Video 1: Introduction

COMS10017 - (Object-Oriented Programming and) 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**

What we want and how we work

Efficiency

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

Mathematics

What we want and how we work

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

What we want and how we work

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

What we want and how we work

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

What we want and how we work

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

What we want and how we work

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 $b \in B$: $C(b) \leftarrow \perp$ {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 $D_1, D_2, \ldots, D_{\log n} \leftarrow \{\}$
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

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: 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 $D_1, D_2, \ldots, D_{\log n} \leftarrow \{\}$
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 $b \in B$: $C(b) \leftarrow 1$ {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 $D_1, D_2, \ldots, D_{\log n} \leftarrow \{\}$
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 $b \in B$: $C(b) \leftarrow$ ⊥ {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 at hand, discover structure within problem, 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 $D_1, D_2, \ldots, D_{\log n} \leftarrow \{\}$
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 $b \in B$: $C(b) \leftarrow$ ⊥ {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 at hand, discover structure within problem, 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 $b \in B$: $C(b) \leftarrow \perp$ {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 at hand, discover structure within problem, 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

My Goals

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

My Goals

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

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

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

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

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

Unit Structure

Teaching Sessions

- **Video lectures:** Each video is assigned to a week (watch by end of week)
- **Problem sheet sessions:** (Mondays and Tuesdays) TA-led problem sheet sessions, come prepared!
- **Recap/Q & A/discussion session:** (Thursdays 2pm-3pm) Material recap, ask questions about the material
- **OPTIONAL Online office hours:** (Fridays 11am-12pm) Ask me anything about the unit

Unit Structure

Teaching Sessions

- **Video lectures:** Each video is assigned to a week (watch by end of week)
- **Problem sheet sessions:** (Mondays and Tuesdays) TA-led problem sheet sessions, come prepared!
- **Recap/Q & A/discussion session:** (Thursdays 2pm-3pm) Material recap, ask questions about the material
- **OPTIONAL Online office hours:** (Fridays 11am-12pm) Ask me anything about the unit

Assessment

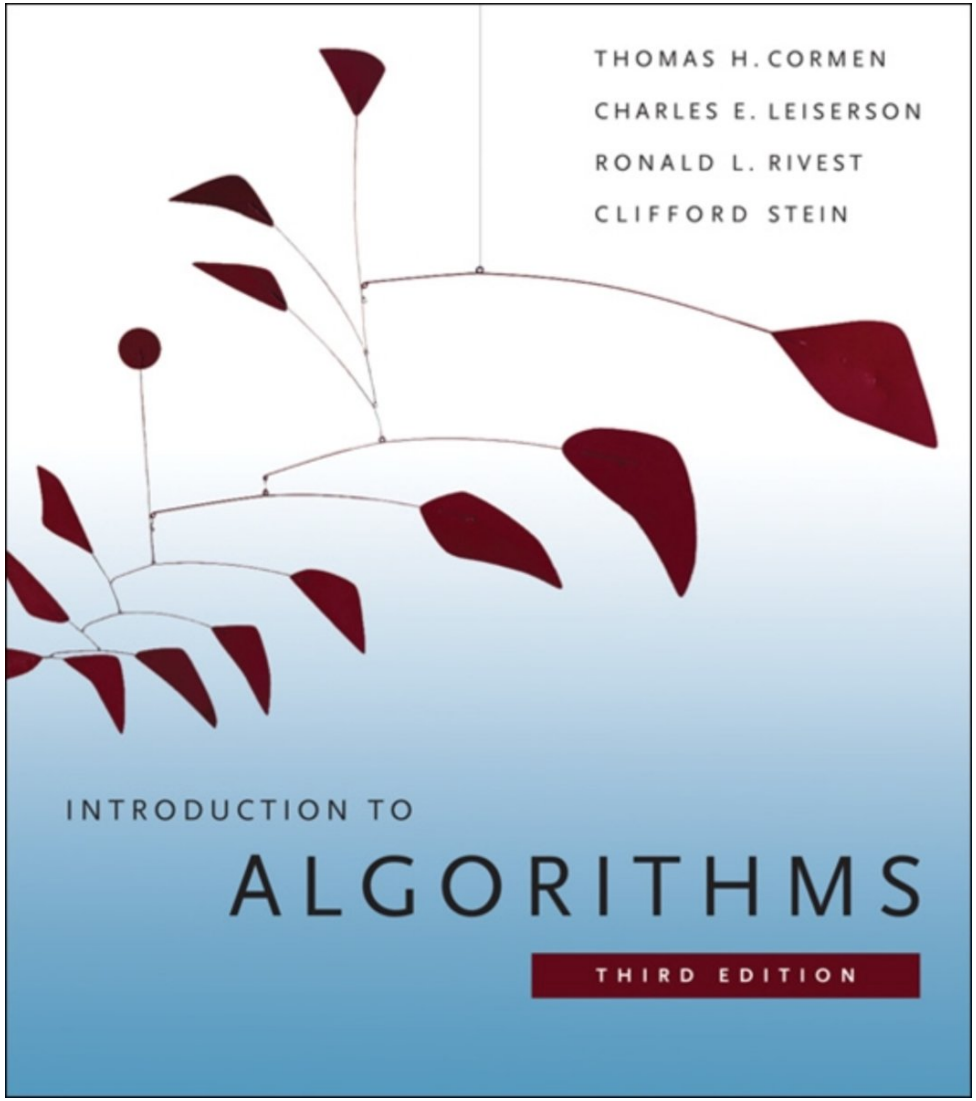
- Exam: Counts 50% towards your final mark in the joint unit "Object-Oriented Programming and Algorithms"
- You pass the joint unit if your final grade is at least 40%

Teaching Staff

Teaching Staff

- **Unit Director:** Dr Christian Konrad (christian.konrad@bristol.ac.uk)
- **Lead TA:** Kheeran Naidu (kn16063@bristol.ac.uk)
- **TAs:** Robert Gabriel Popescu, Cezar Mihail Alexandru, Charlotte Dillon, George Edward Nechitoaia, Llewellyn Forward, Matt Staveley-Taylor, Michael Polvekrov, Ralph Roberts, Satya Rammolian, Sergiu Aracatitei, Zak Duggan, Alex Carpenter

Book


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 discussion board on Piazza for discussions and questions

How to Succeed in this Unit

How to succeed

- Make sure you understand the material
- **Work on provided exercises!**
- Use discussion board on Piazza 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 discussion board on Piazza 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 discussion board on Piazza for discussions and questions
- **Work on provided exercises!!**
- **Work on provided exercises!!!**

Unit webpage: Use link on blackboard

`http://people.cs.bris.ac.uk/~konrad/courses/2021_2022_COMS10017/coms10017.html`

What's next

What to do now

- Check unit webpage
- Register at Piazza (discussion board) using link at unit webpage
- Watch video lectures for week 1

This week

- Tuesday 2pm-3pm: **Introduction**
- Thursday 2pm-3pm: Recap/Q&A/Discussion session
- Exercise sessions start next week

Good luck and enjoy!