**Course content:** Algorithm analysis, lower bound and upper bound, recurrences, divide-and-conquer, dynamic programming, greedy algorithms, data structures, priority queues, binary search trees, red-black trees, hash tables, graph search techniques, Depth-First Search, Breadth-First Search, graph algorithms, minimum spanning trees, single source shortest path trees, all pairs of shortest paths, and more!

**Lecturer:** Prof Weifa Liang, Room N334, CSIT building.

**Tutors:** Mr Mojtaba Rezvani, Mr Meitian Huang, Mr Yu Ma, and Dr Jinbo Huang

**Textbook:** Introduction to Algorithms.
Lectures and tutorials

There will be approximately 28 lectures. Since there are 36 lecture slots (12 weeks $\times$ 3 lectures per week), some slots will be unused. The current schedule is the temporary one, subject to changes if necessary. Should there be such changes, it will be announced in the course forum in Wattle, and on the course web page.

There will be six combined tute-labs, in weeks 2, 4, 6, 8, 10, and 12. They will be 2 hours long and conducted partly in tutorial style and/or partly in lab style. Registration now is open, and use Wattle to register yourself into one of the seven groups.
Quizzes and Assignments

There will be four quizzes (or mini-assignments) with due dates on Fridays, the mark allocations to the four quizzes are 3%, 3.5%, 4.5% and 4% of the final mark.

There will be two assignments.

- The first assignment (due date: Friday, 1 September), worth 15% of the final mark.

- The second assignment (due date: Friday, 27 October), worth 20% of the final mark.

Answer sheets of all quizzes and assignments are preferred to be submitted electronically through Wattle.

COMP6466/Honours students will have an additional question in each assignment. In addition, COMP6466/Honours students may be held to a higher standard.
Course Assessment

The course assessment consists of

- Four Quizzes, 15% of the total
- Two Assignments, 35% of the total
- One Mid-Semester Exam, 20% of the total
- One Final Exam, 30% of the total

Following the School assessment policy, the final mark will be calculated by the following formula, subject to necessary adjustments during the School examination meeting.

$$Total = 0.15 \times Q + 0.35 \times A + 0.2 \times M + 0.3 \times F,$$

$Q$ is the quiz mark and $A$ is the assignment mark, $M$ and $F$ are mid-semester and final exam marks.
Course Assessment

➢ The university-wide late submission penalty policy will be applied, i.e., 5% mark deduction per working day (from Monday to Friday except public holidays) after a submission deadline.

➢ If there is any dispute on the marking and marking score of an assessment item, the request for the reassessment of that item (quizzes and assignments) must proceed within TWO weeks after the marks and solutions of that item have been released. Beyond this two-week period, no further reassessment requests will be considered.
The courses consists of four modules, they are

- Mathematical Tools
- Advanced Data Structures/Graph Search Techniques
- Design Methodologies
- Graph Algorithms.
Other things

Discussion forums in Wattle:

- **Announcements** – for official announcements
- **Student forum** – for your discussions

As always, the forums can be accessed through your Wattle account, and are used for all related issues on study of this course, irrelevant issues (for example, political events, sports and music) are not permitted. Some things such as posting quiz or assignment solutions are not permitted in the forums.

- **Algorithms (Design and Analysis)** ≠ **Programming (Coding!)**

- Bonus questions and other types of posting questions are only for those who want to challenge themselves, they are not the requirements of this course!
Plagiarism issues

➤ The School’s policy on plagiarism will be enforced. A student in this course is expected to be able to explain and defend any submitted assessment item.

➤ The course convener can conduct or initiate an additional interview about any submitted assessment item for any student.

➤ If there is a significant discrepancy between the two forms of assessment, it will be automatically treated as a case of suspected academic misconduct.
Student Reps Recruitment!

We need to recruit TWO student representatives (one for COMP3600 and one for COMP6466) who will serve as the bridges between you and the course convenor.

If you are enthusiastic about this role, pls send me an email. If we have multiple candidates, we will exercise democratic voting, and vote one from the candidates.

As a student rep, you should contact both students and the course convenor to speak out their concerns, comments and suggestions, and the course convenor is expecting to receive your feedback at least no later than every second week.
Question: Given 8 identical coins, there is a fake one, which may be either heavier or lighter, compared with the rest of 7 coins. Assume that there is a balancing-scale available.

To identify the fake coin (might be lighter or heavier compared with the true one),

➢ how many rounds of comparisons are needed?

➢ how many rounds of comparisons are required if the number of coins is $n$?
The solution to the coin identification problem

- $a, b, c, d, e, f, g, h$

The diagram shows a decision tree for identifying which coins are fake. Each question node represents a coin, and the child nodes represent the possible outcomes of the coin being real or fake. The leaf nodes indicate whether the coin is fake or not.
Solution to the coin identification problem

- How many rounds of comparisons are needed? 3

- how many rounds of comparisons are required if the number of coins is \( n \)? \( \lceil \log n \rceil \)

If we also need to identify whether the fake one is lighter or heavier, it should be \( \lceil \log n \rceil + 1 \).
Chapter 0. Introduction

- Algorithms
- Algorithm Analysis
- Algorithm Design
0.1 What is an algorithm?

An algorithm is any well-defined computational procedure that takes some value or a set of values as input, and produces some value or a set of values as output.

Example one: Sort $n$ integers in increasing order.

Input: a sequence of $n$ integers, $a_1, a_2, \ldots, a_n$

Output: a permutation (rearrangement) of the numbers as $a'_1, a'_2, \ldots, a'_n$, such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

Example two: Given an integer $N$, determine whether it is a prime number.

Input: $N$

Output: YES if $N$ is a prime number; otherwise, NO.

Example three: Given an integer $N$, print out prime numbers no greater than $N$.

Input: $N$

Output: a sequence of prime numbers, and no number is larger than $N$. 

0.1 Definition of Algorithms (cont.)

For a given problem, there are many different algorithms for it, e.g., there are a number of algorithms for the sorting problem.

1. insertion sort
2. bubble sort
3. mergesort (*)
4. quicksort
5. shellsort
6. heapsort (*)
7. bucket sort (*) for integers only

(*) indicates the algorithm is an optimal algorithm. What is the optimal algorithm?
0.2 Algorithm Analysis

Analysis of an algorithm is to predict the resources that the algorithm requires. Resources are measured in relation to a model of computation, which is a mathematical abstraction of a computer on which the algorithm is implemented. In the following several popular computational models are listed.

- RAM: time and space (traditional computers)

- PRAM: parallel running time, number of processors, and read-and-write restrictions (SIMD type of parallel computers, multi-core desktops, MapReduce paradigm)

- Message Passing Model: communication cost (number of messages), and computational cost (usually the cost of local computation is ignored, distributed algorithms [computing], peer-to-peer networking, MIMD type of machines, etc)

- Turing Machine: time and space (abstract theoretical machine)
0.2 Algorithm Analysis (continued)

Important concepts used in the analysis of algorithms:

- **Input size**: If the input to an algorithm is “5413133”, is the input size 1, 7, 20 or 5413133? e.g., given a graph $G = (V, E)$, its size is $|V| + |E|$.

- **Running time (worst-case and average-case)**: The running time of an algorithm on a particular input is the number of primitive operations or steps executed. However, there is more than one idea about what a primitive operation is. Unless otherwise specified, we shall concentrate on finding only the worst case running time.

- **Order of growth**: To simplify the analysis of algorithms, we are interested in the growth rate of the running time, i.e., we only consider the leading term of a time formula. e.g., the leading term is $n^2$ in the expression $n^2 + 100n - 5,000$. 

Most analysis of algorithms of the type we will encounter in this course is based on the following assumptions.

- RAM (random access machine) model with unlimited memory.
- Operations such as arithmetic with small integers, boolean operations, fetching and storing small quantities, take unit time.
- Worst case performance is emphasised, but average and amortised performance is also important in practice.

Many other conventions exist (example: bit complexity) but we will not encounter them very much.
0.2 Algorithm Analysis (continued)

Insertion Sorting

Algorithm Insertion$(A)$

0 $n \leftarrow \text{length}[A]$
1 for $j \leftarrow 2$ to $n$ do
2     key \leftarrow A[j]
3 /* Insert $A[j]$ into sorted sequence $A[1 \ldots j - 1]$ */
4     $i \leftarrow j - 1$
5     while $i > 0$ and $A[i] > key$ do
6         $A[i + 1] \leftarrow A[i]$
7     $i \leftarrow i - 1$
8 $A[i + 1] \leftarrow key$
E.g.,

\[
\text{INSERTION-SORT}(A) \\
1 \text{ for } j \leftarrow 2 \text{ to } \text{length}[A] \\
2 \quad \text{do } key \leftarrow A[j] \\
3 \quad \triangleright \text{ Insert } A[j] \text{ into the sorted sequence } A[1 \ldots j-1]. \\
4 \quad i \leftarrow j-1 \\
5 \quad \text{while } i > 0 \text{ and } A[i] > key \\
6 \quad \text{do } A[i+1] \leftarrow A[i] \\
7 \quad \quad i \leftarrow i-1 \\
8 \quad A[i+1] \leftarrow key
\]

The running time of the algorithm is the sum of running times for each statement executed; a statement that takes \( c_i \) steps to execute and is executed \( n \) times will contribute \( c_i n \) to the total running time.\(^5\) To compute \( T(n) \), the running time of \text{INSERTION-SORT}, we sum the products of the \textit{cost} and \textit{times} columns, obtaining

\[
T(n) = c_1 n + c_2 (n - 1) + c_4 (n - 1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8 (n - 1).
\]

Cormen, p26
0.3 Algorithms Design

For a given problem, there are many ways to design algorithms for it (e.g. Insertion sort is an incremental approach). The following is a list of popular design approaches.

- Divide-and-Conquer (D&C)
- Greedy Approach
- Dynamic Programming (DP)
- Linear Programming
- Branch-and-Bound
- $\alpha$–$\beta$ Pruning
Divide-and-Conquer (D&C) Paradigm

Given a problem $A$, if the D&C strategy is applicable, the problem is then solved using the strategy. The detailed steps are as follows.

1. **Divide** $A$ into a number of subproblems with equal size roughly.

2. **Conquer** the subproblems by solving them recursively. For small enough subproblems, solve them directly without dividing them further.

3. **Combine** the solutions of the subproblems into the solution of the original problem.
Divide-and-Conquer (D&C) Paradigm (cont.)

An Example of D&C Strategy: Consider the merge-sort algorithm.

- **Divide:** divide the $n$-element sequence into two subsequences of $n/2$ elements each (or as close as possible if $n$ is odd)

- **Conquer:** sort the two subsequences recursively using the merge-sort

- **Combine:** merge the two sorted subsequences into a sorted sequence
sorted sequence

merge

initial sequence

Cormen, p35