Introduction

COMP4128 Programming Challenges

School of Computer Science and Engineering
UNSW Sydney

Term 2, 2025

Table of Contents

Instructors

- Lecturer: Raveen de Silva (he/him)
  - Email me: cs4128@cse.unsw.edu.au
- Workshop and lab staff: see timetable
- Join the Discourse forum

Course Goals

- Learn algorithms and data structures
- Develop problem solving ability
- Practice implementing algorithms in C++
- Practical evaluation of code correctness and running time
- Prepare for programming competitions

Why competitive programming?

- It's fun
  - Most of the time
  - For those who enjoy a challenge
- Become part of a community
  - Rapidly growing at UNSW
  - Active society (CPMSoc)
- Develop your skills
  - Learn to solve *self-contained* problems *quickly* and *accurately*
  - The exact skills required in most technical interviews!

Prerequisites

- Significant programming experience in C or C++
- Understanding of fundamentals from *Data Structures and Algorithms*
  - Arrays, structs, heaps, merge sort, BSTs, graph search, etc
- *[Extended] Algorithm Design and Analysis*, although most content will be reintroduced
- Most important: enthusiasm for problem solving

Topic Overview

- Problem Solving Paradigms
- Data Structures
- Dynamic Programming
- Graph Algorithms & Shortest Paths
- Network Flow
- Mathematics
- Computational Geometry

There is a tentative course schedule on the website.

Table of Contents

Lectures

- Tue 16:00 – 18:00 at Old Main 150
- Fri 14:00 – 16:00 at June Griffith M10
- Live streams and recordings on Echo360, via Moodle

Lectures

- Lectures for each topic will present the theory, and apply this to some example problems
- Any code in lectures will be in C++
- Slides will be available before each lecture
- Please ask questions at any time if anything is unclear

Consultation

- Weeks 1–10:
  - Tue 14:00 - 15:00 and Fri 13:00 – 14:00 at my office (K17 202)
  - Email me for other arrangements (remote and/or other times)
  - I'm not usually on campus other than Tuesday and Friday
- Additional consultations during STUVAC and the exam period, schedule TBA

Workshops and Labs

- All face to face
- See timetable for rooms

Workshops and Labs

- 120 minute workshop
  - Two to four example problems based on recent lectures
  - Work through problem sheets in small groups
- 90 minute lab
  - Work on the weekly problem sets with your classmates
  - Tutors will help you with the problem sets and other questions
  - Close the loop on problem diaries

Timetable Notes

- All times are in AEST (UTC+10)
- No tute/labs in week 6 (flexibility week)
- Lecture schedule in week 6 TBC
  - Likely one revision lecture, maybe one guest lecture

Table of Contents

Assessment Breakdown

- Weekly problem sets: 40%
- Problem diary: 8%
- Contests: 18%
- Final: 34%

Problem Sets

- A set of 5 problems will be released each week except week 6
- Problem sets are conducted on vjudge
  - Make an account using your zID as the username
  - Join our group
- Suggested timeframe is two weeks

Problem Sets

- Worth 5% each, drop the lowest, for a total of 40%
- Marks are awarded non-linearly. As a rough guide:
  - for PS, aim for 1 per set
  - for CR, aim for 2 per set
  - for DN, aim for 3 per set
  - for HD, aim for 4 per set

Problem Sets

- Some problems will take you minutes, others will take you days
- Work together
  - You are encouraged to discuss problems and share test cases
  - Code must be derived and written individually
  - Acknowledge any collaboration in a header comment
  - Review plagiarism policy

Late Submissions

- No deadlines, no late penalties
  - Special Consideration *not* required
- Don’t fall behind!
  - Contact me and your tutor if you experience interruptions to your studies

Problem Diary

- Brief notes, usually much less than a page (excl code snippets) explaining:
  - your problem-solving process,
  - any challenges you encountered and
  - how you overcame them.
- Write about every problem up to your target grade, whether you solved it or not
- No need to give detailed descriptions or proofs as in the Algorithms courses
- With meta-reflection, worth 8%

Contests

- An account will be made for you on CMS, email coming soon with credentials
- Individual (unlike ICPC)
- Aims:
  - practice coding in a time-constrained environment
  - practice solving problems using a variety of available techniques
  - prepare for the final exam

Contest 1

- At the end of week 1, you will undergo a timed contest with 5 problems, to be completed within 48 hours
- No new material will be tested; only COMP2521 knowledge (e.g. sorting, recursion) is needed
- Test whether your programming fundamentals are sufficient to proceed to the later stages of the course
- We recommend that you try to complete the task within a shorter time frame, say 5 hours, but the full time is available in this case to minimise stress for you

Contests 2 and 3

- In weeks 6 and 9, you will undergo a timed contest with 3 problems
- Contest will be open for 48 hours; you have three hours from when you click ‘Start’
- Further details will be released closer to the date of each contest
- Each problem will be worth 100 points and have a 50 point subtask
- Marks are awarded non-linearly. As a rough guide:
  - for PS, aim for 50 points
  - for CR, aim for 100 points
  - for DN, aim for 150 points
  - for HD, aim for 200 points

Final Exam

- The final exam will be a timed contest with ~ 7 problems, to be completed within ~ 5 hours
- Held at CSE labs, prewritten code allowed
- Further details will be released closer to the date of the exam

Table of Contents

What's Changed

- Longer workshops
- Written diaries only
- Diary marks
- Extra problem set, drop lowest
- Supp exam
- More supplemental resources

What Hasn't Changed

- CMS
- Formatif
- Practice problems
- Computational geometry
- POGIL in workshops

Table of Contents

Competitions: ICPC

- Preliminary Contest on 14th September
- SPAR
    - Practice contests
    - One bonus mark for participating in any of the remaining rounds

Competitions: other

- CPMSoc
  - Term 2 Chicken Contest underway
  - SPARE contests?
- Tech companies
  - Meta: Hacker Cup

Practice

- The best practice is to solve lots of interesting problems
- Join CPMSoc
    - Fortnightly workshops
- Online problem sets and competitions
    - Online judges: Codeforces, TopCoder, CodeChef, AtCoder, etc
    - Informatics Olympiad training resources: USACO, ORAC
    - Maths: Project Euler

Table of Contents

The Anatomy of a Contest Problem

- Problem statement, describing the problem using flavour text
- Input and output specification
- Constraints
- Time limit (usually 1s) and memory limit (usually enough)
- Sample testcases, sometimes with explanation

Judging

- Your program will first be compiled
  - If this fails, you get `COMPILE-ERROR`
  - C++ compile errors are notoriously opaque
- Your program will then be run on the sample testcases and several secret testcases, including
  - large cases for stress testing
  - edge cases to catch bugs

Verdicts

- There are several reasons for your submission to be unsuccessful
    - `WRONG-ANSWER`: your program produced incorrect output for at least one test case
    - `TIME-LIMIT`: your program exceeded the time limit for at least one test case
    - `RUN-ERROR`: many possible reasons, but most commonly because your program crashed for at least one test case
    - If more than one of these apply, you could get any of them (depends on the judge)
- The `CORRECT` verdict is given if your program produced correct output within the time limit for every test case

How to Solve a Problem

- Read the problem statement
  - Reformulate and abstract the problem away from the flavour text
  - Check carefully for any special conditions which might be easy to miss – seemingly small changes to the statement can change the problem greatly
- Identify the input and output specification and any constraints that apply
- Confirm your understanding of the problem using the sample cases

How to Solve a Problem

- Design an algorithm to solve the problem
  - Estimate the runtime of your algorithm
- Implement the algorithm
  - Test the implementation
  - Debug the implementation – often the most time consuming step
- Submit!

Example problem: A + B

- **Problem statement** Alice and Bob are two friends who are visiting a milk bar. The milk bar is owned by the crotchety old Mr Humphries. If Alice buys A dollars worth of items and Bob buys B dollars, how much must they pay in total?
- **Input** Two integers, A and B ($0 \leq A, B \leq 10$)
- **Output** A single integer, the total amount Alice and Bob must pay.

Example problem: A + B

- **Problem** Output $A + B$

- **Algorithm** Calculate $A + B$, and then print it out.

Example problem: A + B

- **Complexity** $O(1)$ time and $O(1)$ space

- **Implementation**

```
#include <iostream>
using namespace std;

int main() {
  // read input
  int a, b;
  cin >> a >> b;

  // compute and print output
  cout << (a + b) << '\n';
}
```