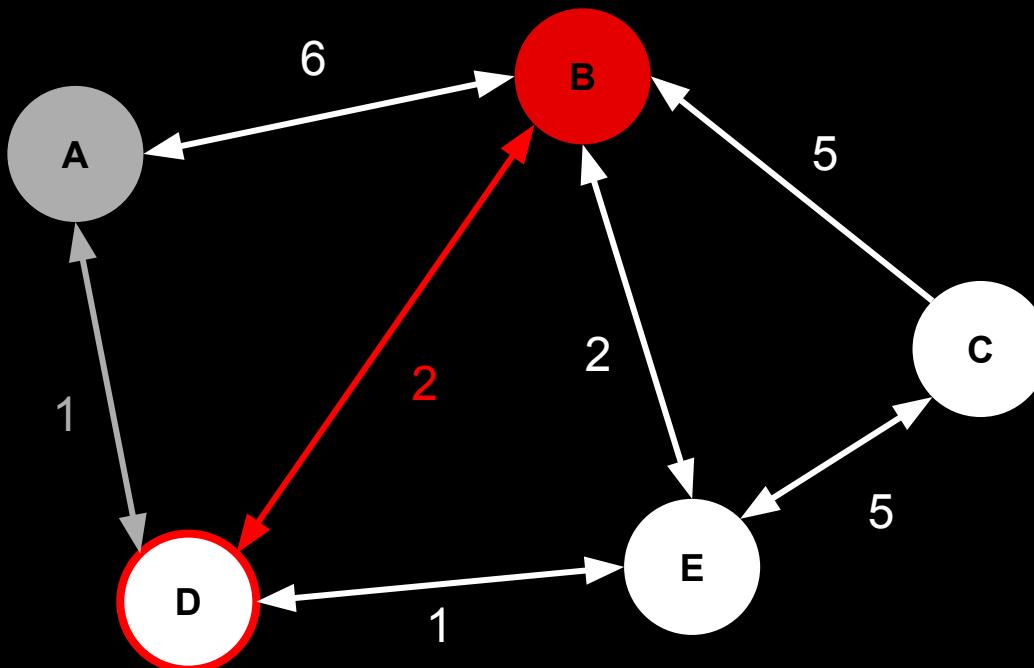
Move to D as next location

Mark A as visited



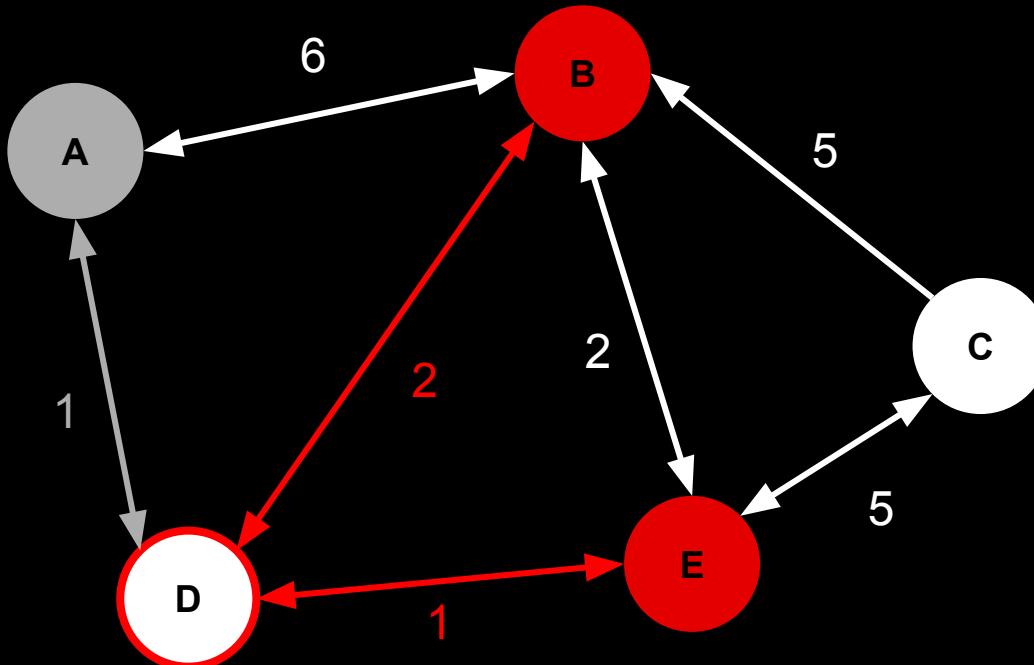
Distances: A = 0, B = 3, C = ∞ , D = 1, E = ∞

Update distance
to B to 3

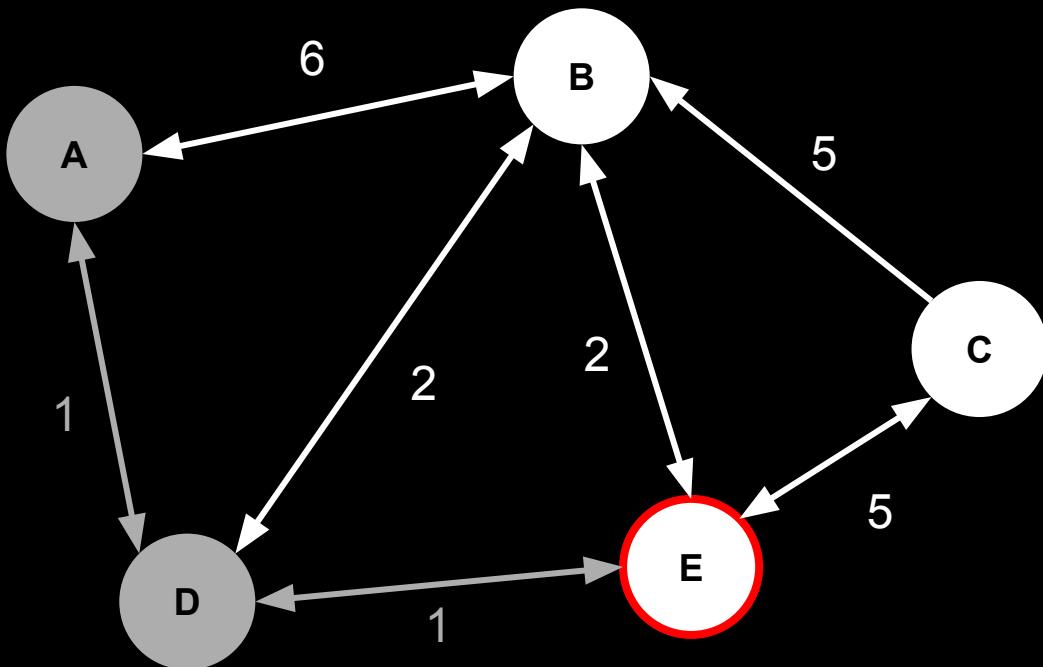


Distances: A = 0, B = 3, C = ∞ , D = 1, E = 2

Update distance
to E to 2

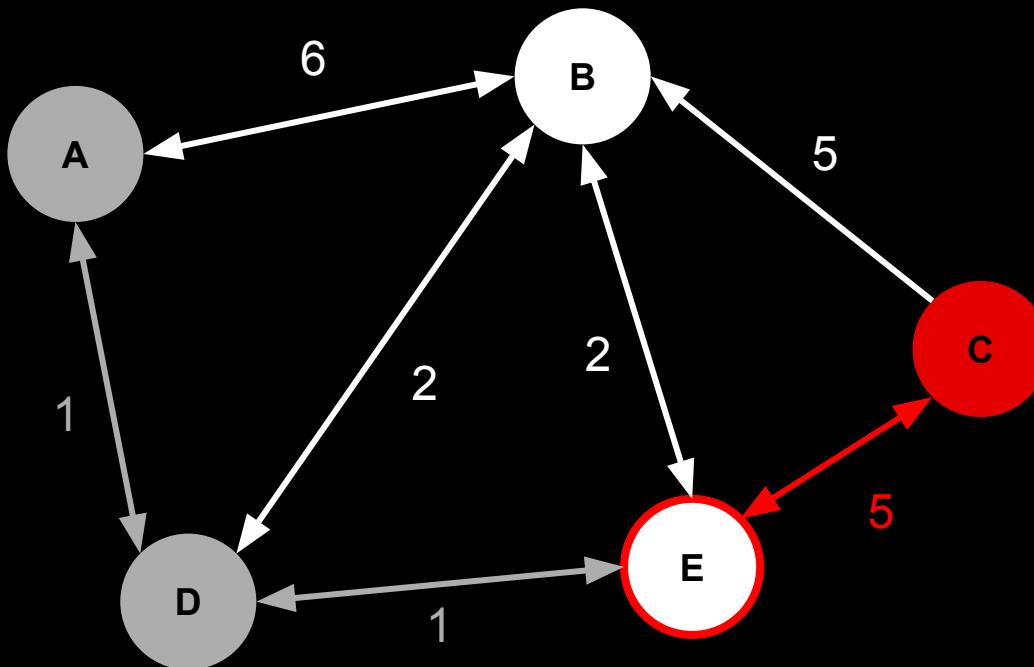


Distances: A = 0, B = 3, C = ∞ , D = 1, E = 2



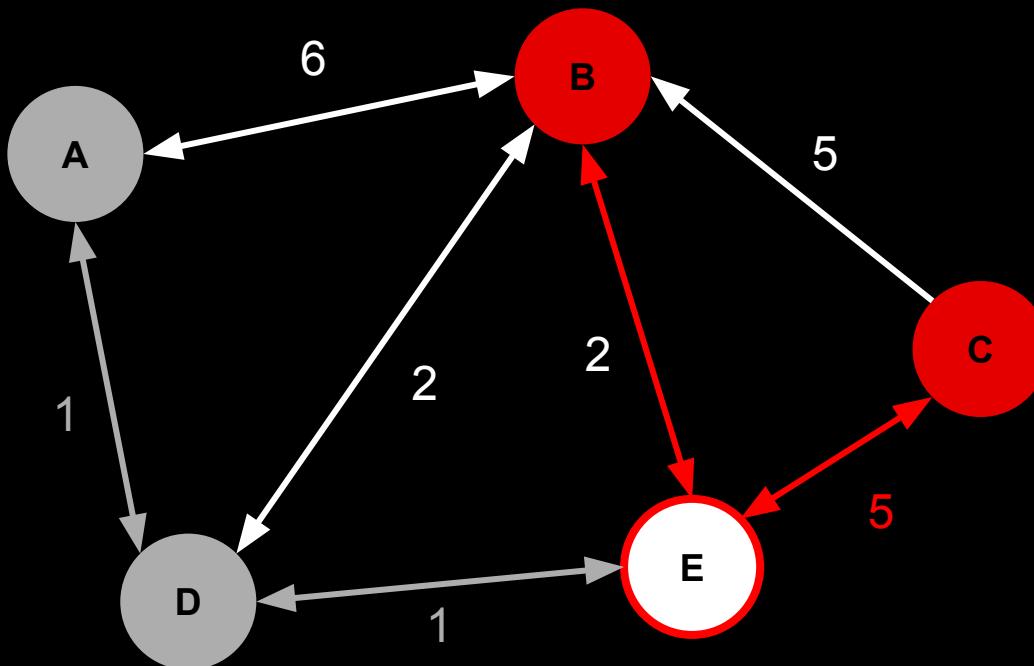
Distances: A = 0, B = 3, C = 7, D = 1, E = 2

Update distance
to C to 7



Distances: A = 0, B = 3, C = 7, D = 1, E = 2

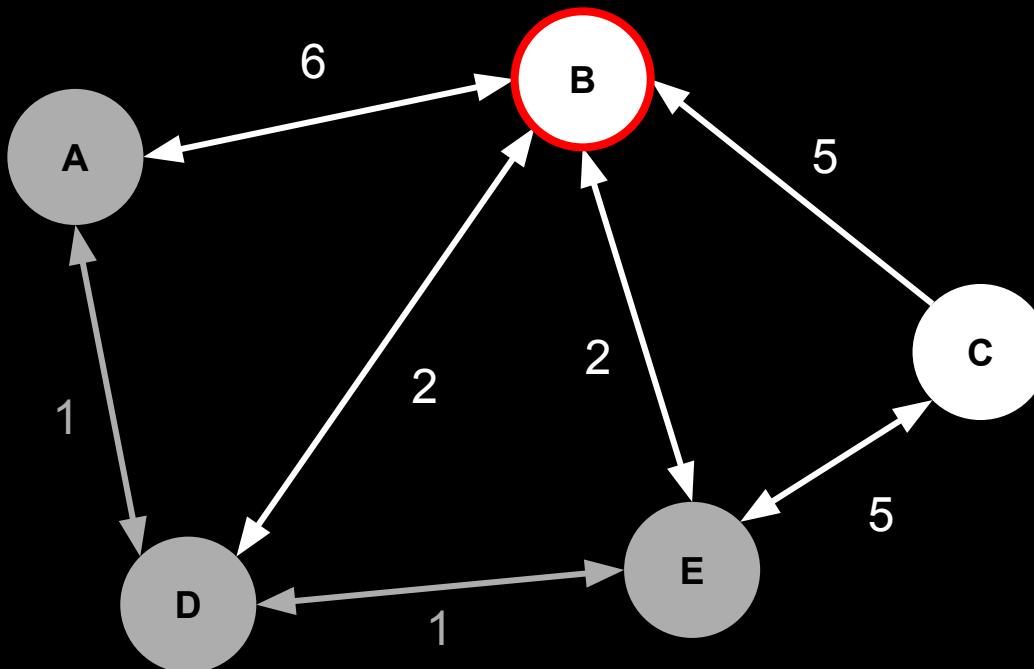
No update for B,
shorter path
exists



Distances: A = 0, B = 3, C = 7, D = 1, E = 2

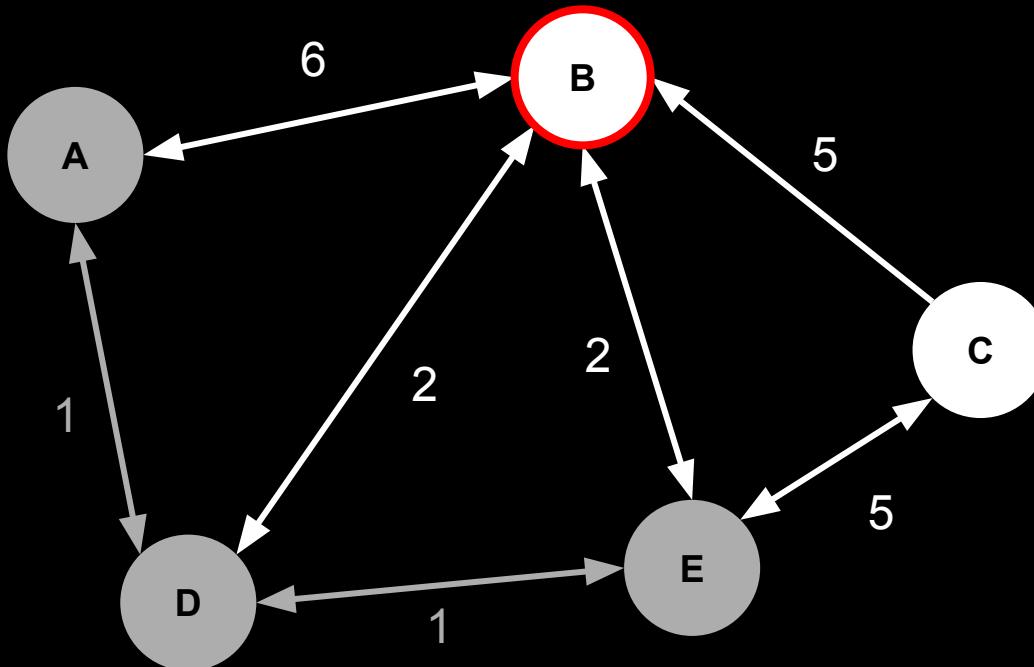
Move to B as
new location

Mark E as
visited



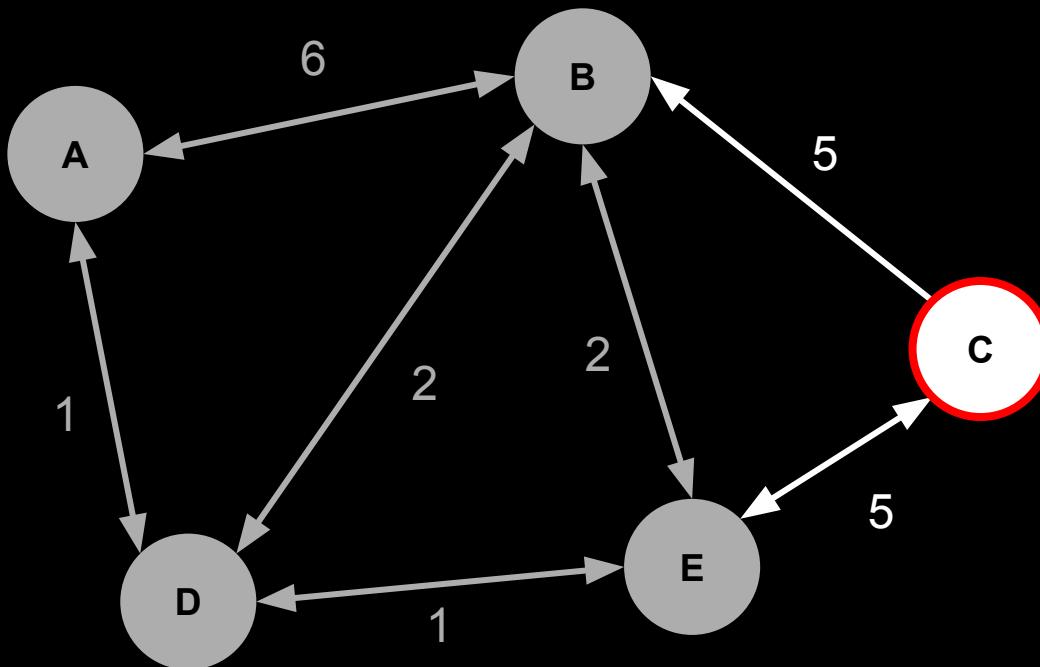
Distances: A = 0, B = 3, C = 7, D = 1, E = 2

No further locations
reachable
from B



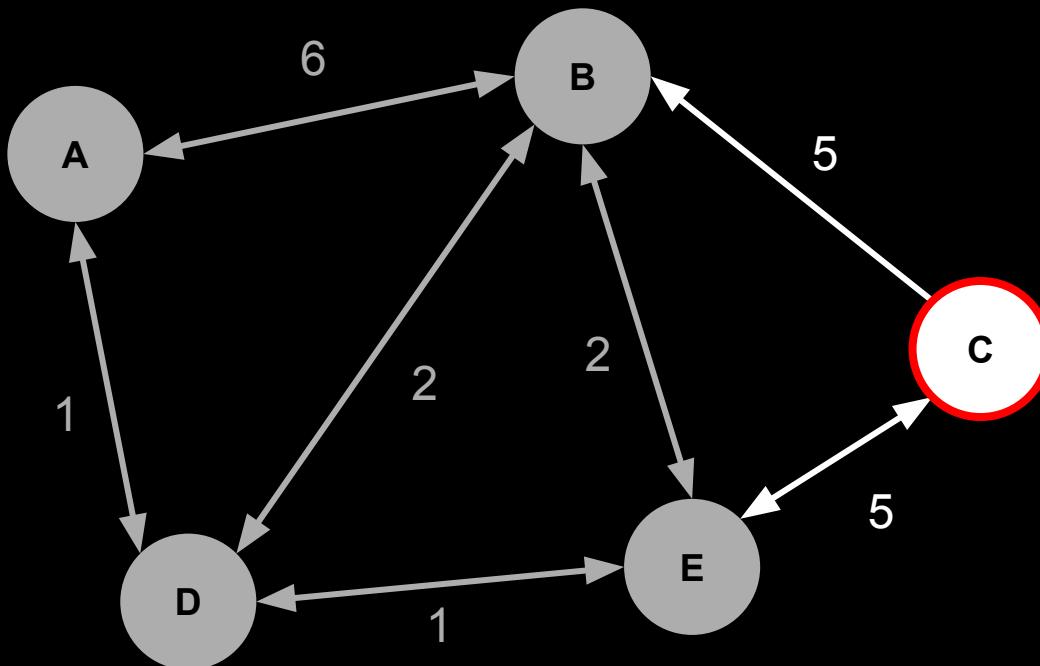
Distances: A = 0, B = 3, C = 7, D = 1, E = 2

Move to C as
new location



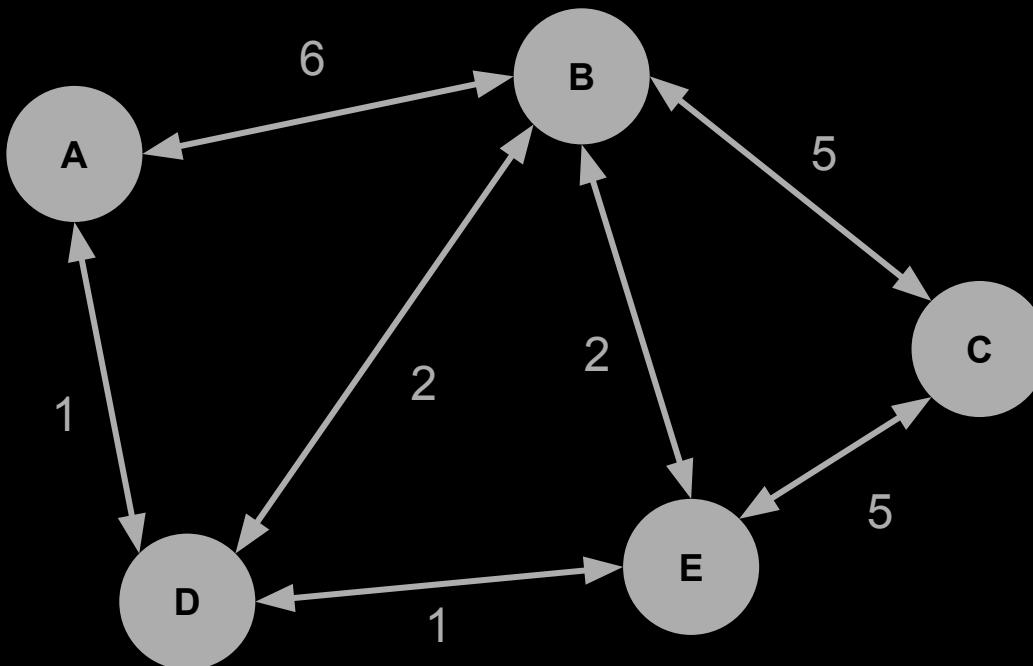
Distances: A = 0, B = 3, C = 7, D = 1, E = 2

No further
locations
reachable from
C



Distances: A = 0, B = 3, C = 7, D = 1, E = 2

All nodes
visited, we're
done!





spam emails



finding oranges in images

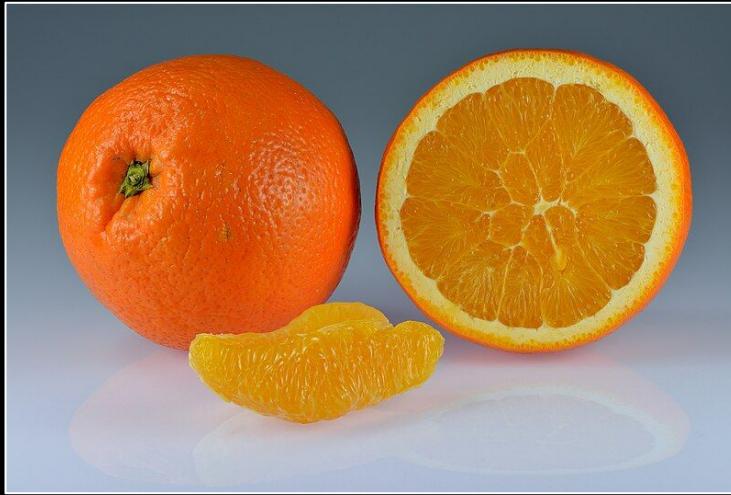


Image source: [Wikimedia](#)

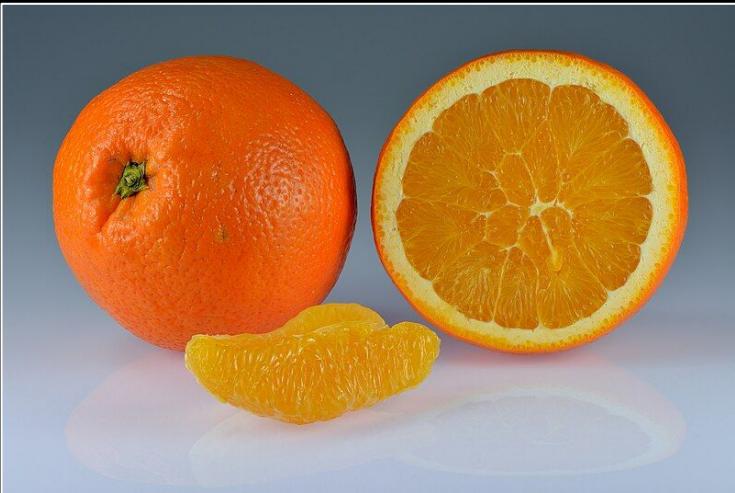


Image source: [Wikimedia](#)

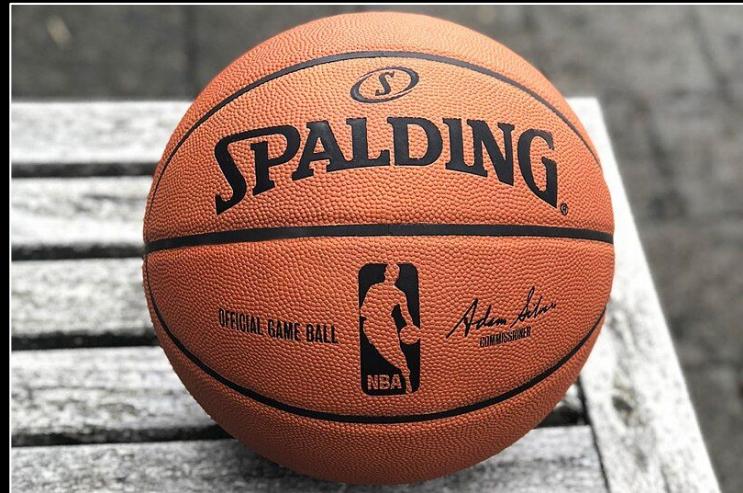


Image source: [Wikimedia](#)

what set of rules can solve this?

machine learning algorithms

