

# Introduction

## COMS10018 - Algorithms

Dr Christian Konrad

Algorithms?

# Algorithms?

# Algorithms?

## Algorithms?

A procedure that solves a *computational problem*

# Algorithms?

## Algorithms?

A procedure that solves a *computational problem*

## Computational Problem?

# Algorithms?

## Algorithms?

A procedure that solves a *computational problem*

## Computational Problem?

- How often does “Juliet” appear in Shakespeare’s “Romeo And Juliet” ?

# Algorithms?

## Algorithms?

A procedure that solves a *computational problem*

## Computational Problem?

- How often does “Juliet” appear in Shakespeare’s “Romeo And Juliet” ? (181 times) ([text/strings](#))

# Algorithms?

## Algorithms?

A procedure that solves a *computational problem*

## Computational Problem?

- How often does “Juliet” appear in Shakespeare’s “Romeo And Juliet”? (181 times) ([text/strings](#))
- Sort an array of  $n$  numbers ([all areas](#))

# Algorithms?

## Algorithms?

A procedure that solves a *computational problem*

## Computational Problem?

- How often does “Juliet” appear in Shakespeare’s “Romeo And Juliet”? (181 times) ([text/strings](#))
- Sort an array of  $n$  numbers ([all areas](#))
- How do we factorize a large number? ([crypto](#))

# Algorithms?

## Algorithms?

A procedure that solves a *computational problem*

## Computational Problem?

- How often does “Juliet” appear in Shakespeare’s “Romeo And Juliet”? (181 times) ([text/strings](#))
- Sort an array of  $n$  numbers ([all areas](#))
- How do we factorize a large number? ([crypto](#))
- Shortest way to travel from Bristol to Glasgow? ([graph algorithms](#))

# Algorithms?

## Algorithms?

A procedure that solves a *computational problem*

## Computational Problem?

- How often does “Juliet” appear in Shakespeare’s “Romeo And Juliet”? (181 times) ([text/strings](#))
- Sort an array of  $n$  numbers ([all areas](#))
- How do we factorize a large number? ([crypto](#))
- Shortest way to travel from Bristol to Glasgow? ([graph algorithms](#))
- How to execute a database query? ([databases](#))

# Algorithms?

## Algorithms?

A procedure that solves a *computational problem*

## Computational Problem?

- How often does “Juliet” appear in Shakespeare’s “Romeo And Juliet”? (181 times) ([text/strings](#))
- Sort an array of  $n$  numbers ([all areas](#))
- How do we factorize a large number? ([crypto](#))
- Shortest way to travel from Bristol to Glasgow? ([graph algorithms](#))
- How to execute a database query? ([databases](#))
- Is it possible to partition the set  $\{17, 8, 4, 22, 9, 28, 2\}$  into two sets s.t. their sums are equal? ([scheduling, load balancing](#))

# Algorithms?

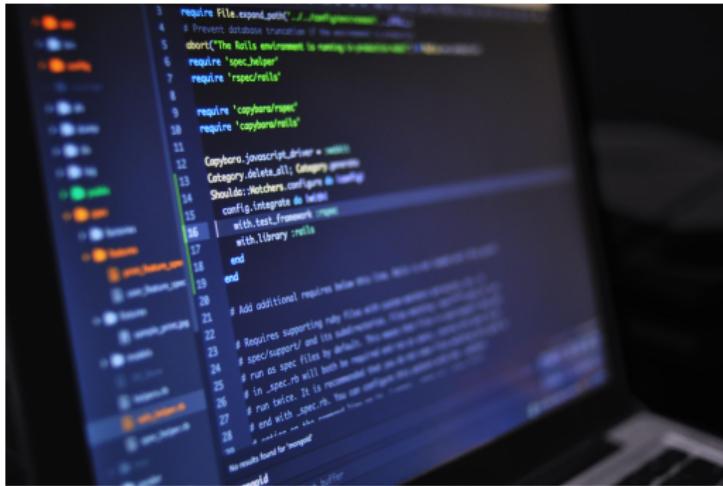
## Algorithms?

A procedure that solves a *computational problem*

## Computational Problem?

- How often does “Juliet” appear in Shakespeare’s “Romeo And Juliet”? (181 times) ([text/strings](#))
- Sort an array of  $n$  numbers ([all areas](#))
- How do we factorize a large number? ([crypto](#))
- Shortest way to travel from Bristol to Glasgow? ([graph algorithms](#))
- How to execute a database query? ([databases](#))
- Is it possible to partition the set  $\{17, 8, 4, 22, 9, 28, 2\}$  into two sets s.t. their sums are equal? ([scheduling, load balancing](#))  
 $\{8, 9, 28\}, \{2, 4, 17, 22\}$

# Brain Behind Your Software



## Algorithms:

- Fabric that Software is made of
- Inner logic of your Software

# Brain Behind Your Software



## Algorithms:

- Fabric that Software is made of
- Inner logic of your Software
- Insufficient computational power → Improve your algorithms!

## Efficiency



## Efficiency

- The faster the better: **Time complexity**



## Efficiency

- The faster the better: **Time complexity**
- Use as little memory as possible: **Space complexity**



## Efficiency

- The faster the better: **Time complexity**
- Use as little memory as possible: **Space complexity**



## Mathematics

## Efficiency

- The faster the better: **Time complexity**
- Use as little memory as possible: **Space complexity**



## Mathematics

- We will prove that algorithms run fast and use little memory

## Efficiency

- The faster the better: **Time complexity**
- Use as little memory as possible: **Space complexity**



## Mathematics

- We will prove that algorithms run fast and use little memory
- We will prove that algorithms are correct

## Efficiency

- The faster the better: **Time complexity**
- Use as little memory as possible: **Space complexity**



## Mathematics

- We will prove that algorithms run fast and use little memory
- We will prove that algorithms are correct
- **Tools:** Induction, algebra, sums, . . . , rigorous arguments

## Efficiency

- The faster the better: **Time complexity**
- Use as little memory as possible: **Space complexity**



## Mathematics

- We will prove that algorithms run fast and use little memory
- We will prove that algorithms are correct
- **Tools:** Induction, algebra, sums, . . . , rigorous arguments

## Theoretical Computer Science

## Efficiency

- The faster the better: **Time complexity**
- Use as little memory as possible: **Space complexity**



## Mathematics

- We will prove that algorithms run fast and use little memory
- We will prove that algorithms are correct
- **Tools:** Induction, algebra, sums, . . . , rigorous arguments

## Theoretical Computer Science

No implementations in this unit!

# What you get out of this unit

## Algorithm 1 Single-pass Semi-Streaming Algorithm for MDS

---

**Require:** Bipartite input graph  $G = (A, B, E)$  with  $|A| = |B| = n$

- 1: Let  $D_1, D_2, \dots, D_{\log n} \leftarrow \{\}$
- 2: For every  $a \in A$ :  $d(a) \leftarrow 0$
- 3:  $U \leftarrow \emptyset$  {Keep track of dominated nodes ( $U \subseteq B$  always holds)}
- 4: For every  $b \in B$ :  $C(b) \leftarrow \perp$  {Output cover certificate}

## Goals:

# What you get out of this unit

## Algorithm 1 Single-pass Semi-Streaming Algorithm for MDS

---

**Require:** Bipartite input graph  $G = (A, B, E)$  with  $|A| = |B| = n$

- 1: Let  $D_1, D_2, \dots, D_{\log n} \leftarrow \{\}$
- 2: For every  $a \in A$ :  $d(a) \leftarrow 0$
- 3:  $U \leftarrow \emptyset$  {Keep track of dominated nodes ( $U \subseteq B$  always holds)}
- 4: For every  $b \in B$ :  $C(b) \leftarrow \perp$  {Output cover certificate}

**Goals:** First steps towards becoming an algorithms designer

# What you get out of this unit

## Algorithm 1 Single-pass Semi-Streaming Algorithm for MDS

---

**Require:** Bipartite input graph  $G = (A, B, E)$  with  $|A| = |B| = n$

- 1: Let  $D_1, D_2, \dots, D_{\log n} \leftarrow \{\}$
- 2: For every  $a \in A$ :  $d(a) \leftarrow 0$
- 3:  $U \leftarrow \emptyset$  {Keep track of dominated nodes ( $U \subseteq B$  always holds)}
- 4: For every  $b \in B$ :  $C(b) \leftarrow \perp$  {Output cover certificate}

**Goals:** First steps towards becoming an algorithms designer

- 1 Learn techniques that help you design & analyze algorithms

# What you get out of this unit

## Algorithm 1 Single-pass Semi-Streaming Algorithm for MDS

---

**Require:** Bipartite input graph  $G = (A, B, E)$  with  $|A| = |B| = n$

- 1: Let  $D_1, D_2, \dots, D_{\log n} \leftarrow \{\}$
- 2: For every  $a \in A$ :  $d(a) \leftarrow 0$
- 3:  $U \leftarrow \emptyset$  {Keep track of dominated nodes ( $U \subseteq B$  always holds)}
- 4: For every  $b \in B$ :  $C(b) \leftarrow \perp$  {Output cover certificate}

**Goals:** First steps towards becoming an algorithms designer

- 1 Learn techniques that help you design & analyze algorithms
- 2 Understand a set of well-known algorithms

# What you get out of this unit

## Algorithm 1 Single-pass Semi-Streaming Algorithm for MDS

**Require:** Bipartite input graph  $G = (A, B, E)$  with  $|A| = |B| = n$

- 1: Let  $D_1, D_2, \dots, D_{\log n} \leftarrow \{\}$
- 2: For every  $a \in A$ :  $d(a) \leftarrow 0$
- 3:  $U \leftarrow \emptyset$  {Keep track of dominated nodes ( $U \subseteq B$  always holds)}
- 4: For every  $b \in B$ :  $C(b) \leftarrow \perp$  {Output cover certificate}

**Goals:** First steps towards becoming an algorithms designer

- 1 Learn techniques that help you design & analyze algorithms
- 2 Understand a set of well-known algorithms

**Systematic Approach to Problem/Puzzle Solving**

# What you get out of this unit

## Algorithm 1 Single-pass Semi-Streaming Algorithm for MDS

---

**Require:** Bipartite input graph  $G = (A, B, E)$  with  $|A| = |B| = n$

- 1: Let  $D_1, D_2, \dots, D_{\log n} \leftarrow \{\}$
- 2: For every  $a \in A$ :  $d(a) \leftarrow 0$
- 3:  $U \leftarrow \emptyset$  {Keep track of dominated nodes ( $U \subseteq B$  always holds)}
- 4: For every  $b \in B$ :  $C(b) \leftarrow \perp$  {Output cover certificate}

**Goals:** First steps towards becoming an algorithms designer

- 1 Learn techniques that help you design & analyze algorithms
- 2 Understand a set of well-known algorithms

## Systematic Approach to Problem/Puzzle Solving

- Study a problem, discover structure, exploit structure and design algorithms

# What you get out of this unit

## Algorithm 1 Single-pass Semi-Streaming Algorithm for MDS

---

**Require:** Bipartite input graph  $G = (A, B, E)$  with  $|A| = |B| = n$

- 1: Let  $D_1, D_2, \dots, D_{\log n} \leftarrow \{\}$
- 2: For every  $a \in A$ :  $d(a) \leftarrow 0$
- 3:  $U \leftarrow \emptyset$  {Keep track of dominated nodes ( $U \subseteq B$  always holds)}
- 4: For every  $b \in B$ :  $C(b) \leftarrow \perp$  {Output cover certificate}

**Goals:** First steps towards becoming an algorithms designer

- 1 Learn techniques that help you design & analyze algorithms
- 2 Understand a set of well-known algorithms

## Systematic Approach to Problem/Puzzle Solving

- Study a problem, discover structure, exploit structure and design algorithms
- Useful in all areas of Computer Science

# What you get out of this unit

## Algorithm 1 Single-pass Semi-Streaming Algorithm for MDS

---

**Require:** Bipartite input graph  $G = (A, B, E)$  with  $|A| = |B| = n$

- 1: Let  $D_1, D_2, \dots, D_{\log n} \leftarrow \{\}$
- 2: For every  $a \in A$ :  $d(a) \leftarrow 0$
- 3:  $U \leftarrow \emptyset$  {Keep track of dominated nodes ( $U \subseteq B$  always holds)}
- 4: For every  $b \in B$ :  $C(b) \leftarrow \perp$  {Output cover certificate}

**Goals:** First steps towards becoming an algorithms designer

- 1 Learn techniques that help you design & analyze algorithms
- 2 Understand a set of well-known algorithms

## Systematic Approach to Problem/Puzzle Solving

- Study a problem, discover structure, exploit structure and design algorithms
- Useful in all areas of Computer Science
- **Interview Questions:** Google, Facebook, Amazon, etc.

# My Goals

## My Goals

# My Goals

## My Goals

- Get you excited about Algorithms

# My Goals

## My Goals

- Get you excited about Algorithms
- Make you contribute to Algorithms research in Bristol

# My Goals

## My Goals

- Get you excited about Algorithms
- Make you contribute to Algorithms research in Bristol

## Algorithms in Bristol

- 1st year: OOP + **Algo** (Algorithms 1)

# My Goals

## My Goals

- Get you excited about Algorithms
- Make you contribute to Algorithms research in Bristol

## Algorithms in Bristol

- 1st year: OOP + **Algo** (Algorithms 1)
- 2nd year: **Algorithms** and Data (Algorithms 2)

# My Goals

## My Goals

- Get you excited about Algorithms
- Make you contribute to Algorithms research in Bristol

## Algorithms in Bristol

- 1st year: OOP + **Algo** (Algorithms 1)
- 2nd year: **Algorithms** and Data (Algorithms 2)
- 3rd year: Advanced Algorithms (Algorithms 3)

# My Goals

## My Goals

- Get you excited about Algorithms
- Make you contribute to Algorithms research in Bristol

## Algorithms in Bristol

- 1st year: OOP + **Algo** (Algorithms 1)
- 2nd year: **Algorithms** and Data (Algorithms 2)
- 3rd year: Advanced Algorithms (Algorithms 3)
- 4th year: Advanced Topics in Theoretical Computer Science (Algorithms 4)

# My Goals

## My Goals

- Get you excited about Algorithms
- Make you contribute to Algorithms research in Bristol

## Algorithms in Bristol

- 1st year: OOP + **Algo** (Algorithms 1)
- 2nd year: **Algorithms** and Data (Algorithms 2)
- 3rd year: Advanced Algorithms (Algorithms 3)
- 4th year: Advanced Topics in Theoretical Computer Science (Algorithms 4)

**BSc/MEng Projects, Reading Group, Summer Internships,  
PhD students**

## Teaching Sessions

- **Lectures:** Mondays 10am, Fridays 2pm
- **Problem sheet sessions:** Throughout the week

Week  $X + 1$  covers sheet released in Week  $X$

- **OPTIONAL Drop-in/discussion session:** (Fridays 1pm-2pm) ask questions about the material or other algorithms-related topics
- **OPTIONAL Office hours:** (Monday 11-noon) Ask me anything about the unit

## Teaching Sessions

- **Lectures:** Mondays 10am, Fridays 2pm
- **Problem sheet sessions:** Throughout the week

Week  $X + 1$  covers sheet released in Week  $X$

- **OPTIONAL Drop-in/discussion session:** (Fridays 1pm-2pm) ask questions about the material or other algorithms-related topics
- **OPTIONAL Office hours:** (Monday 11-noon) Ask me anything about the unit

## Assessment

- **Exam:** Counts 50% towards your final mark in the joint unit “Object-Oriented Programming and Algorithms”
- **Attention:** Algorithms exam is a must-pass component

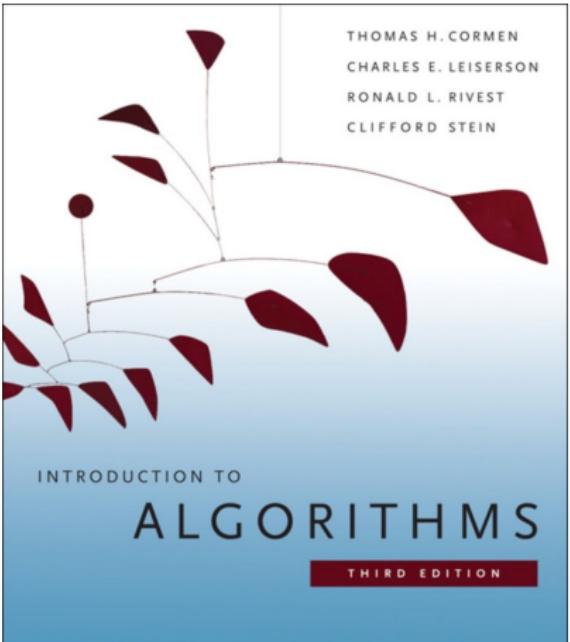
# Teaching Staff

## Teaching Staff

- **Unit Director:** Dr Christian Konrad  
(christian.konrad@bristol.ac.uk)
- **Lead TA:** Conor O'Sullivan  
(c.osullivan@bristol.ac.uk)
- **TAs:** Artem Borisov, Michael Degamo, Amos Holland, Harley Huang, Sophia Izwan, Thomas Parr, Vlad Petre, Kasim Rafiq, Ava Raisian, Archie Walton, Stephen Yang

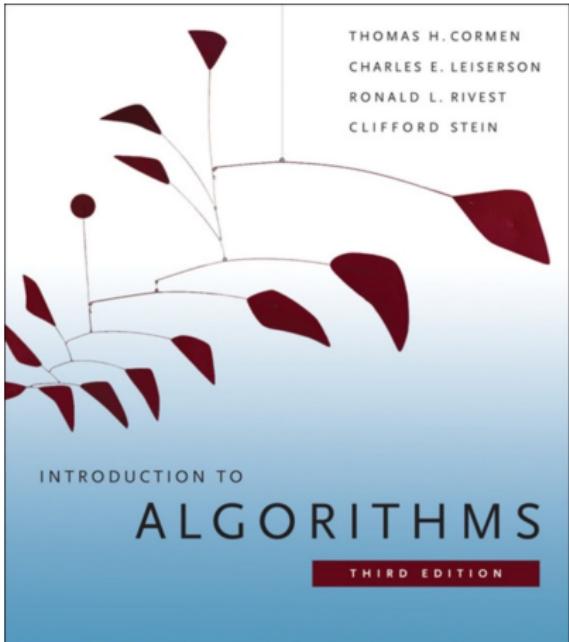


# Further Reading



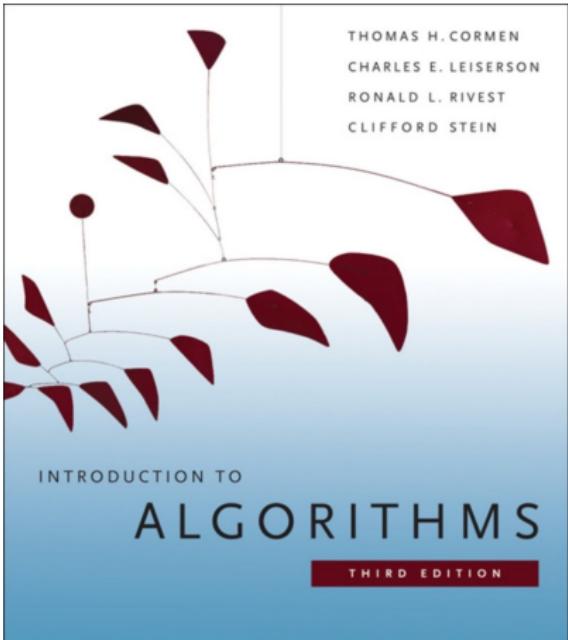
# Further Reading

- More details on many of the topics discussed in this unit



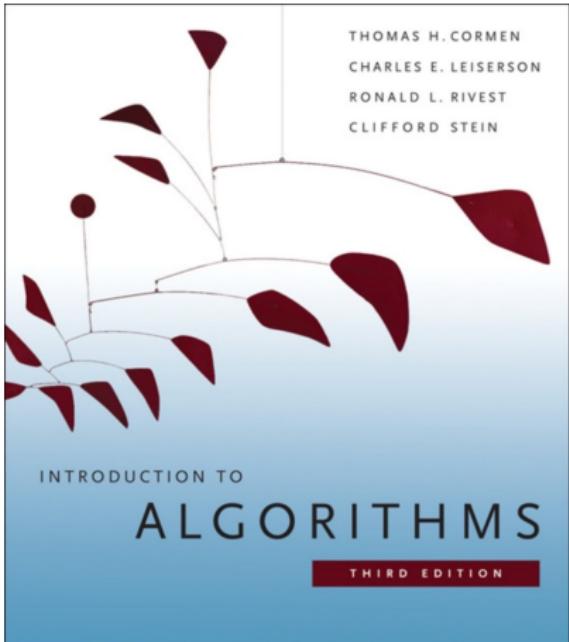
# Further Reading

- More details on many of the topics discussed in this unit
- However, not all topics can be found in this book



# Further Reading

- More details on many of the topics discussed in this unit
- However, not all topics can be found in this book
- Unit materials cover everything you need to know



# How to Succeed in this Unit

## **How to succeed**

# How to Succeed in this Unit

## How to succeed

- Make sure you understand the material

# How to Succeed in this Unit

## How to succeed

- Make sure you understand the material
- **Work on provided exercises!**

# How to Succeed in this Unit

## How to succeed

- Make sure you understand the material
- **Work on provided exercises!**
- Use MS Teams channel for discussions and questions

# How to Succeed in this Unit

## How to succeed

- Make sure you understand the material
- **Work on provided exercises!**
- Use MS Teams channel for discussions and questions
- **Work on provided exercises!!**

# How to Succeed in this Unit

## How to succeed

- Make sure you understand the material
- **Work on provided exercises!**
- Use MS Teams channel for discussions and questions
- **Work on provided exercises!!**
- **Work on provided exercises!!!**

# How to Succeed in this Unit

## How to succeed

- Make sure you understand the material
- **Work on provided exercises!**
- Use MS Teams channel for discussions and questions
- **Work on provided exercises!!**
- **Work on provided exercises!!!**

**Unit webpage:** Use link on blackboard

[https://bristolalgo.github.io/courses/2025\\_2026\\_COMS10018/coms10018.html](https://bristolalgo.github.io/courses/2025_2026_COMS10018/coms10018.html)

# What's next

## This Week

- Monday 10am-11am: [Lecture](#) (today)
- Friday 2pm-3pm: [Lecture](#)

## Next Week and every subsequent week

- Monday 10am-11am: [Lecture](#)
- OPTIONAL Monday 11-noon: [Office Hours](#)
- Monday, Tuesday, Thursday: [Small-group problem sheet sessions](#)
- Friday 2pm-3pm: [Lecture](#)
- OPTIONAL Friday 1pm-2pm: [Drop-in Session](#)



**Good luck and enjoy! Questions?**