Lecture 1

Algorithms & Data Structures

Goldsmiths Computing

October 1, 2018

Outline

Outline

Content

Algorithms

- searching
- sorting
- pathfinding
- matching

Content

Algorithms

- searching
- sorting
- pathfinding
- matching

Data structures

- pairs and vectors
- linear collections
- trees and graphs
- numbers
- hash tables

Content

General

- analysis
- computational thinking

Content

General

- analysis
- computational thinking

Practical

- measurement
- testing

Content

General

- analysis
- computational thinking

Practical

- measurement
- testing

Transferrable

- consistent working
- clear expression

Contact Time

- lectures: Monday 12:00–14:00, WB IGLT (here)
- labs (starting next week):
  - Tuesday 12:00–14:00, RHB 306/306a
  - Wednesday 10:00–12:00, RHB 306/306a

Extra help

- discussion forum
- study groups
- office hours, 25 St James room 18
  - Thursday 14:00–16:00

Assessment

Coursework: 50%

- quizzes
- labs
- peer assessments
- written work

Assessment

Coursework: 50%

- quizzes
- labs
- peer assessments
- written work

Exam: 50%

- bookwork
- problem-solving
- unseen questions

Coursework

Quizzes

- up to 20 in the year
- take them more than once
  - best attempt counts
  - enforced 4-hour break between attempts
  - use that break to review, ask questions on the forum, understand what you don't yet understand
- each quiz has a twelve-day open period
  - Monday 09:00–Friday 16:00

Coursework

Labs

- all 18 lab activities compulsory
  - no labs in first week of each term
- some will have assessment
  - upload to automated marking system
  - “instant” mark and feedback
  - resubmission allowed (sometimes)
  - variable deadlines (in-lab or take-home)
- labsheets
  - documents from learn.gold
  - code bundle and other materials over version control

Coursework

Peer assessments

- helping develop skills
  - deployment of code to unknown systems
  - critical assessment
  - written expression of ideas
- giving expected milestones
  - checkpoints before major deadlines

Coursework

Peer assessments

- helping develop skills
  - deployment of code to unknown systems
  - critical assessment
  - written expression of ideas
- giving expected milestones
  - checkpoints before major deadlines

Written work

- understanding of complex material
- ability to communicate understanding

Coursework

Working together

- what does “working together” mean?

Coursework

Working together

- what does “working together” mean?
- where is the line between good and not-OK?

Coursework

Working together

- what does “working together” mean?
- where is the line between good and not-OK?
- why all this coursework anyway?

Exam

Revision materials available next term:

- bookwork *vs* problem-solving
- seen *vs* unseen material
- choice *vs* compulsory

Exam

Revision materials available next term:

- bookwork *vs* problem-solving
- seen *vs* unseen material
- choice *vs* compulsory

Exam technique helps...

Exam

Revision materials available next term:

- bookwork *vs* problem-solving
- seen *vs* unseen material
- choice *vs* compulsory

Exam technique helps...

- ... but not as much as knowing the material

Reading material

Textbooks:

- Cormen, Leiserson, Rivest, Shamir, *Introduction to Algorithms* [CLRS]
- Dasgupta, Papadimitriou, Vazirani, *Algorithms* [DPV]
- Drozdek, *Algorithms in C++ | Java*

And also:

- academic papers
- online tutorials
- published source code
- video lectures
- blogs

Ground rules

Work

Outline

Motivation

- Describe the lab environment clearly
- Allow you to recreate it on personal computers

Ingredients

1. Compiler and runtime
    - Java
    - C++

Ingredients

1. Compiler and runtime
   - Java
   - C++
2. Build tool
   - Make

Ingredients

1. Compiler and runtime
   - Java
   - C++
2. Build tool
   - Make
3. Testing framework
   - JUnit 4
   - cppunit

Ingredients

1. Compiler and runtime
   - Java
   - C++
2. Build tool
   - Make
3. Testing framework
   - JUnit 4
   - cppunit
4. Version control system
   - git

Ingredients

1. Compiler and runtime
   - Java
   - C++
2. Build tool
   - Make
3. Testing framework
   - JUnit 4
   - cppunit
4. Version control system
   - git
5. Text editor (or IDE) of your choice

Installation: Linux (Debian/Ubuntu)

```
apt install <all the things>
```

1. Java:
   - `default-jdk`, `default-jdk-doc`, `default-jre`
   - `make`, `make-doc`, `git`
2. C++
   - `g++`, `gcc-doc`
   - `libcppunit-dev`, `libcppunit-doc`
   - `make`, `make-doc`, `git`

Installation: Windows (Java)

1. install msys2
2. run msys2 msys and install packages (`pacman -S`):
   - `make`, `git`
3. install Java (the latest JDK) from Oracle

Installation: Windows (C++)

1. install msys2
2. run msys2 msys and install packages (`pacman -S`):
   - `mingw64/mingw-w64-x86_64-gcc`
   - `mingw64/mingw-w64-x86_64-cppunit`
   - `make`, `git`
3. always run lab code using the MinGW 64-bit executable – it won't work, with confusing errors, if you use MSYS
4. but: always update your MSYS system using the MSYS executable

Installation: OS X (C++)

1. install xcode developer tools
   - `xcode-select --install`
2. install homebrew
   - sorry, you're on your own here
3. install cppunit
   - `brew install cppunit`

Installation: OS X (Java)

1. install xcode developer tools
   - `xcode-select --install`
2. install Java (the latest JDK) from Oracle

Work

1. Follow the instructions for your operating system to install the lab environment on your own computer
   - any problems: ask for help on the forum
2. Test your installation
   - start the environment
   - `git clone http://gitlab.doc.gold.ac.uk/crhodes/is52038b-labs`
   - `cd is52038b-labs/01/<lang>`
   - `make test`
   - Read the output carefully.
3. Select your programming language for labs and assignments from the choices provided on learn.gold.

Outline

Motivation

- describe programs
- independent of programming language
- intentionally as simple as possible

Definition

Pseudocode is an informal, high-level description of the operation of a computer program or other algorithm

Implications

- use simplest way to describe things

Definition

Pseudocode is an informal, high-level description of the operation of a computer program or other algorithm

Implications

- use simplest way to describe things
  - even if that is in English

Definition

Pseudocode is an informal, high-level description of the operation of a computer program or other algorithm

Implications

- use simplest way to describe things
  - even if that is in English
- not executable by a computer

Definition

Pseudocode is an informal, high-level description of the operation of a computer program or other algorithm

Implications

- use simplest way to describe things
  - even if that is in English
- not executable by a computer
  - walk-through by humans

Definition

Pseudocode is an informal, high-level description of the operation of a computer program or other algorithm

Implications

- use simplest way to describe things
  - even if that is in English
- not executable by a computer
  - walk-through by humans
  - reasonable

Variable assignment

Variable assignment is indicated by the ← symbol:

x ← 1

Variables in pseudocode do not need to be declared

Sequencing

Vertical space

Statements separated by vertical space happen in sequence

$x \leftarrow 1$

$y \leftarrow x$

$x \leftarrow 2$

What value does x have after this? What about y?

Sequencing

Vertical space

Statements separated by vertical space happen in sequence

$x \leftarrow 1$

$y \leftarrow x$

$x \leftarrow 2$

What value does x have after this? What about y?

Semicolons

Space sometimes gets tight, and more than one thing needs to go on a line. Semicolons separate statements in a sequence:

$x \leftarrow 1; y \leftarrow x; x \leftarrow 2$

Conditionals

if

Use **if then** to decide whether to do a sequence or not; end the sequence with **end if**

```
x ⟵ 0
if x > −6 then
    x ⟵ x + 1
end if
```

What value does x have after this?

Conditionals

if

Use **if then** to decide whether to do a sequence or not; end the sequence with **end if**

```
x ⟵ 0
if x > −6 then
    x ⟵ x + 1
end if
```

What value does x have after this?

Conditional Operators

Use mathematical notation (not code notation) in pseudocode:

- =, <, >
- ≤, ≥ (not <=, =>, >=)
- ∨, ∧, ¬

Conditionals

else

Use **else** to delimit a sequence to execute if the conditional is not true

```
x ⟵ 0
if x > 17 then
    x ⟵ x + 1
else
    x ⟵ x – 1
end if
```

What value does x have after this?

Conditionals

else if

Define chains of conditionals using **else if**. At most one of the sequences is executed.

```
x ← 0
if x > 3 then
    x ← 5
else if x > −3 then
    x ← 7
else if x > −8 then
    x ← 9
else
    x ← 11
end if
```

What is the value of x after this?

Work

1. Reading
   - CLRS, section 2.1
   - DPV, sections 0.1, 0.2
2. Quiz
   - available now on learn.gold
   - open until 16:00 Friday 12th October
   - try multiple times
   - mark is $30 + 70 \times (\text{score}/10)^2$