COMP9020

Foundations of Computer Science

Lecture 17: Course Review

Lecturers: Katie Clinch (LIC)
Paul Hunter
Simon Mackenzie

Course admin: Nicholas Tandiono

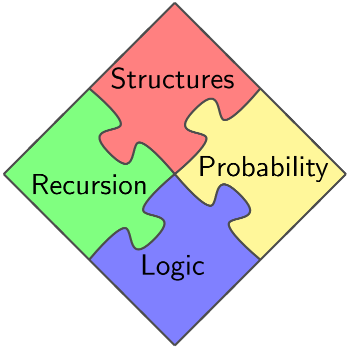
Course email: `cs9020@cse.unsw.edu.au`

Outline

Course Review

Goal: for you to become a competent computer **scientist**.

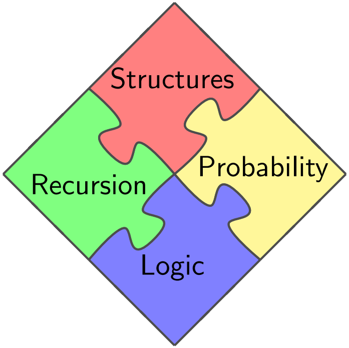
Requires an understanding of fundamental mathematics concepts:



Course Review

Goal: for you to become a competent computer **scientist**.

Requires an understanding of fundamental mathematics concepts:

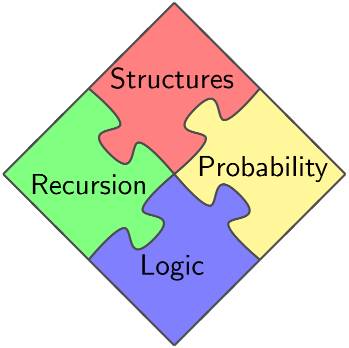


Topic 0: Number Theory

Course Review

Goal: for you to become a competent computer **scientist**.

Requires an understanding of fundamental mathematics concepts:



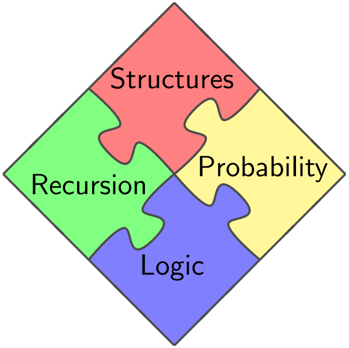
Topic 1: Structures

- Sets
- Graph Theory
- Relations
- Functions

Course Review

Goal: for you to become a competent computer **scientist**.

Requires an understanding of fundamental mathematics concepts:



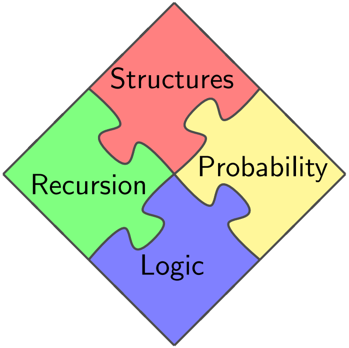
Topic 2: Recursion

- Recursion
- Induction
- Algorithmic Analysis

Course Review

Goal: for you to become a competent computer **scientist**.

Requires an understanding of fundamental mathematics concepts:



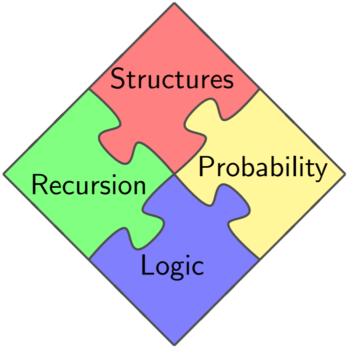
Topic 3: Logic

- Boolean Logic
- Propositional Logic

Course Review

Goal: for you to become a competent computer **scientist**.

Requires an understanding of fundamental mathematics concepts:



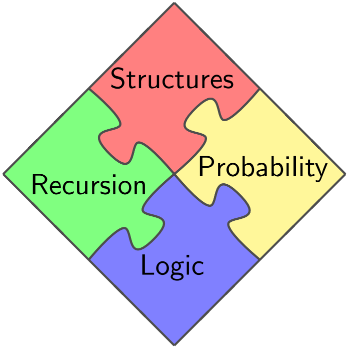
Topic 4: Probability

- Combinatorics
- Probability
- Statistics

Course Review

Goal: for you to become a competent computer **scientist**.

Requires an understanding of fundamental mathematics concepts:



In CS/CE these are used to:

- formalise problem specifications and requirements
- develop abstract solutions (algorithms)
- analyse and prove properties of your programs

Assessment Summary

During the semester:

- **Quizzes.** Best 7 out of 9 – (10 marks)
- **Assignments.** 4 assignments worth 7.5 marks each – (30 marks)

After the semester:

- **Final exam** – (60 marks)

NB

You must achieve 40% on the final exam AND 50% overall to pass the course.

Outline

Final Exam - preparation

Goal: to check whether you are a competent computer scientist.

Requires you to demonstrate:

- understanding of mathematical concepts
- ability to apply these concepts and explain how they work

Lectures, quizzes and assignments have built you up to this point.

Examiner's comments

The questions are intended to assess your **understanding** and your **ability** rather than your knowledge.

Unless specified, *any* valid (mathematical) proof technique is acceptable.

Partial marks are always available for incomplete answers.

How hard is it?

A common question is how will the exam compare to assignment/quiz/practice questions?

The exam situation is significantly different to assignments.

- You have less time for the exam, so on the face of it questions may appear harder (even if this is not the case).
- We are aware of the reduced timeframe, so exam marking will likely be more forgiving than assignment marking.
- Assignments had a wide range of difficulty.

Answer

Assignment questions are indicative of the questions likely to appear on the exam.

Final exam - revision strategy

- Re-read lecture slides
- Read the corresponding chapters in the books (LLM; R & W; Rosen)
- **Review/solve assignments and quizzes**
- Solve more problems from the books
- Attempt sample exams

(Applying mathematical concepts to solve problems is a skill that improves with practice)

Outline

Final exam - access on the day

Available: Tuesday 30th April, 08:15 (AEST)
Due: Tuesday 30th April, 12:00 (AEST)
Duration: 3 hours, 15 minutes.
Access: Inspera through Moodle

- It is a remote exam on Inspera, you can complete the exam anywhere, on your own computer.
- You can start the exam at any time between 8:15am and 8:45am. From the time you start, you will have 3 hours and 15 minutes to complete the exam (so if you start the exam at 8:15am, you will lose access to it at 11:30am).
- You have 15 minutes to read the questions, plus 3 hours to answer questions.
- The exam auto-saves frequently, so if you have technical difficulties during the exam, you should still have a lot of your work saved. It also auto-submits your final version.
- If you have any technical issues during the exam, please contact the course account: **cs9020@cse.unsw.edu.au**.

Final exam - contents

There will be :

- 9 Questions, roughly 3 parts per question
- Each question is worth 20 marks. So 180 marks total
- Aim to spend about 20 minutes on each question.

Each question will roughly correspond to one week of lectures:

1. Number theory
2. Set theory and languages
3. Graph theory
4. Relations
5. Functions and algorithmic analysis
6. Recursion and induction
7. Boolean and propositional logic
8. Combinatorics
9. Probability and statistics

Final exam - contents

Question contents

- Each question will contain roughly 3 parts
- Parts are mostly long-answer questions that you input in text-boxes (e.g. proofs/counterexamples)
- Within each 20 mark / 20 minute question:
  - some parts will be easier and some parts will be harder.
  - the number of marks does not indicate how easy or hard a part is.
  - the number of marks does not indicate how much time you should spend on each part
- If you spot any issues with the questions during the final exam, please contact **cs9020@cse.unsw.edu.au**
- If this results in us needing to make and corrections or clarifications, then we will announce these on webCMS during the exam. So keep an eye on **webCMS announcements**!

Final exam - Materials

The questions are intended to assess your understanding and your ability rather than your knowledge.

Access to **passive sources** (i.e. *existing* material) is permitted – should be properly referenced.

Accessing **active sources** (e.g. collaborating with other students, actively seeking answers through forums/social media/ChatGPT, contract cheating, etc) is a breach of the University's Plagiarism Policy and will be subject to disciplinary action as outlined in the Student code of conduct

Strongly recommended to have external writing materials for rough working

NB

No concepts other than those taught in this course are assumed.

Final exam - inputting your answers

The exam is online in Inspera.

- In the text-boxes you can enter plain text and/or unicode and/or the built-in LaTeX editor.
- The course symbols toolbar and LaTeX resources will be available to help with inputting mathematical symbols.
- The proof assistant will not be available.
- As long as your notation is clear, anything is acceptable (e.g. `v_0`, or `x^2`, or `phi`). If in doubt, explain your notation.
- For diagrams, there is a built-in drawing tool. Copy-paste of images is not fully supported.

In case of technical difficulties, we provide two extra input sections:

- One section of text-boxes where you can input answers for e.g. drag-and-drop questions if the drag-and-drop functionality is not working.
- One section for pdf uploads for if you run out of time inputting your answers into the text-boxes and need to photograph your rough work instead.

Final exam - Special consideration

Review the UNSW policy on Special Consideration.

UNSW has a “fit-to-sit” policy: by undertaking the assessment you are declaring that you are fit to do so.

If there are any foreseeable issues you must apply before the exam. If circumstances prevent you from applying before the exam, you must apply as soon as possible (within 3 days).

NB

Supplementary exams are only available to students granted Special Consideration.

Final exam - Technical issues

If you experience technical issues before or during the exam, you should follow these instructions:

- Take screenshots (including time and date) of as many of the following as possible:
  - error messages
  - screen not loading
  - timestamped speed tests
  - power outage maps
  - messages or information from your internet provider regarding the issues experienced
- Make contact with us immediately through **cs9020@cse.unsw.edu.au** and advise us of the issue.
- Submit a Special Consideration application immediately at the conclusion of your assessment and upload your screenshots.
- Note: There are "blank pages" at the end of the exam where you can enter the answer in case of minor technical issues (see slide 14).

Outline

Week 1 - proofs and number theory

Recap

- What is a proof?
  - Nature of propositions
  - Examples of proof strategies
    - Direct proof, Contrapositive, Proof by contradiction
- Number theory
  - Floor $\lfloor\cdot\rfloor$ and ceiling $\lceil\cdot\rceil$ functions
  - Absolute value $|\cdot|$ function

Techniques

- How to present proofs

Week 2 - Formal languages and set theory

Recap

- Symbols, words, languages
- Language definitions: Σ^*, length(), concatenation, Kleene star

Techniques

- Definitions around formal languages
- Proofs using Laws of Set Operations

Week 2 - Formal languages and set theory

Recap

- Set notation: $\in$, $\emptyset$, $\mathcal{U}$, $\subseteq$, $\{\ldots\}$, $[\ldots]$, $(\ldots)$
- Set operations: $\cap$, $\cup$, ^{c}, $\setminus$, $\oplus$, Pow(), $\times$
- Cardinality: $|X| = \#(X) = \mathrm{card}(X)$
- Venn diagrams

Techniques

- How to define sets
- Proofs using the Laws of Set Operations
- Cartesian product
- Cardinality computations
- Proving two sets A and B satisfy $A = B$ either (i) element-wise, (ii) using $A \subseteq B$ and $B \subseteq A$, or (iii) using the Laws of Set Operations

Week 2 - Laws of Set Operations

For all sets A, B, C:

Commutativity	$A \cup B = B \cup A$
	$A \cap B = B \cap A$
Associativity	$(A \cup B) \cup C = A \cup (B \cup C)$
	$(A \cap B) \cap C = A \cap (B \cap C)$
Distribution	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
Identity	$A \cup \emptyset = A$
	$A \cap \mathcal{U} = A$
Complementation	$A \cup (A^c) = \mathcal{U}$
	$A \cap (A^c) = \emptyset$

Week 2 - Other useful set laws

Idempotence	$A \cap A = A$
	$A \cup A = A$
Double complementation	$(A^c)^c = A$
Annihilation	$A \cap \emptyset = \emptyset$
	$A \cup \mathcal{U} = \mathcal{U}$
de Morgan's Laws	$(A \cap B)^c = A^c \cup B^c$
	$(A \cup B)^c = A^c \cap B^c$

Theorem (Principle of Duality)

If you can prove $A_1 = A_2$ using the Laws of Set Operations then you can prove $dual(A_1) = dual(A_2)$

Theorem (Uniqueness of complement)

$A \cap B = \emptyset$ *and* $A \cup B = \mathcal{U}$ *if, and only if,* $B = A^c$.

Week 3 - Graph theory

Recap

- Definitions and notation: vertices, edges, paths, cycles, connectedness, isomorphisms
- Graph classes: Trees, Complete graphs, complete k-partite graphs
- Graph traversals: DFS/BFS, Eulerian path/circuit, Hamiltonian path/cycle
- Graph properties: Chromatic number, Clique number, Planarity

Techniques

- How to model "real-world" problems using graphs
- How to show/calculate graph properties:
  - Connectedness
  - Eulerian/Hamiltonian traversals
  - Chromatic number
  - Clique number
  - Planarity
- Interactions between graph properties

Week 4 - Relations

Recap

Relations:

- Definitions, *n*-ary relations

Binary Relations:

- Representations: Matrix, graphical, directed graph
- Relational image ($R(A)$), converse relation ($R^{\leftarrow}$), inverse image ($R^{\leftarrow}(B)$), relation composition ($R; S$)
- Properties: Fun, Tot, Inj, Sur
- Properties: Reflexivity, Antireflexivity, Symmetry, Antisymmetry, Transitivity
- Functions: Domain, codomain, image

Week 4 - Relations

Equivalence Relations:

- Generalize equality
- Reflexive, Symmetric, Transitive
- Equivalence classes, Partitions

Partial Orders:

- Generalize $\leq$
- Reflexive, Antisymmetric, Transitive
- Hasse diagram, minimum vs minimal, glb/lub, lattice

Techniques

- Relational image, relational composition
- Properties of binary relations – examples and non-examples
- Equivalence classes
- Hasse diagram, glb/lub

Week 5 - Functions

Recap

Partial Orders:

- Reflexive, Antisymmetric, Transitive
- Hasse diagram, minimum vs minimal, glb/lub, lattice, topological sort, product/lexicographic/lenlex order

Functions:

- Functional composition: $g \circ f = f; g$
- Inverse function: $f^{\leftarrow}$, when it is a function
- Matrices: Add, Scalar multiplication, Matrix multiplication, Transpose
- big-O notation: O, Ω, Θ

Techniques

- Hasse diagram, glb/lub, topological sort, lexicographic/lenlex order
- Inverse function, function composition
- Basic matrix definitions and operations
- How to compare a variety of functions using big-O

Week 5 - Algorithmic analysis

Recap

- Count "cost" (default: running time) of elementary operations as a function of (a parameter of) the input
- Approximates real-world cost
- Using both big-O and worst-case to simplify analysis
- Recursive algorithms lead to recurrence equations

Techniques

- How to compute algorithm run-time in a variety of situations

Week 7 - Recursion and Induction

Due to Easter, this was actually weeks 5 (recursion) and 7 (induction).

Recursion: Defining more complex objects/processes in terms of simpler ones

Recap

- Recursive datatypes: Natural numbers, Words, Expressions, Well-formed formulas
- Recursive programming/functions: Factorial, concatenate, length
- Recurrence equations
  - Unrolling
  - Approximating with big-O
  - Master Theorem

Techniques

- How and when to define datatypes and functions recursively
- How to solve recurrence equations (up to big-O)

Week 7 - Recursion and Induction

Induction: Proving properties about recursively defined objects.

Recap

- Basic induction
- Variants of basic induction
- Structural induction

Techniques

- How to use induction to prove results about natural numbers
- How to use structural induction on recursive structures

Week 7 - Master Theorem

The following result covers many recurrences that arise in practice (e.g. divide-and-conquer algorithms)

Theorem

Suppose

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

where $f(n) \in \Theta(n^c (\log n)^k)$.

Let $d = \log_b(a)$. *Then:*

- **Case 1:** *If* $c < d$ *then* $T(n) = \Theta(n^d)$
- **Case 2:** *If* $c = d$ *then* $T(n) = \Theta(n^c (\log n)^{k+1})$
- **Case 3:** *If* $c > d$ *then* $T(n) = \Theta(f(n))$

Week 8 - Boolean logic and propositional logic

Recap

- 2-element Boolean Algebra
- Boolean functions
- CNF/DNF, canonical DNF
- Karnaugh maps, optimal DNF

Techniques

- How to compute algorithm run-time in a variety of situations
- Properties of 2-element Boolean Algebra
- How to compute canonical CNF/DNF
- How to prove properties of Boolean Algebras

Week 8 - Boolean logic and propositional logic

Recap

- Syntax:
    - Well-formed formulas
    - Parse trees
    - CNF and DNF
- Semantics:
    - Truth assignments
    - Truth tables
    - Tautologies, Contradictions, Contingencies
    - Logical equivalence
    - Entailment

Techniques

- How to prove/disprove logical equivalence and logical entailment
- How to model problems using Propositional Logic

Week 9,10 - Combinatorics, probability, statistics

Recap

Combinatorics

- Basic counting rules: Disjoint sets, Cartesian products
- Permutations and Combinations
- Balls in boxes
- Using recursion to count
- Approximate counting

Probability

- Sample spaces, probability distributions, events
- Independence
- Recursive probability calculations

Statistics

- Random variables
- Expectation, linearity of expectation
- Variance, standard deviation

Week 9,10 - Combinatorics, probability, statistics

Techniques

- How to count the number of elements in various sets:
  - Counting selections - which one to apply and how
  - Identifying symmetries
  - Using recursion to count
- How to calculate probabilities in various situations:
  - Uniform distributions
  - Event combinations
  - Sequences of independent events
  - Recursive scenarios
- Interaction between properties such as independence, conditionality
- How to calculate expected value of random variables in various situations

Weeks 9,10 - Selecting items summary

Selecting k items from a set of n items:

With replacement	Order matters	Balls in boxes	Formula
Yes	Yes	Distinguishable balls Multiple balls per box	n^k
No	Yes	Distinguishable balls At most one ball per box	$(n)_k$
No	No	Indistguishable balls At most one ball per box	$\binom{n}{k}$
Yes	No	Indistinguishable balls Multiple balls per box	$\left(\!\binom{n}{k}\!\right) = \binom{n+k-1}{k}$

Outline

Sample Question - Graph Theory

Draw a single graph with 6 vertices and 10 edges that satisfies each of the following:

a. is planar,
b. contains a Hamiltonian circuit, and
c. does not contain an Eulerian path.

Sample Question - Relations

Consider the relation $R \subseteq \mathbb{R} \times \mathbb{R}$ defined by aRb if, and only if, $b + 0.5 \geq a \geq b - 0.5$. Is R

a. reflexive?
b. antireflexive?
c. symmetric?
d. antisymmetric?
e. transitive?

Sample Question - Functions

Let $\Sigma = \{a, b\}$ and define $f : \Sigma^* \to \mathbb{R}$ recursively as follows:

- $f(\lambda) = 0$,
- $f(aw) = \frac{1}{2} + \frac{1}{2}f(w)$ for $w \in \Sigma^*$, and
- $f(bw) = -\frac{1}{2} + \frac{1}{2}f(w)$ for $w \in \Sigma^*$.

Prove that f is injective.

Sample Question - Algorithmic Analysis

Provide an asymptotic upper bound for $T(n)$, the running time of the following code snippet:

```
my_func(n):
    if n = 0:
        print('*')
    else:
        i = 0
        while i < n:
            my_func(i)
            my_func(n-i-1)
            i = i + n/2
        end while
    end if
```

Sample Question - Recursion/Induction

Let $\Sigma = \{1, 2, 3\}$.

a. Give a recursive definition for the function $\mathsf{sum} : \Sigma^* \to \mathbb{N}$ which, when given a word over Σ returns the sum of the digits. For example $\mathsf{sum}(1232) = 8$, $\mathsf{sum}(222) = 6$, and $\mathsf{sum}(1) = 1$. You should assume $\mathsf{sum}(\lambda) = 0$.

b. For $w \in \Sigma^*$, let $P(w)$ be the proposition that for all words $v \in \Sigma^*$, $\mathsf{sum}(wv) = \mathsf{sum}(w) + \mathsf{sum}(v)$. Prove that $P(w)$ holds for all $w \in \Sigma^*$.

c. Consder the function $\mathsf{rev} : \Sigma^* \to \Sigma^*$ defined recursively as follows:
   - $\mathsf{rev}(\lambda) = \lambda$
   - For $w \in \Sigma^*$ and $a \in \Sigma$, $\mathsf{rev}(aw) = \mathsf{rev}(w)a$

   Prove that for all words $w \in \Sigma^*$, $\mathsf{sum}(\mathsf{rev}(w)) = \mathsf{sum}(w)$

Sample Question - Logic

Let $(T, \wedge, \vee, ', 0, 1)$ be a Boolean Algebra. Define $\oplus : T \times T \to T$ as follows:

$$x \oplus y = (x \wedge y') \vee (x' \wedge y)$$

a. Prove using the laws of Boolean Algebra that for all $x \in T$, $x \oplus 1 = x'$.
b. Prove using the laws of Boolean Algebra that $x \wedge (y \oplus z) = (x \wedge y) \oplus (x \wedge z)$.
c. Find a Boolean Algebra (and x, y, z) which demonstrates that $x \oplus (y \wedge z) \neq (x \oplus y) \wedge (x \oplus z)$

Sample Question - Probability

A 4-letter word is selected at random from Σ^4, where $\Sigma = \{a, b, c, d, e\}$.

a. What is the probability that the letters in the word are distinct?
b. What is the probability that there are no vowels in the word?
c. What is the probability that the word begins with a vowel?

Good luck everyone!