Introduction
COMS10018 - Algorithms

Dr Christian Konrad

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

Brain Behind Your Software

Algorithms:

- Fabric that Software is made of
- Inner logic of your Software
- Insufficient computational power $\rightarrow$ 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 $D_1, D_2, \ldots, D_{\log n} \leftarrow \{\}$
2: For every $a \in A$: $d(a) \leftarrow 0$
3: $U \leftarrow \varnothing$ {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 $D_1, D_2, \ldots, D_{\log n} \leftarrow \{\}$
2: For every $a \in A$: $d(a) \leftarrow 0$
3: $U \leftarrow \varnothing$ {Keep track of dominated nodes ($U \subseteq B$ always holds)}
4: For every $b \in B$: $C(b) \leftarrow \bot$ {Output cover certificate}

Goals: First steps towards becoming an algorithms designer

What you get out of this unit

Goals: First steps towards becoming an algorithms designer

1. Learn techniques that help you design & analyze algorithms

Algorithm 1 Single-pass Semi-Streaming Algorithm for MDS

Require: Bipartite input graph $G = (A, B, E)$ with $|A| = |B| = n$
1: Let $D_1, D_2, \ldots, D_{\log n} \leftarrow \{\}$
2: For every $a \in A$: $d(a) \leftarrow 0$
3: $U \leftarrow \varnothing$ {Keep track of dominated nodes ($U \subseteq B$ always holds)}
4: For every $b \in B$: $C(b) \leftarrow \bot$ {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

Algorithm 1 Single-pass Semi-Streaming Algorithm for MDS

Require: Bipartite input graph $G = (A, B, E)$ with $|A| = |B| = n$
1: Let $D_1, D_2, \ldots, D_{\log n} \leftarrow \{\}$
2: For every $a \in A$: $d(a) \leftarrow 0$
3: $U \leftarrow \varnothing$ {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

Algorithm 1 Single-pass Semi-Streaming Algorithm for MDS

Require: Bipartite input graph $G = (A, B, E)$ with $|A| = |B| = n$
1: Let $D_1, D_2, \ldots, D_{\log n} \leftarrow \{\}$
2: For every $a \in A$: $d(a) \leftarrow 0$
3: $U \leftarrow \varnothing$ {Keep track of dominated nodes ($U \subseteq B$ always holds)}
4: For every $b \in B$: $C(b) \leftarrow \bot$ {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

Algorithm 1 Single-pass Semi-Streaming Algorithm for MDS

Require: Bipartite input graph $G = (A, B, E)$ with $|A| = |B| = n$
1: Let $D_1, D_2, \ldots, D_{\log n} \leftarrow \{\}$
2: For every $a \in A$: $d(a) \leftarrow 0$
3: $U \leftarrow \varnothing$ {Keep track of dominated nodes ($U \subseteq B$ always holds)}
4: For every $b \in B$: $C(b) \leftarrow \bot$ {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 $D_1, D_2, \ldots, D_{\log n} \leftarrow \{\}$
2: For every $a \in A$: $d(a) \leftarrow 0$
3: $U \leftarrow \varnothing$ {Keep track of dominated nodes ($U \subseteq B$ always holds)}
4: For every $b \in B$: $C(b) \leftarrow \bot$ {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

- 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

Unit Structure

Teaching Sessions

- **Lectures:** Mondays 3pm, Fridays 9am
- **Problem sheet sessions:** (Tuesdays) TA-led problem sheet sessions, come prepared!
- **OPTIONAL Drop-in/discussion session:** (Fridays 1pm-2pm) ask questions about the material or other algorithms-related topics
- **OPTIONAL Office hours:** (Thursdays 12-1pm) Ask me anything about the unit

Unit Structure

Teaching Sessions

- **Lectures:** Mondays 3pm, Fridays 9am
- **Problem sheet sessions:** (Tuesdays) TA-led problem sheet sessions, come prepared!
- **OPTIONAL Drop-in/discussion session:** (Fridays 1pm-2pm) ask questions about the material or other algorithms-related topics
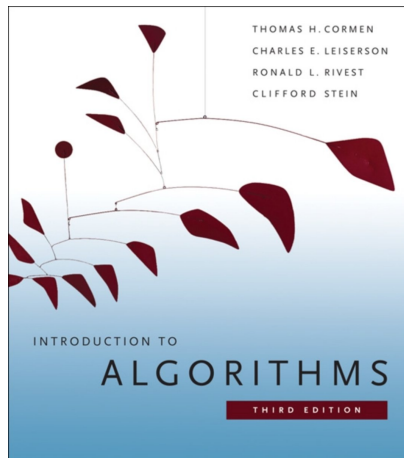- **OPTIONAL Office hours:** (Thursdays 12-1pm) Ask me anything about the unit

Assessment

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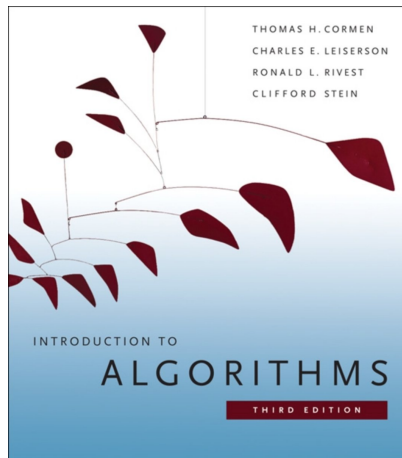- **Attention:** Algorithms exam is a must-pass component

Teaching Staff

- **Unit Director:** Dr Christian Konrad (christian.konrad@bristol.ac.uk)
- **Lead TA:** Adithya Diddapur (adi.diddapur@bristol.ac.uk)
- **TAs:** Alexander Bell, Artem Borisov, Michael Degamo, Amos Holland, Piotr Kozicki, Conor O'Sullivan, Thomas Parr, Kasim Rafiq, Ava Raisian, Charlie Walpole, Archie Walton, Eric Wang, Mara Yap
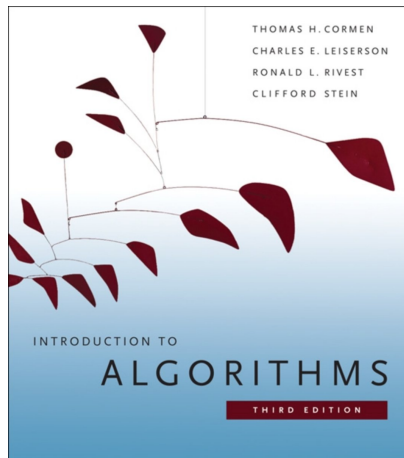
Book

Book

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

THOMAS H. CORMEN
CHARLES E. LEISERSON
RONALD L. RIVEST
CLIFFORD STEIN

INTRODUCTION TO
ALGORITHMS

THIRD EDITION

Book

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

THOMAS H. CORMEN
CHARLES E. LEISERSON
RONALD L. RIVEST
CLIFFORD STEIN

INTRODUCTION TO
ALGORITHMS
THIRD EDITION

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

THOMAS H. CORMEN
CHARLES E. LEISERSON
RONALD L. RIVEST
CLIFFORD STEIN

INTRODUCTION TO
ALGORITHMS
THIRD EDITION

How to Succeed in this Unit
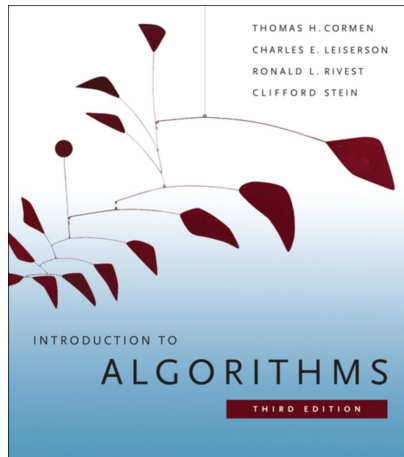
How to succeed

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/2024_2025_
COMS10018/coms10018.html

What's next

This Week

- Monday 3pm-4pm: Lecture (today)
- Friday 9am-10am: Lecture

Next Week and every subsequent week

- Monday 3pm-4pm: Lecture
- Tuesday: Small-group problem sheet sessions
- OPTIONAL Thursday 12-1pm: Office Hours
- Friday 9am-10am: Lecture
- OPTIONAL Friday 1pm-2pm: Drop-in Session

Good luck and enjoy! Questions?