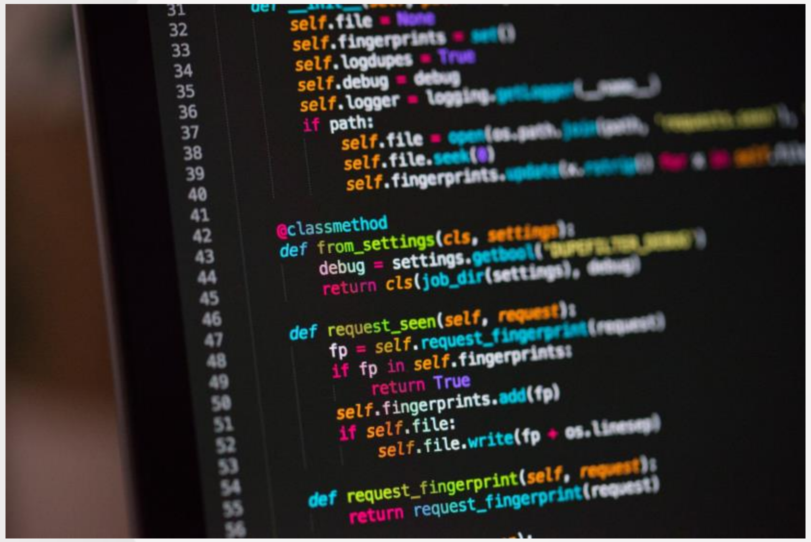
Level 5 Data Engineer Module 3 Topic 6

Algorithmic thinking



L5 Data Engineer Higher Apprenticeship
Module 3 / 12 (**“Programming and Scripting Essentials”**)
Topic 7 / 9

Algorithms in real-life

An algorithm 'recipe'

Chocolate Chip Cookies

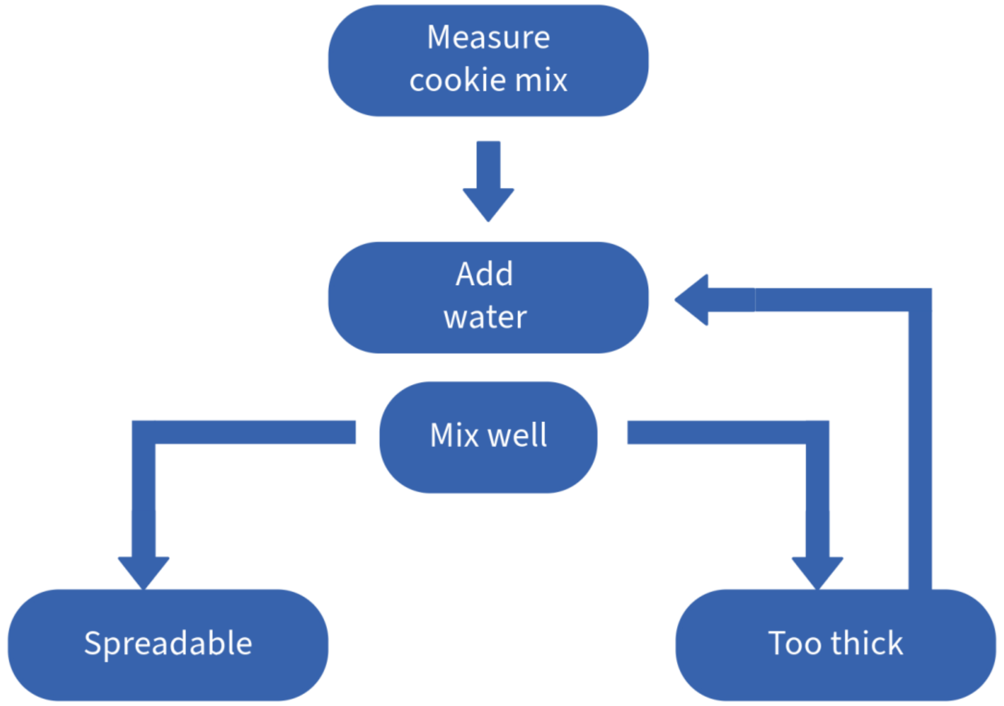
DATA (ingredients)

2 1/4 cups flour	1 tsp salt
1 tsp baking soda	2 eggs
3/4 cup brown sugar	1 tsp vanilla ext.
3/4 cup gran'd sugar	1 cup soft butter

12oz. semi-sweet chocolate chips

CODE (steps)

1. Preheat oven to 180° C
2. Combine flour, salt, baking soda, in bowl, set aside
3. Combine sugars, butter, vanilla, beat until creamy
4. Add eggs and beat more
5. Add dry mixture and mix well
6. Stir in chocolate chips
7. Drop mix by teaspoons onto ungreased cookie sheet
8. Bake 8-10 minutes*



A refinement to the 'algorithm'

*** what's the issue with this instruction?**

Algorithms in real-life

Adaptive learning technology

- **Standardised Testing and Algorithms** are closely linked
- **Coding** is the foundation for adaptive technology used in classrooms
- If students select the **correct answer** to a question, the next question is **moderately more difficult**
- This process is driven by an **iterative algorithm** that starts with a pool of questions
- After each answer, the **pool of questions is adjusted** accordingly
- This **adjustment process repeats continuously** throughout the assessment (We call that a LOOP)

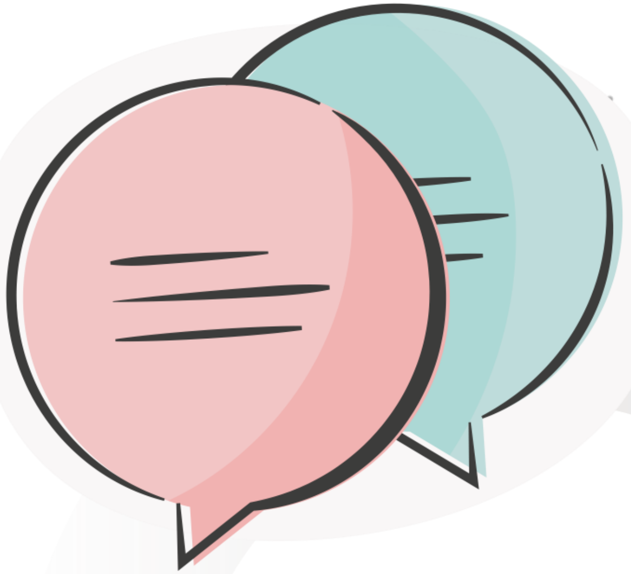
Adaptive learning technology

Spotlight on your experience

Share your thoughts on the following questions:

- Can you share an example of a project where you had to use a specific algorithm? Why was that algorithm chosen?
- How do data structures relate to algorithms in your work? Can you give an example?
- What challenges have you faced when implementing algorithms in your projects?

Submit any thoughts to the chat!



Discussion activity

Session aim and objectives

Building Careers Through Education

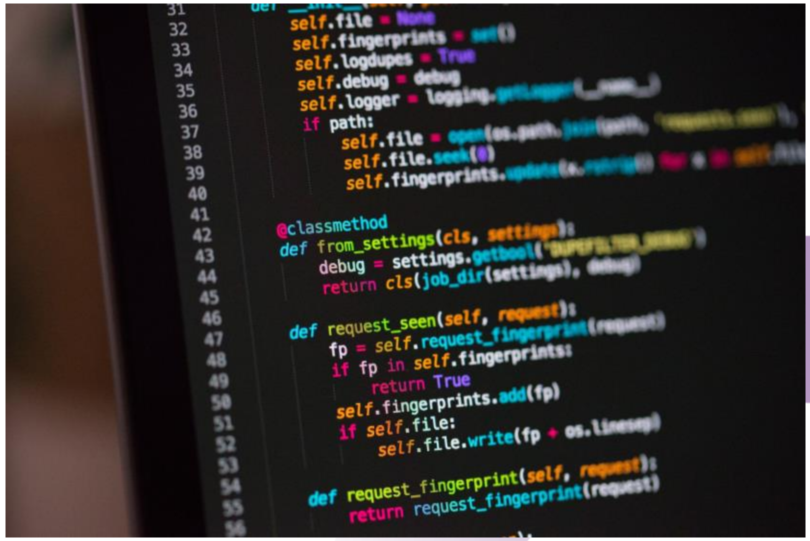
Everybody Matters. Stronger Together. Student, Learner and Client Centric. Trust and Respect. Embrace Change.

Completion of this topic supports the following outcomes:

- Employ software development tools and techniques for designing, deploying and maintaining secure data products and pipelines, including debugging, version control and testing
- Construct algorithms that correctly and efficiently handle data at scale whilst mitigating risks
- Demonstrate the knowledge of the steps needed to prepare the code for production

BPP

Computer science and programming



```
    def [illegible]
        self.file = None
        self.fingerprints = set()
        self.logdupes = True
        self.debug = debug
        self.logger = logging.getLogger(__name__)
        if path:
            self.file = open(os.path.join(path, [illegible]), [illegible]
            self.file.seek(0)
            self.fingerprints.update([illegible]

    @classmethod
    def from_settings(cls, settings):
        debug = settings.getbool([illegible])
        return cls(job_dir(settings), debug)

    def request_seen(self, request):
        fp = self.request_fingerprint(request)
        if fp in self.fingerprints:
            return True
        self.fingerprints.add(fp)
        if self.file:
            self.file.write(fp + os.linesep)

    def request_fingerprint(self, request):
        return request_fingerprint(request)
```

Knowledge Check Poll

What are the properties of a good algorithm in computer science?

A. A set of data for a computer to store

B. A list of rules to follow in order to complete a task or solve a problem

C. The process of writing computer code to create a program, in order to solve a problem

D. The study of computer hardware and software

Submit your responses to the chat!

Knowledge Check Poll

What are the properties of a good algorithm in computer science?

A. A set of data for a computer to store

B. A list of rules to follow in order to complete a task or solve a problem

C. The process of writing computer code to create a program, in order to solve a problem

D. The study of computer hardware and software

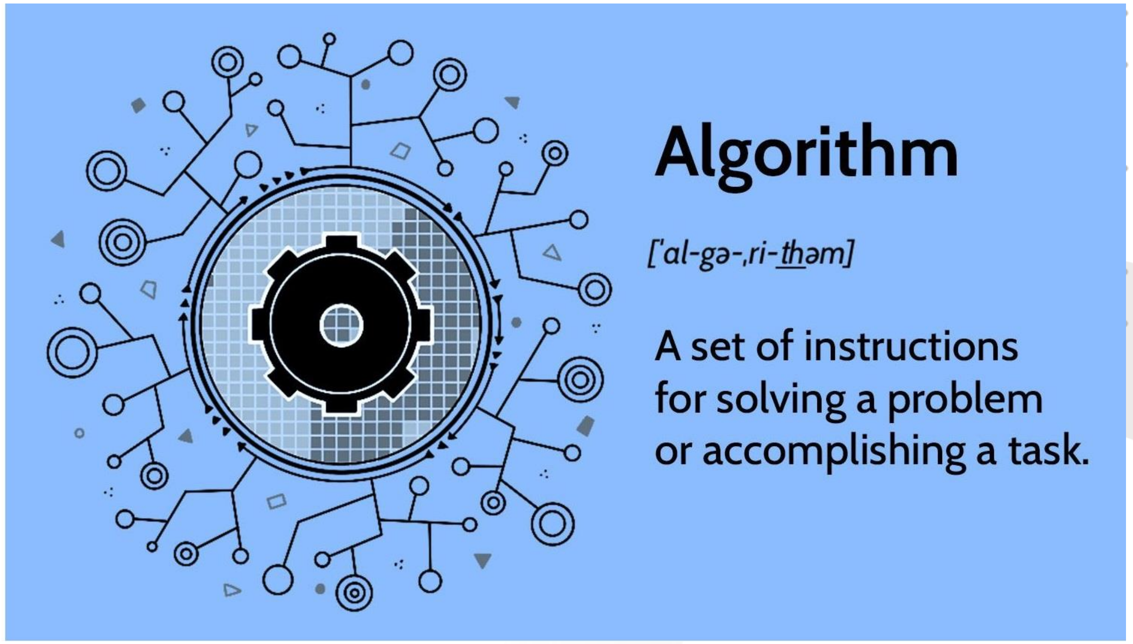
Submit your responses to the chat!

Feedback
The correct statement is **B** – Algorithms can be thought of as simply a A list of rules to follow in order to complete a task or solve a problem.

The algorithmic model

Let’s break this down by answering these questions:

- What is Computer Science?
- What is Programming?
- What are Algorithms?
- What are the properties of good algorithms?
- What are the components of Algorithms?


Algorithm
['al-gə-ˌri-t͟həm]
A set of instructions for solving a problem or accomplishing a task.

So what is programming, really?

Let’s break this concept down…

- **Programming requires two skills:**

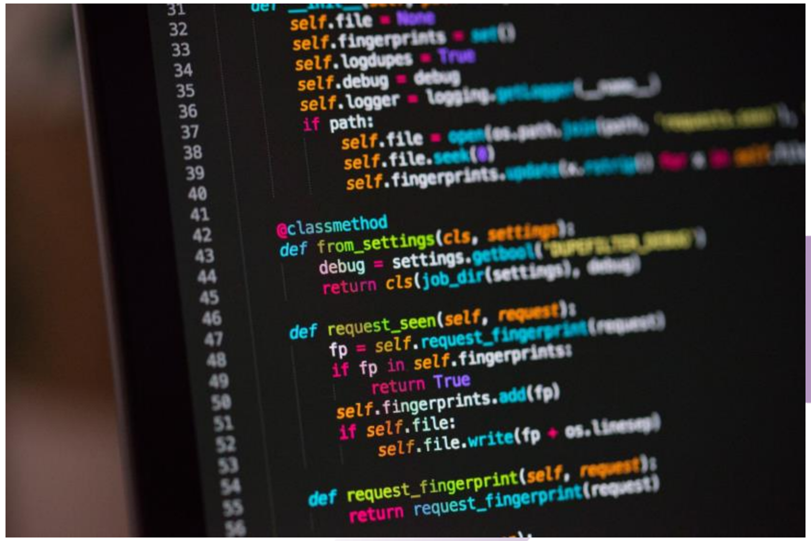
1. Algorithmic thinking
2. Knowledge of programming language syntax

To be able to turn your pseudocode into actual code, you need to understand the syntax of your chosen programming language.

Programming relies heavily on abstraction (next slides)

```
});
describe('Key', () => {
  it('Should be a string', () => {
    expect(Project.getFields().key.type.ofType).toBe(GraphQLString);
  });
  it('Should be non null', () => {
    expect(Project.getFields().key.type.constructor).toBe(GraphQLNonNull);
  });
});
describe('Name', () => {
  it('Should be a string', () => {
    expect(Project.getFields().name.type.ofType).toBe(GraphQLString);
  });
  it('Should be non null', () => {
    expect(Project.getFields().name.type.constructor).toBe(GraphQLNonNull);
  });
});
describe('Sort', () => {
  it('Should be an integer', () => {
    expect(Project.getFields().sort.type).toBe(GraphQLInt);
  });
});
describe('Type', () => {
  it('Should be a string', () => {
    expect(Project.getFields().type.type).toBe(GraphQLString);
  });
});
```

Abstraction

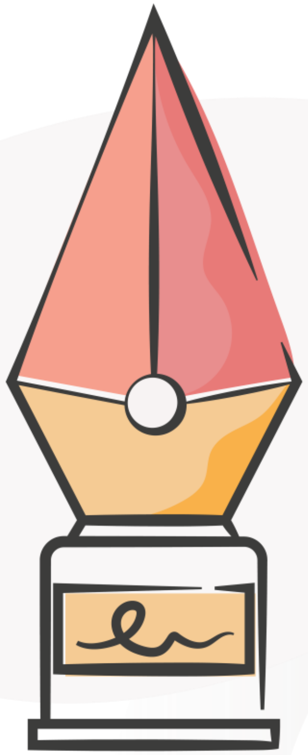


```
    def __init__(self, [illegible]
        self.file = None
        self.fingerprints = set()
        self.logdupes = True
        self.debug = debug
        self.logger = logging.getLogger(__name__)
        if path:
            self.file = open(os.path.join(path, 'requests.seen'),
            self.file.seek(0)
            self.fingerprints.update(x.rstrip() for x in self.fi

    @classmethod
    def from_settings(cls, settings):
        debug = settings.getbool('DUPEFILTER_DEBUG')
        return cls(job_dir(settings), debug)

    def request_seen(self, request):
        fp = self.request_fingerprint(request)
        if fp in self.fingerprints:
            return True
        self.fingerprints.add(fp)
        if self.file:
            self.file.write(fp + os.linesep)

    def request_fingerprint(self, request):
        return request_fingerprint(request)
```

Abstraction

What is meant by this term?

- A general idea or term
- An impractical idea; visionary and unrealistic
- General characteristics apart from concrete realities, specific objects or actual instances
- Absent-mindedness; inattention
- A work of art

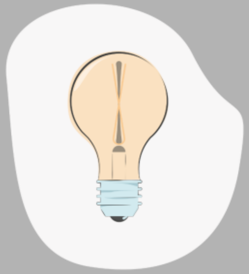


Definition: *Random house dictionary of the English Language*

Abstraction in computing

What does this involve…?

- Refers to the logical grouping of concepts or objects
- Define/implement the **general idea**
- Isolate the details
- Helps **readability and understanding** of our algorithms



Abstraction examples:

Algorithms, variable names, procedures and functions, data structures, and the computer language itself.

Abstraction in algorithms

An example

The Abstraction:

Get to the College of Computing at Georgia Tech

One Implementation:

1. Begin
2. Get on I-85 heading towards Midtown Atlanta
3. Exit I-85 at the 10th street exit
4. Proceed along exit ramp to 10th street
5. Turn west on 10th street
6. Turn left on Atlantic Drive
7. Stop in front of CoC building end

Abstraction examples:

Other implementations of the same abstraction can be just as correct, better, or worse.

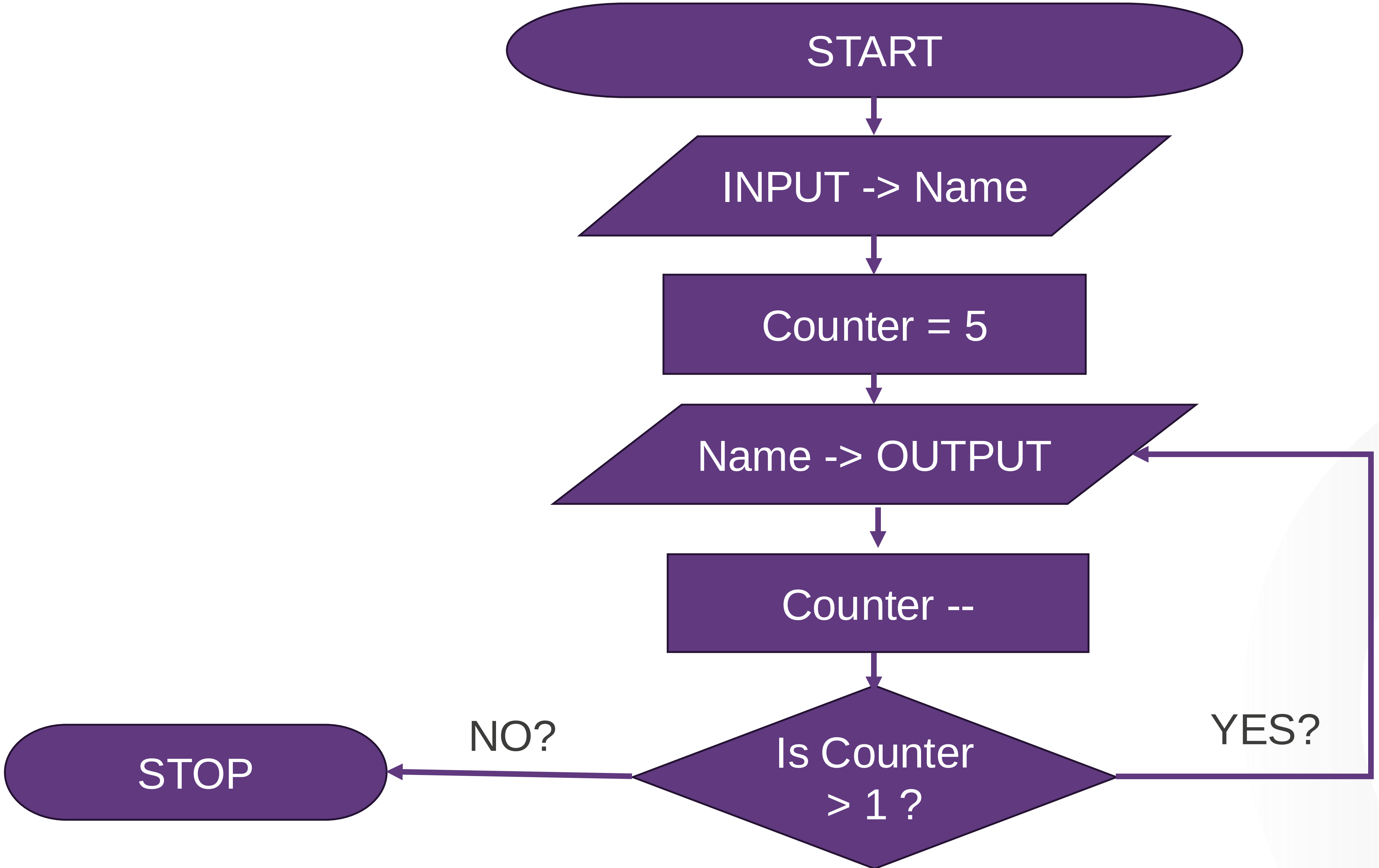
Abstraction (via a brief of a spec)

Let’s say we want to write somebody’s name five times on a computer screen.

Implementation (via an algorithm)

1. Read the name from the input source.
2. Set the counter to 5
3. Write the name on the output destination.
4. Remove 1 from the counter
5. If the count is more than 0, go to step 2.

A flowchart for our example algorithm



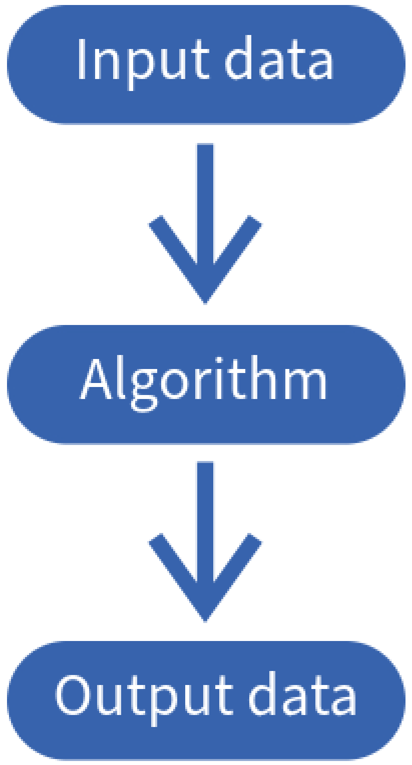
Algorithms in computing

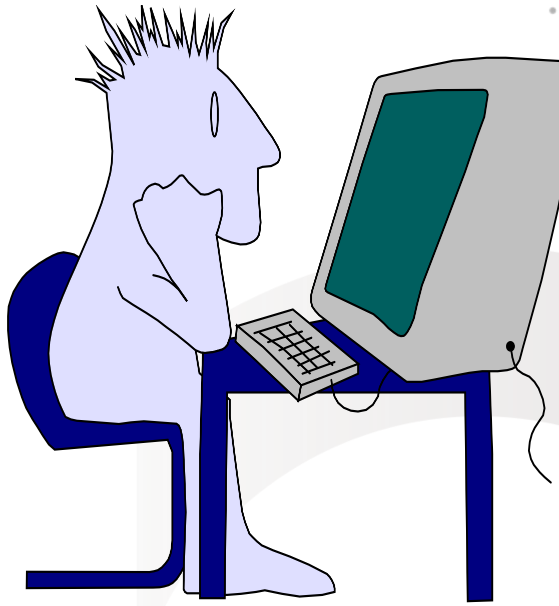
Let’s break this concept down…

- A set of logical steps to accomplish a task
- One way to solve a problem
- A “recipe of action”
- A way of describing behavior

Algorithms contain:

- Data
- Instructions



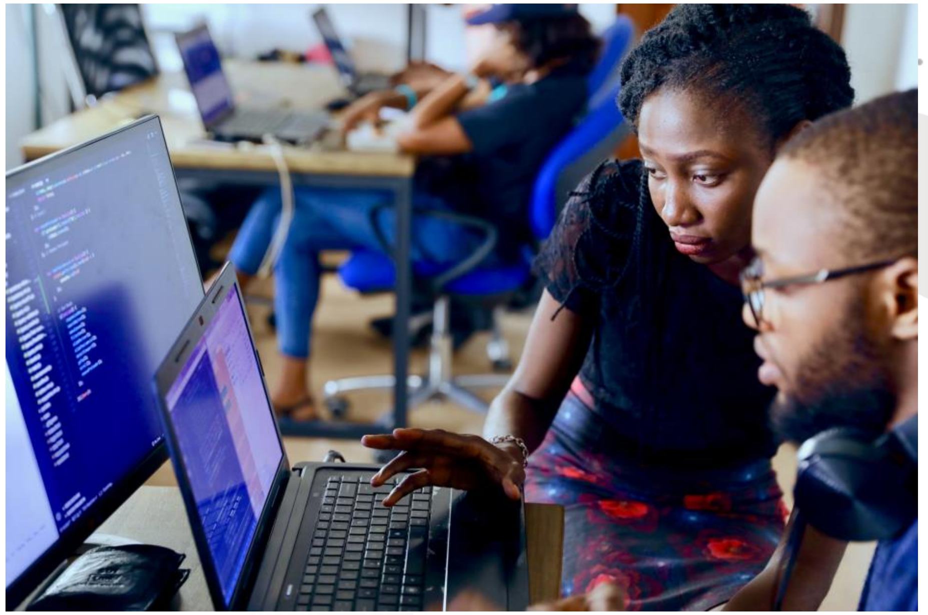


The basic process of an algorithm

So what is computer science?

And how can we learn it?

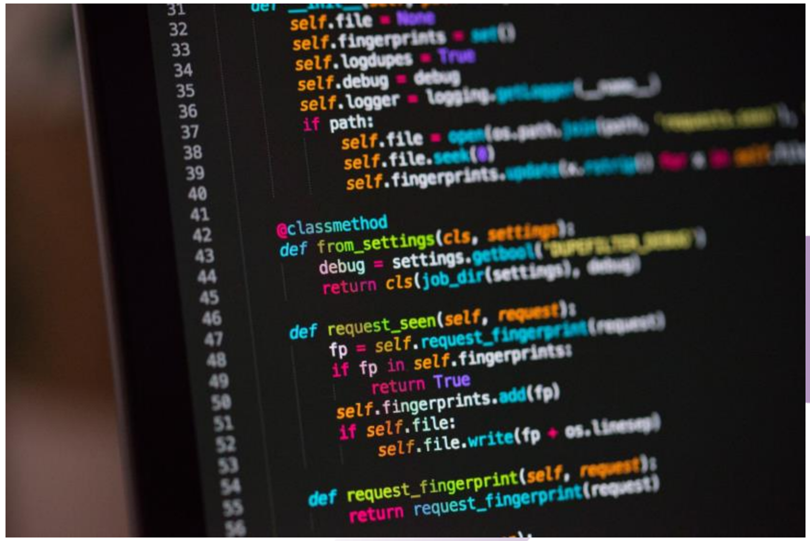
- It is a **study of algorithms** rather than study of computers!
- Often uses **pseudo-code** to focus on **algorithmic thinking**
- Not using a computer to compile and run your program forces you to **mentally execute your program**



Pseudocode example

```
1. Make a list of courses you want to register for, in order of priority
2. Start with empty schedule.  Number of hours = 0.
3. Choose highest priority class on list.
4. If the chosen class is not full and its class time does not conflict with classes
   already scheduled, then register for the class (2 steps):
     • 4.a. Add the class to the schedule
     • 4.b. Add the class hours to the number of hours scheduled
5. Cross that class off of your list.
6. Repeat steps 3 through 5 until the number of hours scheduled is >= 15, or until
   all classes have been crossed out.
7. Stop.
```

Understanding algorithms



```
        self.file = None
        self.fingerprints = set()
        self.logdupes = True
        self.debug = debug
        self.logger = logging.getLogger(__name__)
        if path:
            self.file = open(os.path.join(path, 'requests.seen'),
            self.file.seek(0)
            self.fingerprints.update(x.rstrip() for x in self.

    @classmethod
    def from_settings(cls, settings):
        debug = settings.getbool('DUPEFILTER_DEBUG')
        return cls(job_dir(settings), debug)

    def request_seen(self, request):
        fp = self.request_fingerprint(request)
        if fp in self.fingerprints:
            return True
        self.fingerprints.add(fp)
        if self.file:
            self.file.write(fp + os.linesep)

    def request_fingerprint(self, request):
        return request_fingerprint(request)
```

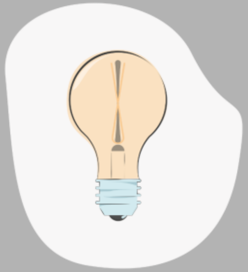
Discussion activity

What's wrong with this algorithm…?

(From back of shampoo bottle)

Directions:

1. Wet Hair
2. Apply a small amount of shampoo
3. Lather
4. Rinse
5. Repeat



A better version would be:

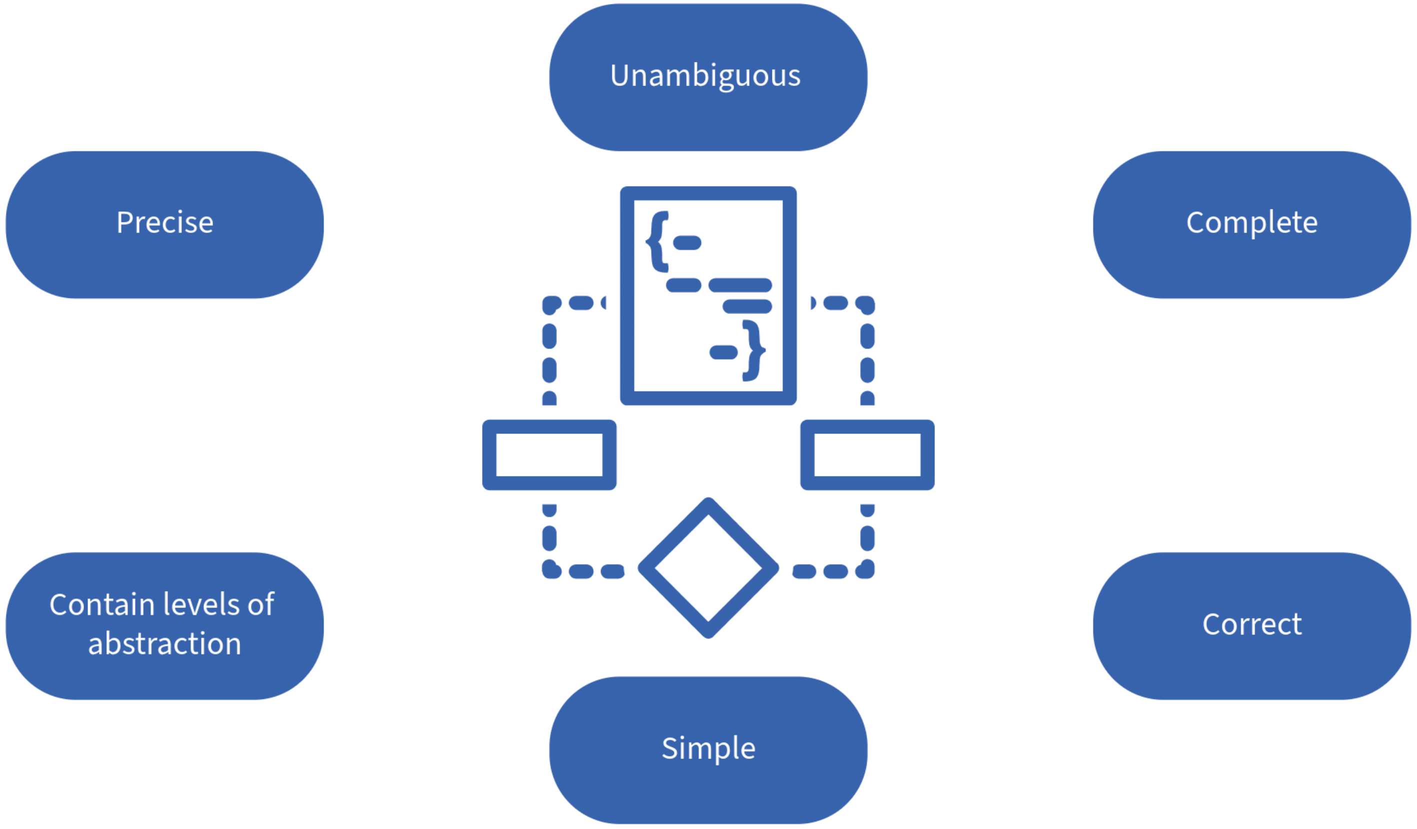
1. Wet Hair
2. Apply a small amount of shampoo
3. Lather
4. Rinse
5. If hair is not clean, go back to step 2

Submit your responses to the chat!

Properties of good algorithms

Good algorithms comprise the following properties:

- Natural language (English)
- Pictures
- Pseudo-code or a specific programming language

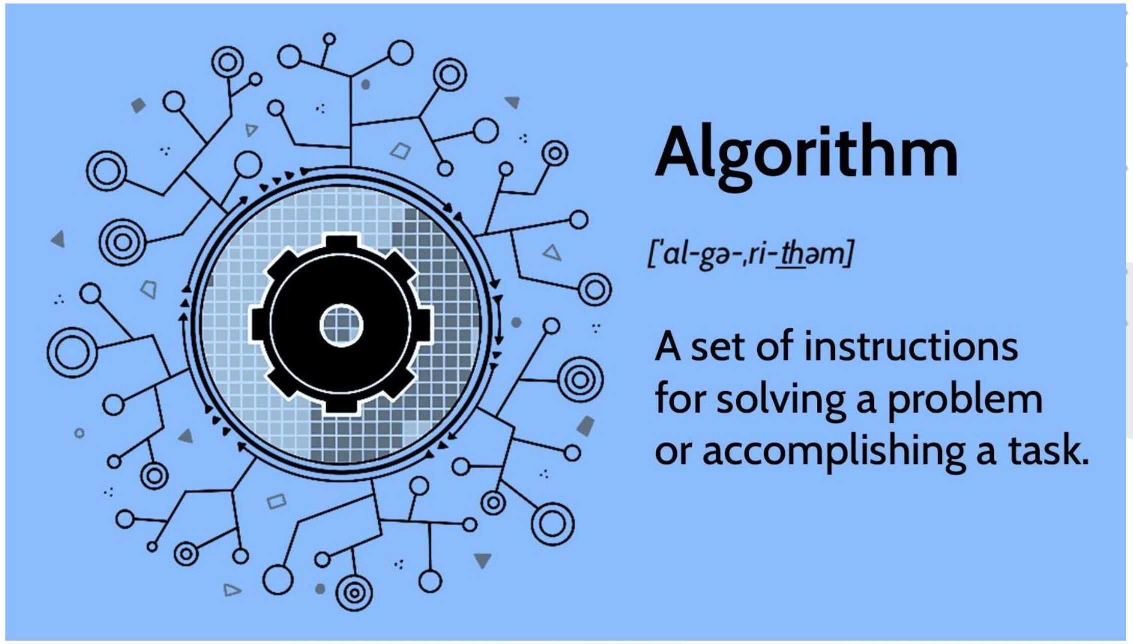


The properties of algorithms

The components of algorithms

Any computing algorithm will have AT MOST five kinds of components:

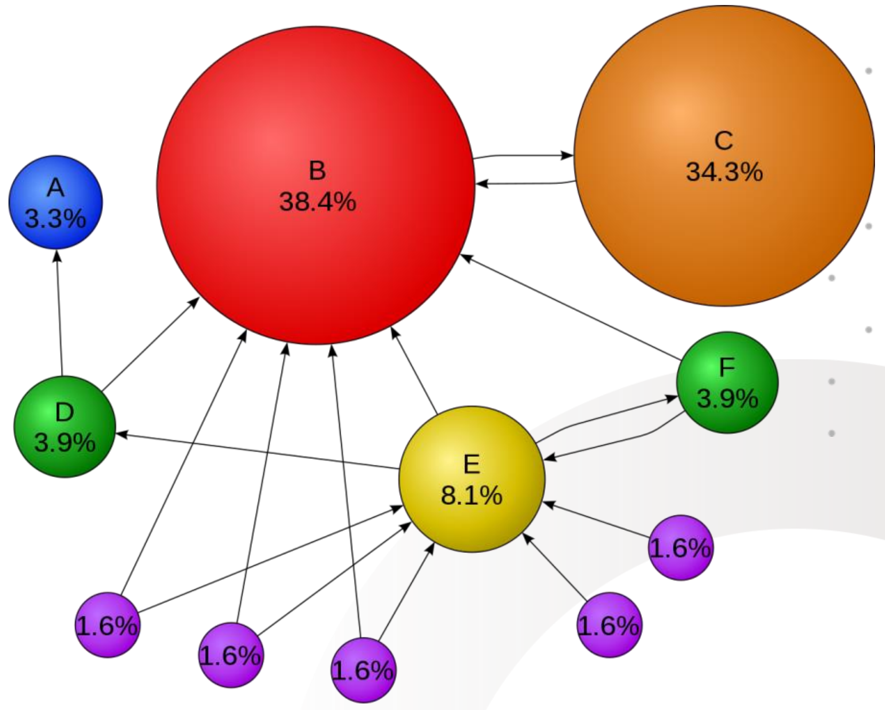
1. **Data structures –** to hold data
2. **Instructions –** to change data values
3. **Conditional expressions –** to make decisions
4. **Control structures –** to act on decisions
5. **Modules –** to make the algorithm manageable by abstraction; i.e – grouping related components


Algorithm
['al-gə-ˌri-t͟həm]
A set of instructions for solving a problem or accomplishing a task.

Algorithms in real-life

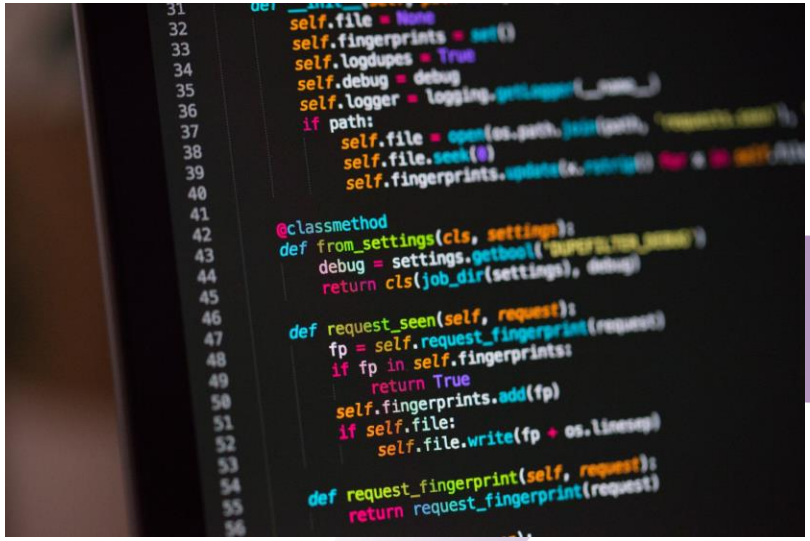
Adaptive learning technology

- Google's search results are determined, in part, by the **PageRank algorithm**
- The PageRank algorithm assigns a webpage's importance based on the **number of sites linking to it**
- When we google a phrase like "What are examples of algorithmic thinking?", the chosen pages likely have the **most links** to them for that specific topic



A simple illustration of the Pagerank algorithm - *The percentage shows the perceived importance, and the arrows represent hyperlinks.*

Turning abstractions into code

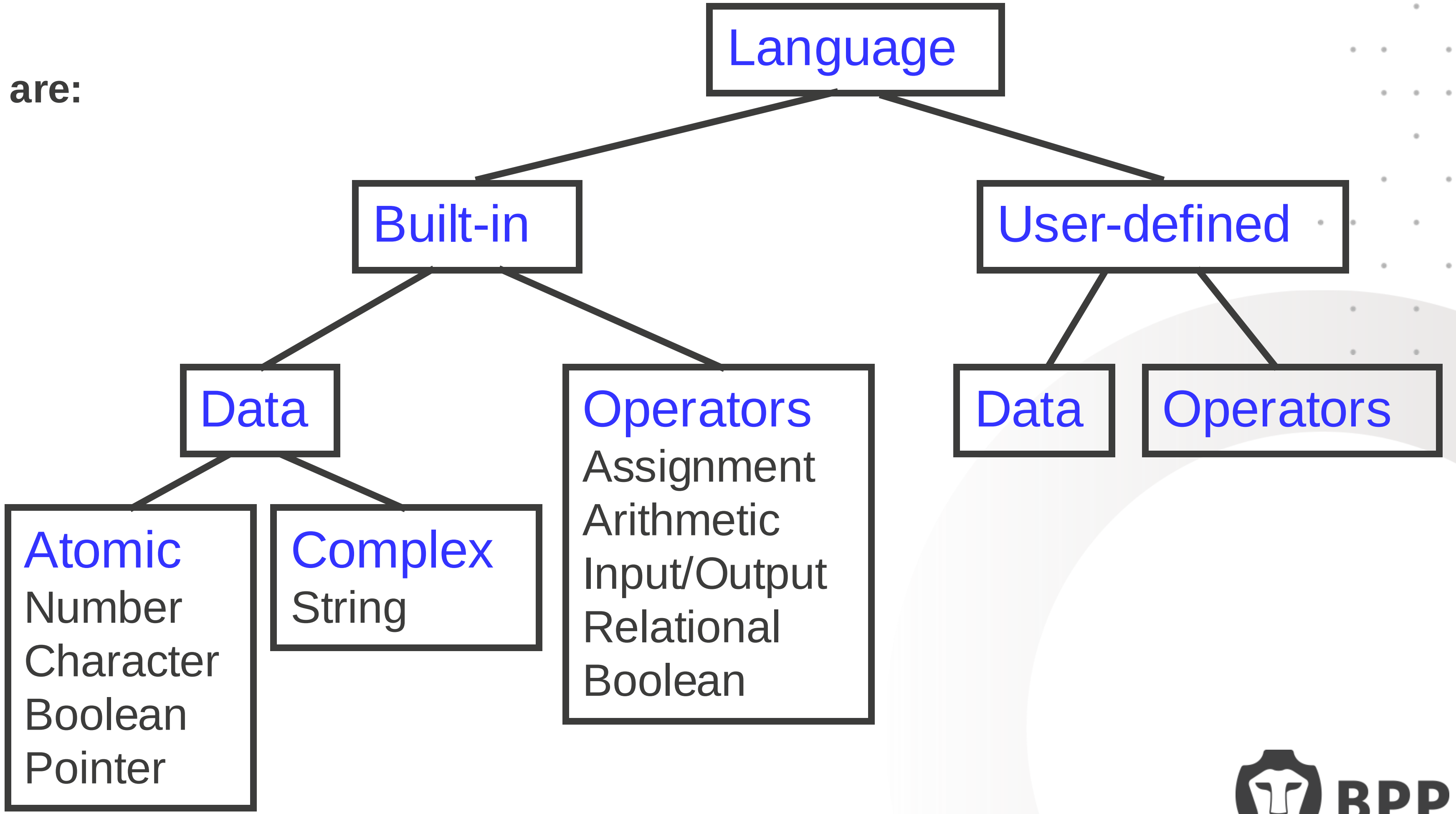


```
    def __init__(self, path=None, debug=False):
        self.file = None
        self.fingerprints = set()
        self.logdupes = True
        self.debug = debug
        self.logger = logging.getLogger(__name__)
        if path:
            self.file = open(os.path.join(path, 'requests.seen'), 'a+')
            self.file.seek(0)
            self.fingerprints.update(x.rstrip() for x in self.file)

    @classmethod
    def from_settings(cls, settings):
        debug = settings.getbool('DUPEFILTER_DEBUG')
        return cls(job_dir(settings), debug)

    def request_seen(self, request):
        fp = self.request_fingerprint(request)
        if fp in self.fingerprints:
            return True
        self.fingerprints.add(fp)
        if self.file:
            self.file.write(fp + os.linesep)

    def request_fingerprint(self, request):
        return request_fingerprint(request)
```

What are the elements of a programming language?

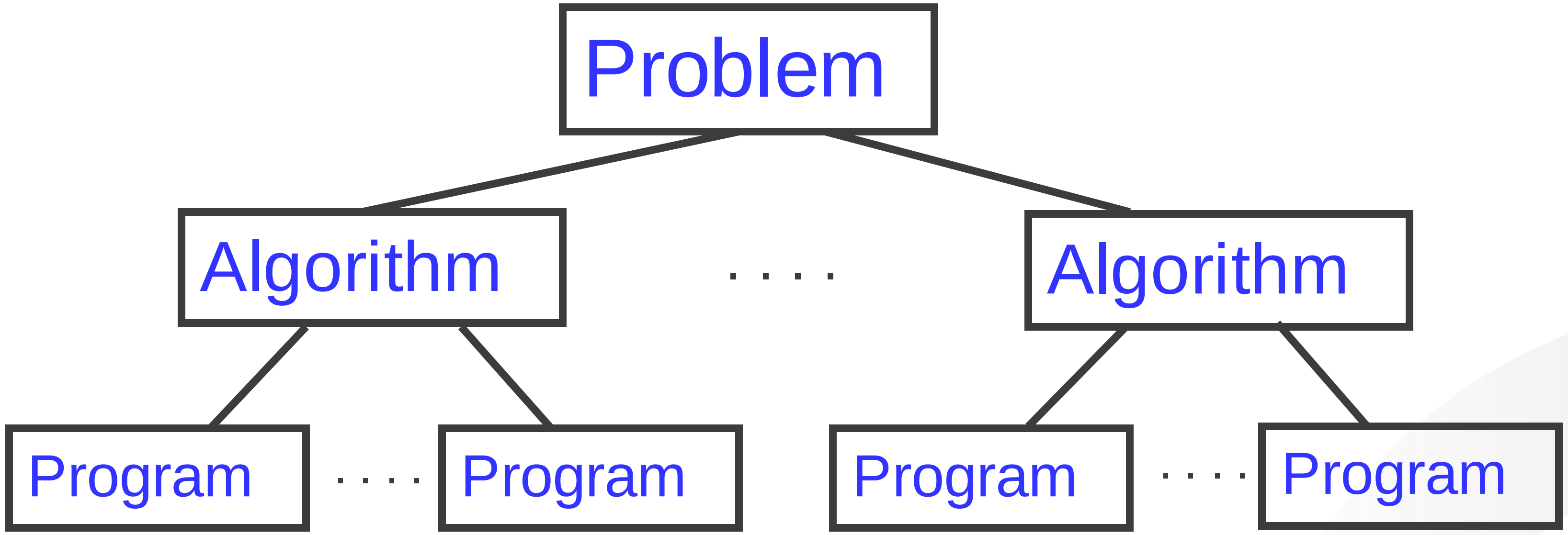
Consists of the elements that are:

- Built-in
- User-defined



Expressing a problem as programs

The relations between problems, algorithms and programs



The relation between problems, algorithms and programs

Input and output

I/O Operators allow us to communicate with the “outside world,” the world beyond the algorithm itself.

We’re not concerned with formatting.

Print (pseudocode)

Displays output items to the user

Syntax:

```
print(item_1, item_2, ..., item_n)
```

Examples:

```
print(“Please enter your info.”)
print(num_one, my_char, is_Student)
```

Read (pseudocode)

Obtains input items from the user

Syntax:

```
read(item_1, item_2, ..., item_n)
```

Examples:

```
read(menu_choice)
read(is_Student, name, age)
```

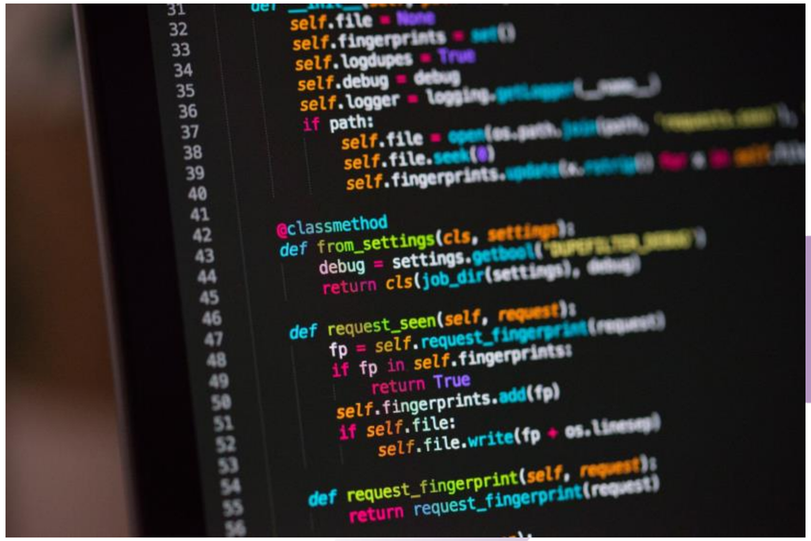
No automatic prompting!

Input and output

Examples

```
algorithm IO_Example
  num_one, num_two, average isoftype Num

// obtain two numbers
   print("Please enter two numbers")
   read (num_one, num_two)

// output a literal text message and the
// value of the sum
   print ("Sum = ", num_one + num_two)

// output a string literal and average
   average <- (num_one + num_two) / 2
   print ("Average = ", average)
endalgorithm
```

Computational performance



```
    self.file = None
    self.fingerprints = set()
    self.logdupes = True
    self.debug = debug
    self.logger = logging.getLogger(__name__)
    if path:
        self.file = open(os.path.join(path, 'requests.seen'), 'a+')
        self.file.seek(0)
        self.fingerprints.update(x.rstrip() for x in self.file)

@classmethod
def from_settings(cls, settings):
    debug = settings.getbool('DUPEFILTER_DEBUG')
    return cls(job_dir(settings), debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

def request_fingerprint(self, request):
    return request_fingerprint(request)
```

Introduction

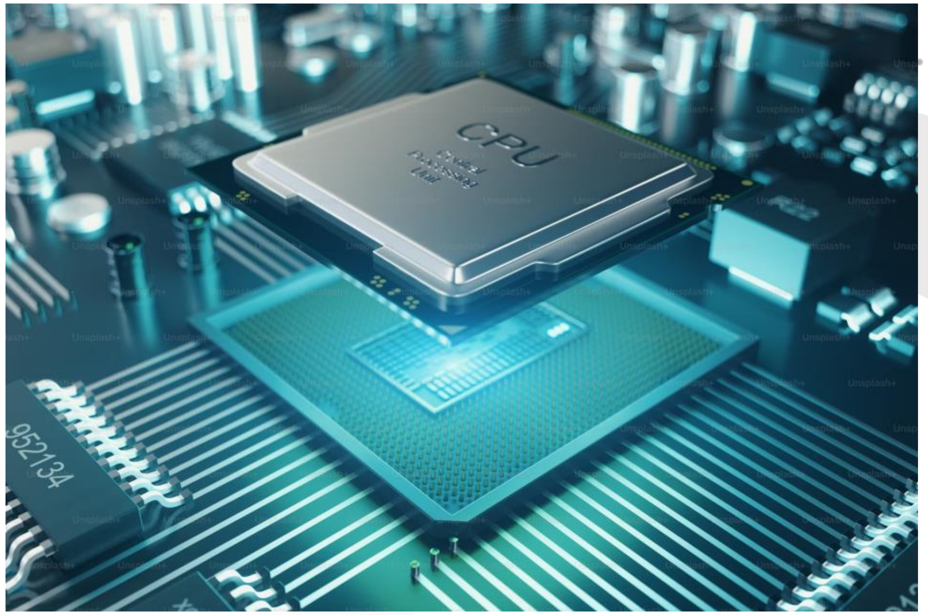
Building Careers Through Education

Everybody Matters. Stronger Together. Student, Learner and Client Centric. Trust and Respect. Embrace Change.

Do we need to care about performance when computers keep getting faster?

- Our history is to do bigger problems, not the same ones faster

- More complex problems are less tied to our everyday “common sense” experience



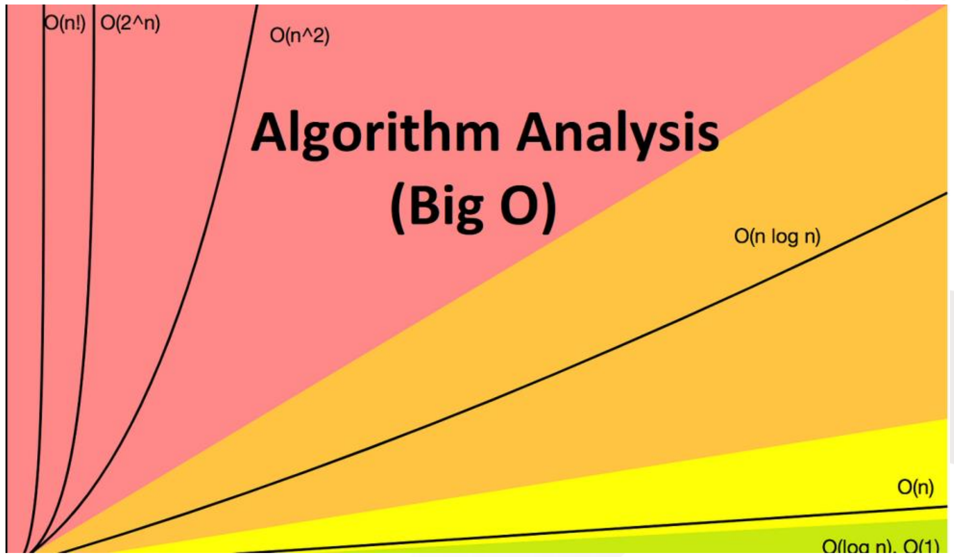
BPP

Algorithm analysis

How should you approach this?

- ➢ **We could compare two programs by running them side-by-side**
  - But that means we have to implement them!
- ➢ **We want a way to easily evaluate programs before they are written**
  - Look at the algorithm, not the program
- ➢ **Algorithm Analysis estimates problem cost as a function of growth rate**

It is the input that we want to grow, and then observe how the algorithm “handles” this scaling



More on this diagram in a minute!

Sorting: Insertion sort

How should you approach this?

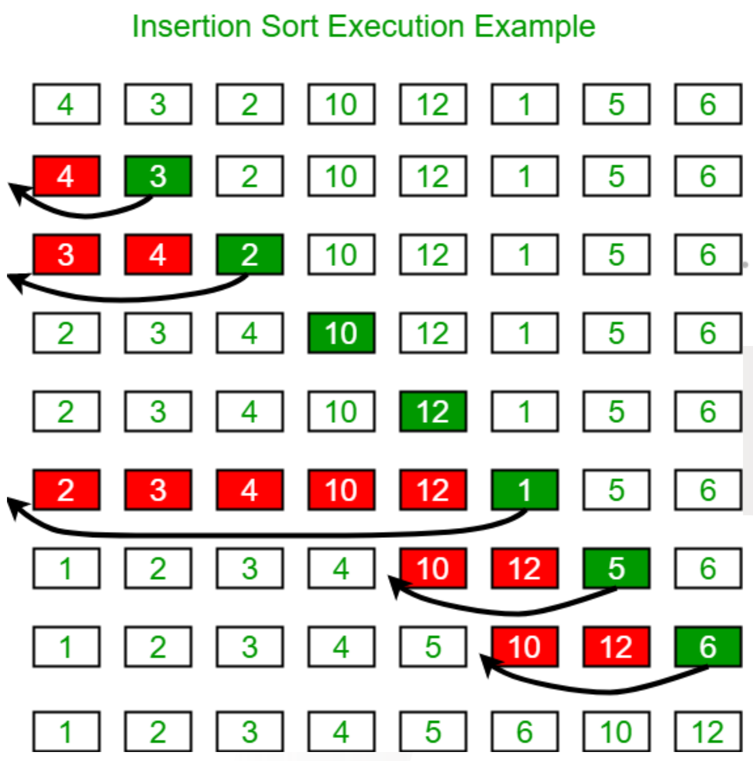
For each record:

- Insert it into the sorted list made from the records already seen
- Go from left to right, and move elements to the left if needed

➢ Might have to look at each such record already in the (sorted) list – n work

➢ Since we do this for n records, this is n*n in the worst (and average) cases

➢ So the cost is proportional to n2 (unless we are really lucky)



*Insertion sort execution example, **image source:** GeekforGeeks*

Sorting: Merge sort

How should you approach this?

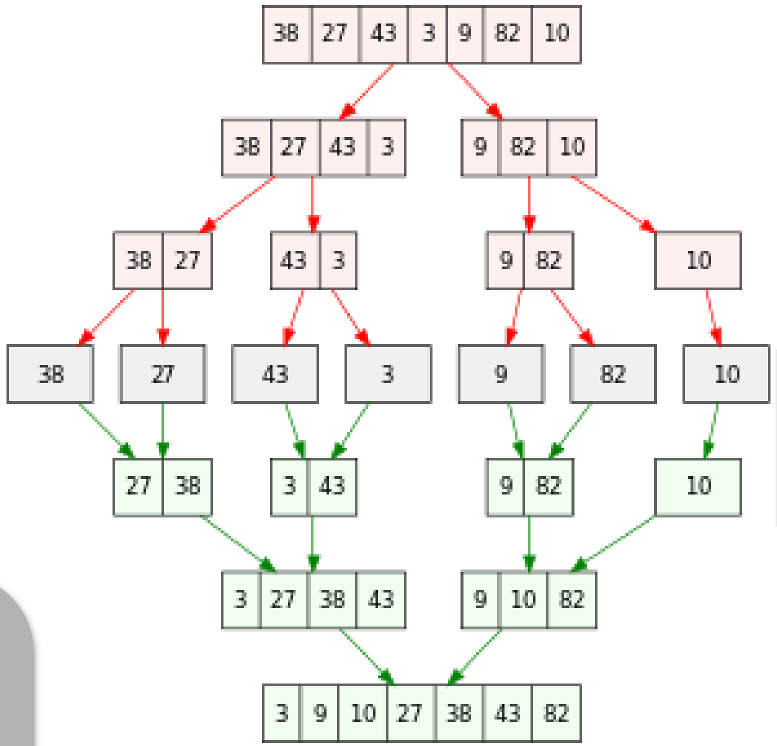
For a list of n records:

- Split the list in half. Keep splitting.
- When list has two elements, swap the elements around if necessary
- Merge the records together (needs n work)
- **Total cost:** proportional to n log n

Does it matter?

- 1000 records:
  - Insertion sort: 1,000,000
  - Mergesort: 10,000
  - Factor of 100 difference

- 1,000,000 records
- Insertion sort: 1,000,000,000,000
- Mergsort: 20,000,000
- Factor of 50,000 difference
- Hours vs. seconds on a real computer



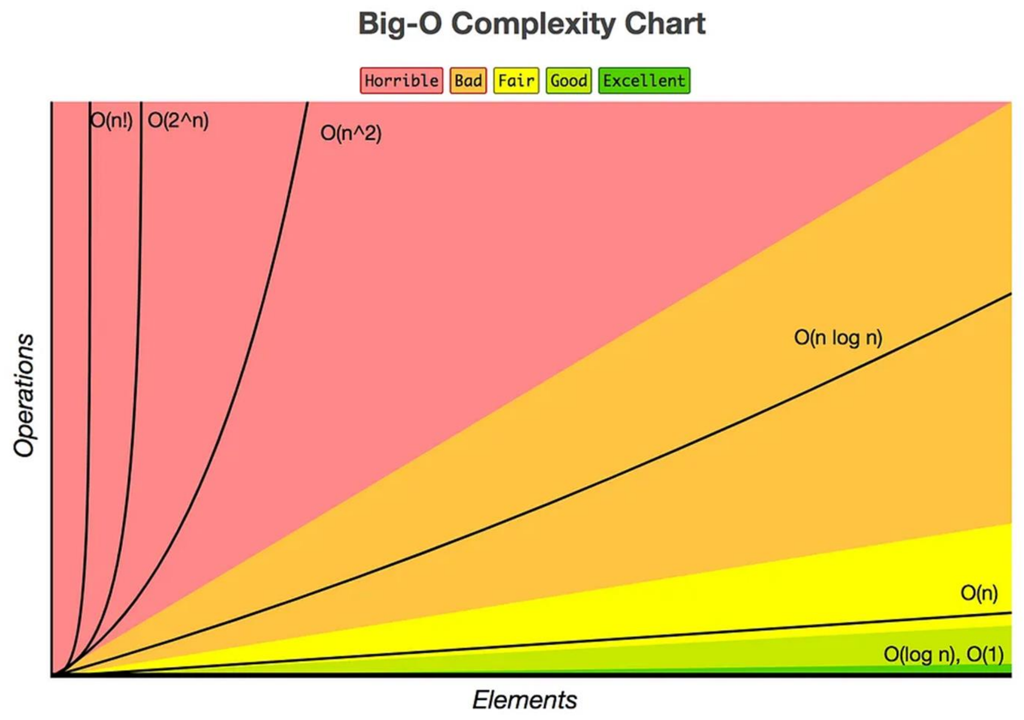
Merge sort execution example, ***image source:*** *Wikipedia*

Algorithm analysis

Based on cost and complexity

Algorithm complexity can be expressed in Order notation, e.g. “at what rate does work grow with N?”:

- O(1) Constant
- O(logN) Sub-linear
- O(N) Linear
- O(NlogN) Nearly linear
- O(N2)Quadratic
- O(XN) Exponential



The Big-O complexity chart

But, for a given problem, how do we know if a better algorithm is possible?

Practical application

Building Careers Through Education

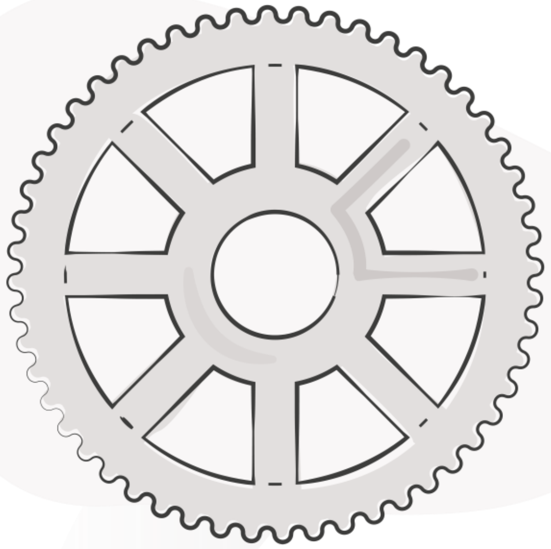
Everybody Matters. Stronger Together. Student, Learner and Client Centric. Trust and Respect. Embrace Change.

Notebook activity

Your instructor will now walk you through the associated Python notebook for this topic.

This file can be found at the following link:

Python notebook link



Practical application

BPP

Thank you

Do you have any questions, comments, or feedback?

How confident do you now feel about your knowledge of algorithmic thinking following this webinar?

- **A:** Very confident
- **B:** More confident than before this webinar
- **C:** What was this webinar session even about?!

Submit your responses to the chat!