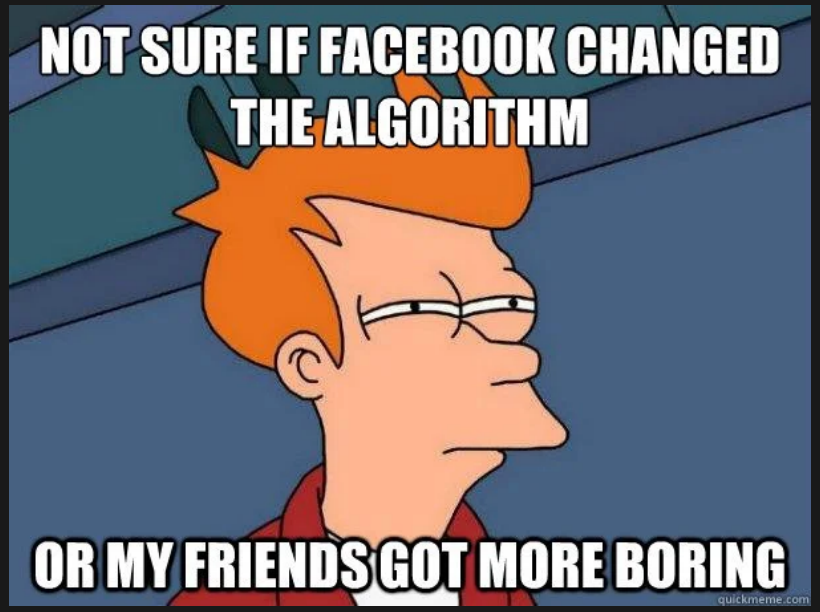
PROGRAMMING BASICS - 2021-07-13

ALGORITHMS

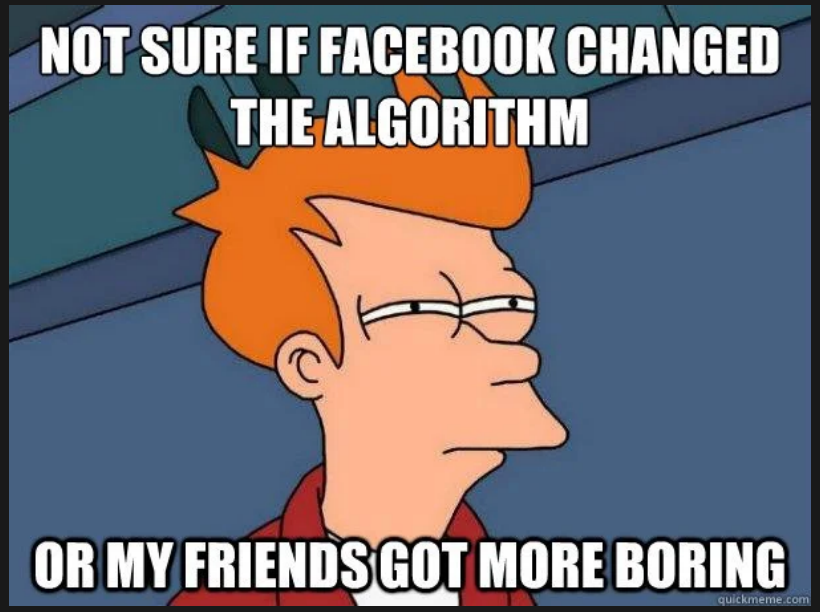
What is an algorithm?



ALGORITHMS

What is an algorithm?

Does it have anything to do with social networks?



ALGORITHMS

What does the word algorithm even mean?

ALGORITHMS

What does the word algorithm even mean?

Nothing, really.

ALGORITHMS

It comes from this guy:

Muḥammad ibn Mūsā al-Khwārizmī:

ALGORITHMS

It comes from this guy:

Muḥammad ibn Mūsā al-Khwārizmī:

- a Persian mathematician, astronemer and geographer of the 9th century

ALGORITHMS

It comes from this guy:

Muḥammad ibn Mūsā al-Khwārizmī:

- a Persian mathematician, astronemer and geographer of the 9th century
- popularised the decimal positional number system into the Western world

ALGORITHMS

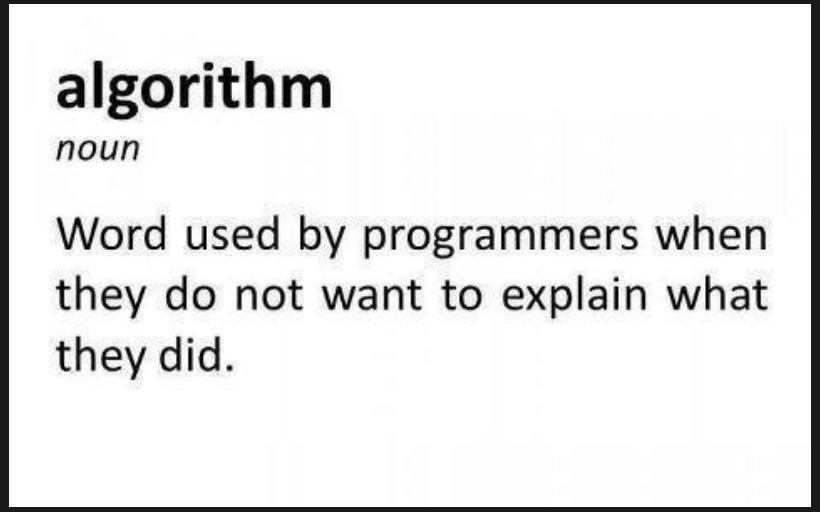
It comes from this guy:

Muḥammad ibn Mūsā al-Khwārizmī:

- a Persian mathematician, astronemer and geographer of the 9th century
- popularised the decimal positional number system into the Western world
- his name al-Khwārizmī was latinized to "Algorithmi"

ALGORITHMS

A possible definition:



ALGORITHMS

ALGORITHMS

- a finite set of precise instructions for performing a computation or for solving a problem

ALGORITHMS

- a finite set of precise instructions for performing a computation or for solving a problem
- a sequence of unambiguous instructions for solving a problem, to get an output from an input in a finite amount of time

ALGORITHMS

- a finite set of precise instructions for performing a computation or for solving a problem
- a sequence of unambiguous instructions for solving a problem, to get an output from an input in a finite amount of time
- a process or set of rules to be followed in calculations or other problem-solving operations

ALGORITHMS

- a finite set of precise instructions for performing a computation or for solving a problem
- a sequence of unambiguous instructions for solving a problem, to get an output from an input in a finite amount of time
- a process or set of rules to be followed in calculations or other problem-solving operations
- a set of step by step rules for solving a problem

ALGORITHMS

- a finite set of precise instructions for performing a computation or for solving a problem
- a sequence of unambiguous instructions for solving a problem, to get an output from an input in a finite amount of time
- a process or set of rules to be followed in calculations or other problem-solving operations
- a set of step by step rules for solving a problem
- a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of problems or to perform a computation.

ALGORITHMS

- step-by-step
- precise, unambiguous
- finite
- input, output
- problem solving

ALGORITHMS

- given an input (even an empty one) algorithms produce an output, a result
- algorithm steps must be finite and unambiguous
- execution time must be finite

ALGORITHMS

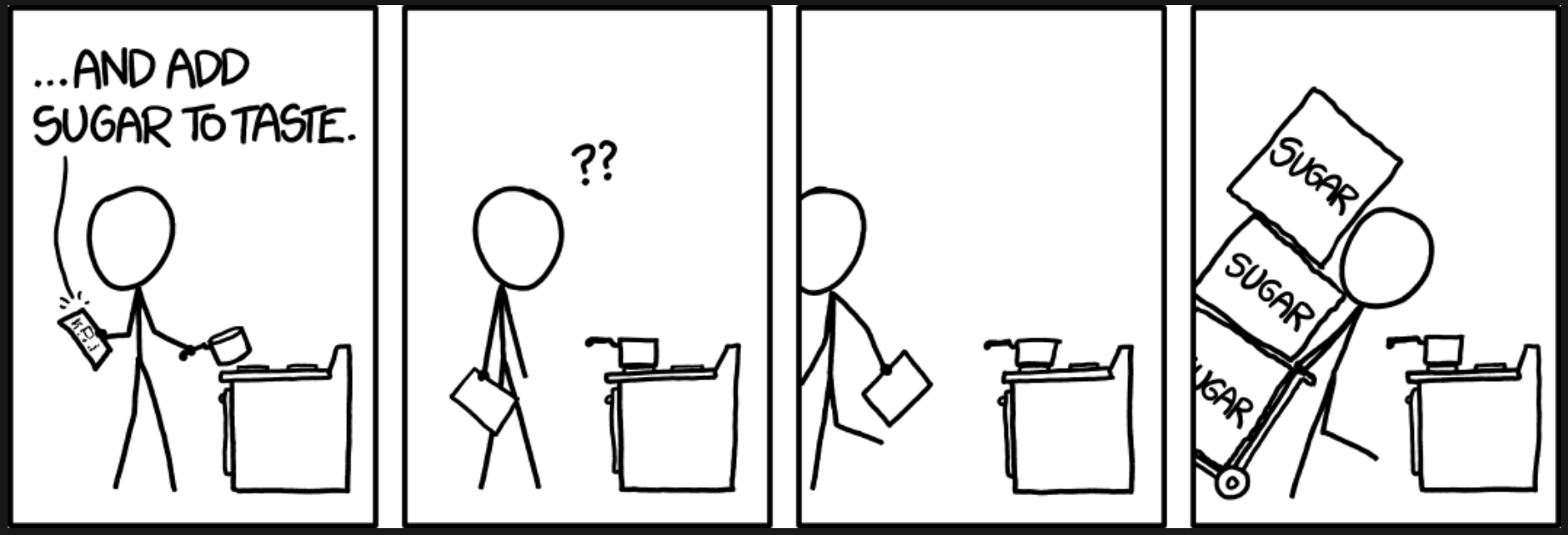
A METAPHOR

Pizza recipe

- input -> ingredients
  - (flour, water, salt, oil, tomatoes, mozzarella)
- instructions
  - mix ingredients; prepare the dough; cook
- output -> pizza

ALGORITHMS

Unlike proper algorithms, recipes have often vague instructions:



"Look, recipe, if I knew how much was gonna taste good, I wouldn't need you."

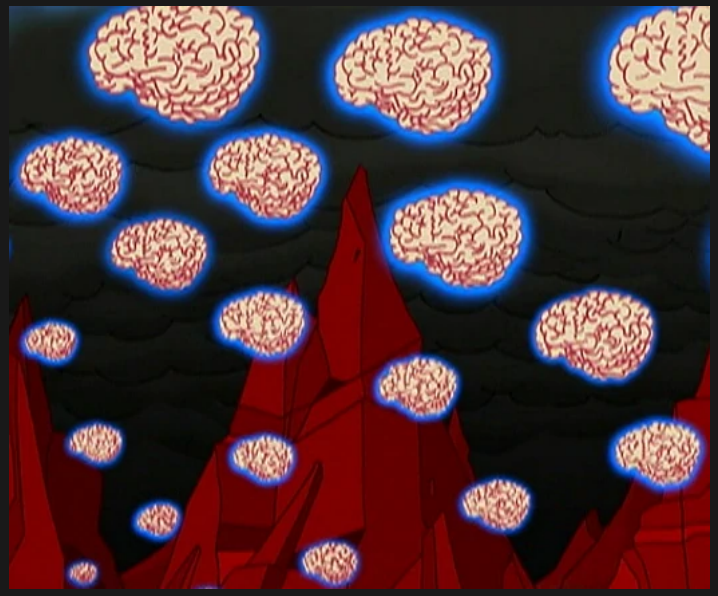
ALGORITHMS

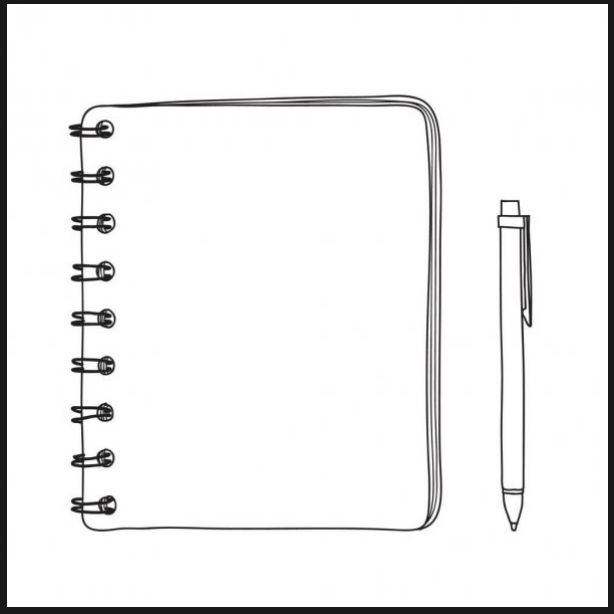
And if you put pineapple on pizza your algorithm is wrong.

ALGORITHMS

Tools:

- no programming language
- no computer

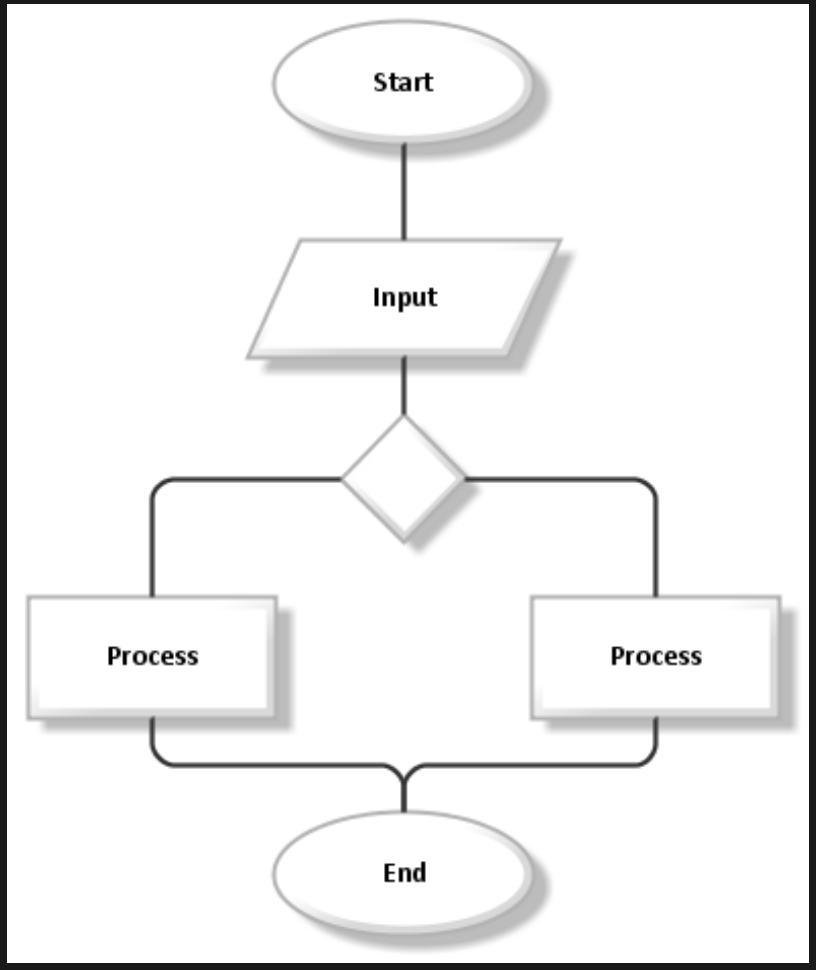




ALGORITHMS

What does an algorithm look like?

Flowchart



ALGORITHMS

Pseudocode

```
// count even numbers in a list

numbers = 1, 2, 3, 4, 5, 6, 7, 8, 9
evenTotal = 0

for each number in numbers
  if number is even
    evenTotal++

Answer is: evenTotal
```

for = go through

ALGORITHMS

Even "pseudoer" code

```
// count even numbers in a list

start with a list numbers with those values: 1, 2, 3, 4, 5, 6,
The count of even numbers starts at 0

Go through all the numbers one by one
  if a number is even
     increase the count of even numbers
  otherwise do not do anything
```

As long as instructions are precise enough, doesn't matter how you write them

ALGORITHMS

Algorithm examples

- specific calculations
- manipulating list of numbers
- sorting lists
- finding a specific value in a list of numbers (min, max, avg etc.)
- generating values (Fibonacci sequence, even numbers etc.)

ALGORITHMS

Why studying algorithms?

- problem solving skills
- learn common programming patterns
- learn to think like a computer

ALGORITHMS: PRACTICAL EXAMPLE

Find the minimum value from a list

```
list = 9, 10, 2, 7
```

ALGORITHMS: PRACTICAL EXAMPLE

Find the minimum value from a list

```
list = 9, 10, 2, 7
```

Finding the result is quite easy and obvious, but the point of algorithms is not (only) the result, but the process

ALGORITHMS: PRACTICAL EXAMPLE

Try to think like a computer

```
list = 9, 10, 2, 7
```

It's easy for us to view the list all at once and come up with the correct answer.

A computer can't see numbers, it can only read them, one by one

ALGORITHMS: PRACTICAL EXAMPLE

A better way to approach the problem is this:

ALGORITHMS: PRACTICAL EXAMPLE

A better way to approach the problem is this:

- there is a list of 4 numbers

ALGORITHMS: PRACTICAL EXAMPLE

A better way to approach the problem is this:

- there is a list of 4 numbers
- you have to find the minimum (the number with the smallest value)

ALGORITHMS: PRACTICAL EXAMPLE

A better way to approach the problem is this:

- there is a list of 4 numbers
- you have to find the minimum (the number with the smallest value)
- you receive the numbers only one at a time

ALGORITHMS: PRACTICAL EXAMPLE

A better way to approach the problem is this:

- there is a list of 4 numbers
- you have to find the minimum (the number with the smallest value)
- you receive the numbers only one at a time

You might still think: with only 4 numbers, I can wait till the end and find the answer at that point

ALGORITHMS: PRACTICAL EXAMPLE

ALGORITHMS: PRACTICAL EXAMPLE

What if the list contains 100 numbers? Or maybe 1.000.000?

ALGORITHMS: PRACTICAL EXAMPLE

What if the list contains 100 numbers? Or maybe 1.000.000?

With so many numbers it's clear that we need a better plan, some kind of process.

ALGORITHMS: PRACTICAL EXAMPLE

What if the list contains 100 numbers? Or maybe 1.000.000?

With so many numbers it's clear that we need a better plan, some kind of process.

We need an algorithm.

ALGORITHMS: PRACTICAL EXAMPLE

Let's start from the things we know for certain

ALGORITHMS: PRACTICAL EXAMPLE

Let's start from the things we know for certain

- one way or another, we have to scan the whole list

ALGORITHMS: PRACTICAL EXAMPLE

Let's start from the things we know for certain

- one way or another, we have to scan the whole list
  - the minimum might be the first number, but it might as well be in the middle or at the end

ALGORITHMS: PRACTICAL EXAMPLE

Let's start from the things we know for certain

- one way or another, we have to scan the whole list
  - the minimum might be the first number, but it might as well be in the middle or at the end
  - even if we find a 1 or a 0 right away, nothing tells us that there are no smaller (possibly negative) numbers later

ALGORITHMS: PRACTICAL EXAMPLE

Let's start from the things we know for certain

- one way or another, we have to scan the whole list
  - the minimum might be the first number, but it might as well be in the middle or at the end
  - even if we find a 1 or a 0 right away, nothing tells us that there are no smaller (possibly negative) numbers later
- if we find a number which is bigger than one other number, we can discard it

ALGORITHMS: PRACTICAL EXAMPLE

A possible approach:

ALGORITHMS: PRACTICAL EXAMPLE

A possible approach:

- go through all the numbers

ALGORITHMS: PRACTICAL EXAMPLE

A possible approach:

- go through all the numbers
- compare each number with the one next to it

ALGORITHMS: PRACTICAL EXAMPLE

A possible approach:

- go through all the numbers
- compare each number with the one next to it
- discard the ones that are bigger

ALGORITHMS: PRACTICAL EXAMPLE

A possible approach:

- go through all the numbers
- compare each number with the one next to it
- discard the ones that are bigger
- repeat

ALGORITHMS: PRACTICAL EXAMPLE

A possible approach:

- go through all the numbers
- compare each number with the one next to it
- discard the ones that are bigger
- repeat

Does it work?

ALGORITHMS: PRACTICAL EXAMPLE

ALGORITHMS: PRACTICAL EXAMPLE

It might work, but with this approach we have to go through the list many many times.

ALGORITHMS: PRACTICAL EXAMPLE

It might work, but with this approach we have to go through the list many many times.

Algorithms should be as efficient as possible.

ALGORITHMS: PRACTICAL EXAMPLE

It might work, but with this approach we have to go through the list many many times.

Algorithms should be as efficient as possible.

Is there a way to go through the list only once?

ALGORITHMS: PRACTICAL EXAMPLE

Another approach:

ALGORITHMS: PRACTICAL EXAMPLE

Another approach:

We go through the whole list only once and as we go we write down each new minimum value we encounter.

ALGORITHMS: PRACTICAL EXAMPLE

Another approach:

We go through the whole list only once and as we go we write down each new minimum value we encounter.

This way, we should be able to go through the list only once and get the minimum at the end.

ALGORITHMS: PRACTICAL EXAMPLE

Another approach:

We go through the whole list only once and as we go we write down each new minimum value we encounter.

This way, we should be able to go through the list only once and get the minimum at the end.

How do we write that with precise instructions?

ALGORITHMS: PRACTICAL EXAMPLE

The first step is to go through the numbers one by one.

We read the first number. It's 563.

ALGORITHMS: PRACTICAL EXAMPLE

The first step is to go through the numbers one by one.

We read the first number. It's 563.

What do we do about it?

ALGORITHMS: PRACTICAL EXAMPLE

The first step is to go through the numbers one by one.

We read the first number. It's 563.

What do we do about it?

Shall we keep it? Discard it?

ALGORITHMS: PRACTICAL EXAMPLE

The first step is to go through the numbers one by one.

We read the first number. It's 563.

What do we do about it?

Shall we keep it? Discard it?

Is there a way to decide if a number is a potential minimum?

ALGORITHMS: PRACTICAL EXAMPLE

We have to assume that every number is a potential minimum.

We should take out our notebook and write down

```
minimum = 563
```

It seems like a pretty big number to be a minimum, but we have no idea how big are all other numbers.

ALGORITHMS: PRACTICAL EXAMPLE

We move on to the second number. It's 725.

ALGORITHMS: PRACTICAL EXAMPLE

We move on to the second number. It's 725.

What shall we do?

ALGORITHMS: PRACTICAL EXAMPLE

We move on to the second number. It's 725.

What shall we do?

We can check if we have a minimum already, which we do (563). We can then check if `725 < minimum` -> `725 < 563`.

ALGORITHMS: PRACTICAL EXAMPLE

We move on to the second number. It's 725.

What shall we do?

We can check if we have a minimum already, which we do (563). We can then check if `725 < minimum` -> `725 < 563`.

It is not, so we can move on.

ALGORITHMS: PRACTICAL EXAMPLE

The third number is 500.

`500 < minimum`?

Yes, so:

```
minimum = 500
```

And we continue like this.

ALGORITHMS: PRACTICAL EXAMPLE

The whole algorithm would be something like this:

- go through all the numbers in the list, one by one
- read a number
- if we don't have a minimum yet, minimum will be equal to this number
  - if we have a minimum, let's check if the current number is smaller than the current minimum
    - if it's smaller, update the minimum with the current number value

ALGORITHMS: PRACTICAL EXAMPLE

In pseudocode:

```
minimum = null

go through all the numbers one by one
  if we do not have a minimum yet
    set minimum equal to the currentNumber
  otherwise, if we do have a minimum
    check if currentNumber is smaller than minimum
      if it is, set minimum equal to the currentNumber
```

ALGORITHMS: PRACTICAL EXAMPLE

Proper pseudocode:

```
numbers = 523, 750, 500, ...
minimum = null

for each number in numbers:
  if ! minimum || number < minimum:
    minimum = number

print minimum
```

Pseudocode doesn't actually work, but once the logic of an algorithm is clear, implementing in a programming language is much easier, once we have all the tools necessary

ALGORITHMS: RECAP

ALGORITHMS: RECAP

- instructions must be precise

ALGORITHMS: RECAP

- instructions must be precise
- instructions must be broken up in atomic operations

ALGORITHMS: RECAP

- instructions must be precise
- instructions must be broken up in atomic operations
- instructions must be explicit: a computer doesn't do anything that we don't tell it to do

ALGORITHMS: RECAP

- instructions must be precise
- instructions must be broken up in atomic operations
- instructions must be explicit: a computer doesn't do anything that we don't tell it to do
- no pineapple on pizza

ALGORITHMS: CURIOSITY

Ada Lovelace

ALGORITHMS: CURIOSITY

Ada Lovelace

- mathematician and writer

ALGORITHMS: CURIOSITY

Ada Lovelace

- mathematician and writer
- creative genius

ALGORITHMS: CURIOSITY

Ada Lovelace

- mathematician and writer
- creative genius
- at a very young age wrote a book (Flyology) trying to discover a method to fly

ALGORITHMS: CURIOSITY

Ada Lovelace

- mathematician and writer
- creative genius
- at a very young age wrote a book (Flyology) trying to discover a method to fly
- had a cat named Mrs. Puff

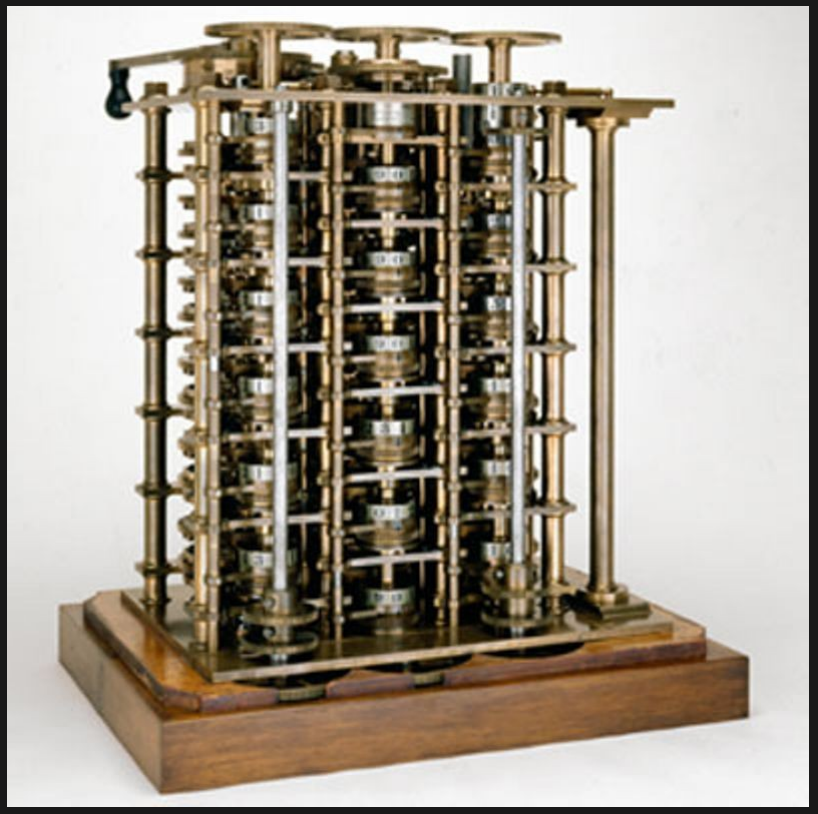
ALGORITHMS: CURIOSITY

Ada Lovelace

- mathematician and writer
- creative genius
- at a very young age wrote a book (Flyology) trying to discover a method to fly
- had a cat named Mrs. Puff
- she wrote the first computers algorithm, 100 years before computers even existed

ALGORITHMS: CURIOSITY

Steampunk computer



ALGORITHMS: PRACTICE

Can you write an algorithm that given the input on the left, produces the output on the right?

Input	Output
2, 5	false
0, 0	true
8, 8	true
9, 7	false
100, 52	false

ALGORITHMS: PRACTICE

```
Let a be the first input.
Let b be the second input.
If a is equal to b, print true.
If a is different than b, print false.
```

LINKS

- Donald Knuth (comic)
- The general problem (comic)
- Learning to cook (comic)
- Recipes (comic)
- Why algorithms are called algorithms? (video)
- Ada Lovelace (video with nice animations, but in German)
- Ada Lovelace (video)
- Untangling the Tale of Ada Lovelace (article)
- Women in computing (wiki page)

CONDITIONAL STATEMENTS

Until now the code we've seen has always been executed from top to bottom, line by line, without skipping anything.

`if/else` statements let us create blocks of code that are executed on certain conditions

CONDITIONAL STATEMENTS

```
if (condition) {
  // this code is executed only if "condition" is true
}
```

Condition can be a boolean or any other valid JavaScript expression that has a truthy value

CONDITIONAL STATEMENTS

Examples:

```
if (hour > 8) {
  console.log("Wake up!");
}

if (day === "Saturday" || day === "Sunday") {
  console.log("Weekend!");
}

if (isPizzaWithPineapple === true) {
  console.log("Get a new pizza");
}
```

CONDITIONAL STATEMENTS

After an if statement code execution continues normally, whether what's inside has been executed or not

```
let money = 10;
let applePrice = 2;
let breadPrice = 3;

console.log("Let's buy 4 apples");
money = money - (applePrice * 4);

if (money > breadPrice) {
  console.log("Let's buy some bread");
  money = money - breadPrice;
}

console.log("Money left:", money);
```

CONDITIONAL STATEMENTS

An if statement can be combined with an optional else block

```
if (condition) {
  // this is executed if condition is true
} else {
  // this is executed if condition is false
}
```

The else block never has a condition, it's executed automatically if the first condition is false

CONDITIONAL STATEMENTS

```
let hour = 7;
if (hour > 8) {
  console.log("Wake up!");
} else {
  console.log("Keep sleeping");
}
```

CONDITIONAL STATEMENTS

With the if/else statement, we can have more than one alternative:

```
if (condition) {
  // do something
} else if (otherCondition) {
  // do something else
} else if (yetAnotherCondition) {
  // do something something else
} else {
  // or do this
}
```

Conditions are evaluated from top to bottom. If none are true, the code inside else is executed.

CONDITIONAL STATEMENTS

```
if (day === 'Monday') {
  console.log("😱");
} else if (day === "Tuesday" || day === "Wednesday") {
  console.log("😩");
} else if (day === "Thursday") {
  console.log("🤯 Almost there");
} else if (day === "Friday") {
  console.log("🍸🍺🍹");
} else {
  console.log("🛌🛌🛌🥳🎉🎉🎉🥳");
}
```

CONDITIONAL STATEMENTS

What happens if we create a variable inside an if statement?

```
if (tired === true) {
  let action = "Take a nap";
}

console.log(action);
```

CONDITIONAL STATEMENTS

What happens if we create a variable inside an if statement?

```
if (tired === true) {
  let action = "Take a nap";
}

console.log(action);
```

`action` is undefined

SWITCH STATEMENTS

Similar to the if/else statement, there is the `switch` statement

```
switch(day) {
  case "Monday":
    console.log("😱");
    break;
  case "Tuesday":
  case "Wednesday":
    console.log("😩");
    break;
  case "Thursday":
    console.log("🤯 Almost there");
    break;
  case "Friday":
    console.log("🍸🍺🍹");
    break;
  default:
```