

Introduction

COMS10017 - (Object-Oriented Programming and) Algorithms

Dr Christian Konrad

Algorithms?

Algorithms?

Algorithms?

A procedure that solves a *computational problem*

Algorithms?

A procedure that solves a *computational problem*

Computational Problem?

Algorithms?

A procedure that solves a *computational problem*

Computational Problem?

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

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?

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?

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?

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?

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?

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?

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
- 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 $b \in B$: $C(b) \leftarrow \emptyset$ {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 $b \in B$: $C(b) \leftarrow \emptyset$ {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 $b \in B$: $C(b) \leftarrow \emptyset$ {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 $b \in B$: $C(b) \leftarrow \emptyset$ {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 $b \in B$: $C(b) \leftarrow \emptyset$ {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 $b \in B$: $C(b) \leftarrow \emptyset$ {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 within it, 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 $b \in B$: $C(b) \leftarrow \emptyset$ {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 within it, 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 within it, exploit structure and design algorithms
- Useful in all areas of Computer Science
- **Interview Questions:** Google, Facebook, Amazon, etc.

My Goals

My Goals

- Get you excited about Algorithms

My Goals

- Get you excited about Algorithms
- Shape new generation of Algorithm Designers at Bristol

My Goals

- Get you excited about Algorithms
- Shape new generation of Algorithm Designers at Bristol

Algorithms in Bristol

- 1st year: Algorithms (Algorithms 1)

My Goals

- Get you excited about Algorithms
- Shape new generation of Algorithm Designers at Bristol

Algorithms in Bristol

- 1st year: Algorithms (Algorithms 1)
- 2nd year: Algorithms 2

My Goals

- Get you excited about Algorithms
- Shape new generation of Algorithm Designers at Bristol

Algorithms in Bristol

- 1st year: Algorithms (Algorithms 1)
- 2nd year: Algorithms 2
- 3rd year: Advanced Algorithms (Algorithms 3)

My Goals

- Get you excited about Algorithms
- Shape new generation of Algorithm Designers at Bristol

Algorithms in Bristol

- 1st year: Algorithms (Algorithms 1)
- 2nd year: Algorithms 2
- 3rd year: Advanced Algorithms (Algorithms 3)
- 4th year: Advanced Topics in Theoretical Computer Science (Algorithms 4)

My Goals

- Get you excited about Algorithms
- Shape new generation of Algorithm Designers at Bristol

Algorithms in Bristol

- 1st year: Algorithms (Algorithms 1)
- 2nd year: 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, Thursdays 3pm
- **Problem sheet sessions:** (Tuesdays) TA-led problem sheet sessions, come prepared!
- **OPTIONAL Drop-in/discussion session:** (Thursdays 1pm-2pm) ask questions about the material or other algorithms-related topics
- **OPTIONAL Office hours:** (Fridays 1pm-2pm) Ask me anything about the unit

Teaching Sessions

- **Lectures:** Mondays 10am, Thursdays 3pm
- **Problem sheet sessions:** (Tuesdays) TA-led problem sheet sessions, come prepared!
- **OPTIONAL Drop-in/discussion session:** (Thursdays 1pm-2pm) ask questions about the material or other algorithms-related topics
- **OPTIONAL Office hours:** (Fridays 1pm-2pm) Ask me anything about the unit

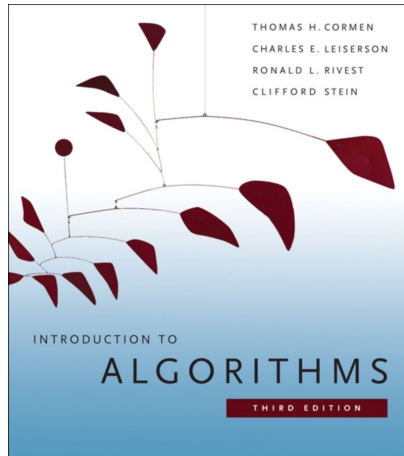
Assessment

- Exam: Counts 50% towards your final mark in the joint unit “Object-Oriented Programming and Algorithms”
- You pass the joint unit if your final grade is at least 40%

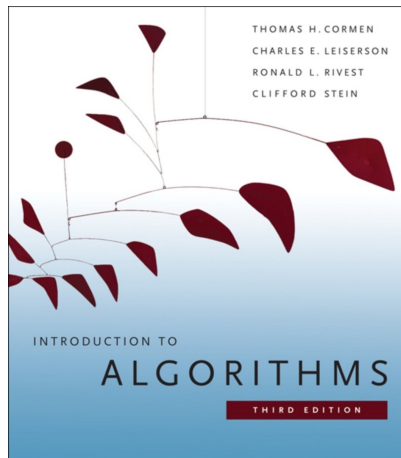
Teaching Staff

- **Unit Director:** Dr Christian Konrad
(christian.konrad@bristol.ac.uk)
- **Lead TA:** Kheeran Naidu
(kn16063@bristol.ac.uk)
- **TAs:** Imran Zamin Ali, Cezar Alexandru, Elliot Buckingham, Jeremy Colfer, Adithiya Diddapur, Charlotte Dillon, Amos Holland, Ollie Mamuda, Philip Mortimer, Michael Poluektov, Nathalie Alexandra Tcherdakoff, Viet-anh Tran, Harry Tune

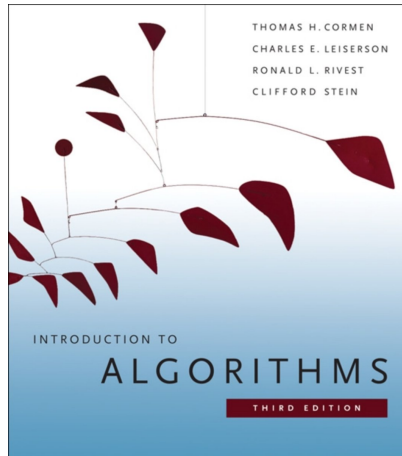




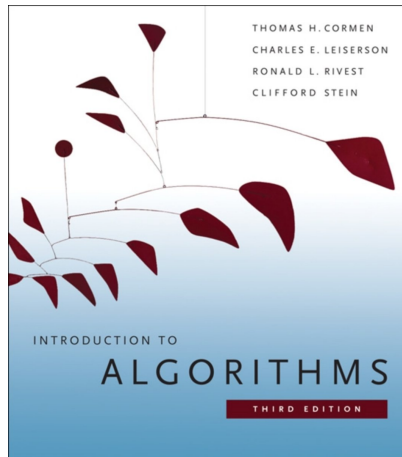
- More details on many of the topics discussed in this unit



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



- 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

- 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/2022_2023_COMS10017/coms10017.html`

This week

- Monday 10am-11am: Lecture
- Thursday 1pm-2pm: OPTIONAL Drop-in Session
- Thursday 3pm-4pm: Lecture
- Friday 1pm-2pm: OPTIONAL office hours
- Exercise sessions start next week



Good luck and enjoy! Questions?