Administrative Issues

EECS3101 E:
Design and Analysis of Algorithms
Fall 2025

CHEN-WEI WANG

Instructor

- How may you call me?

  “Jackie” (most preferred),

  “Professor Jackie”, “Professor”, “Professor Wang”, “Sir”, “Hey”, “Hi”, “Hello”
- When you need ***advice*** on the course, speak to me!
- Throughout the semester, feel free to suggest ways for helping your learning.

If You Are Not Enrolled Yet

- Send me an email ASAP requesting access to the course eClass site, with your *name*, *student number*, *Passport York ID*.
- Still keep up with lectures & study items (e.g., notes, tutorials).
- Still complete labs & tests (***no extension***).

Class Protocol

- If you ever had to act as a presenter, you would just agree that any of the following exhibitions from the audience gives you ***unpleasant*** and ***disrespectful*** feelings.
  - Talking

    I am **easily distracted** by noise (even when it's whispering).
    It is then **unfair** to your fellow students who want to learn.
    ⟹ **Only one** person talking at a time in the room please.
  - Using your laptop to do tasks **unrelated** to the current lecture

    ⟹ I'd rather that you do it **elsewhere**.
  - Using mobile phones

    ⟹ Please keep it to a ***minimum***!
- Slides are ***self-contained***, so I may **not** just read them off.
- I will focus on explaining core concepts with examples.
- Your ***engagement*** is the key: ask ***questions***!

Writing E-Mails to Your Instructor

- Think of me as your ***colleague*** who is happy to help you learn.
  - ***formality*** is unnecessary
  - ***courtesy*** is expected
- This sounds ***very rude*** (and may be delayed, if not ignored):

```
On the link you sent us for our mark
my mark for lab0 did not appear on it
and i submitted lab0 during my lab session
```

- This sounds ***much nicer***:

```
Hello Jackie, the link you sent didn’t work.
I did submit my lab0. Could you please look into this?
Thanks! Jim
```

- ***in-person*** communication may be the *most effective*

  Slow/No responses to your email inquiries ⇒
  Jackie is happy to help during office hours and/or appointments.

Course Information

- A single eClass site:
  - ***LE/EECS 3101 E - Design and Analysis of Algorithms (Fall 2025-2026)***
    - Announcements
    - Assignment Instructions
    - Written Part of Term Tests [instructions & submissions]
- Check your emails regularly!

Required Study Materials

- Lecture materials (recordings, iPad notes, slides, codes) will be posted for you to ***re-iterate concepts and examples***:

  `https://www.eecs.yorku.ca/~jackie/teaching/lectures/index.html#EECS3101_F25`

- The ***course syllabus*** is posted in the above lectures site.
- Though Jackie **attempts** to record each lecture entirely:
  - *Not meant to be a replacement for classes!*
  - The purpose of recording is that you can focus on reaching ***maximum comprehension***.
    - *Ask questions!*
    - Take (even ***incomplete***) notes: they help when re-visiting lectures.
    - Review points which you need to ***re-iterate*** from the recordings.
  - It'd be ***your call*** to use the posted ***lecture recordings***:
    - either as a way to **review** details not understood for the first time;
    - or as an **excuse** to skip classes!

Course Syllabus

Let’s go over the ***course syllabus***.

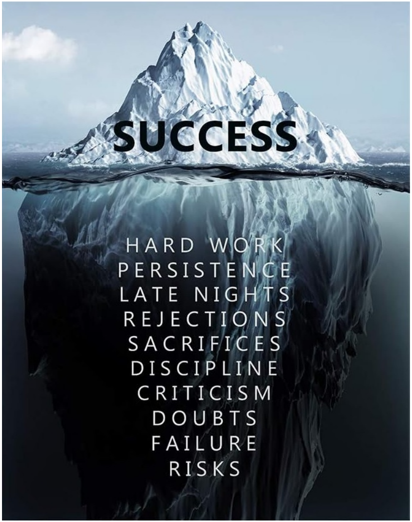
General Tips about Studying in a University

- To do well, ***inspiration*** is more important than ***perspiration***.
- Hard work does not necessarily guarantee success, but no success is possible without ***hard work***

  ⇒

  - Don't be too satisfied just by the fact that you work hard.
  - Make sure you work hard both on ***mastering "ground stuffs"*** and, more importantly, on ***staying on top of what's being taught***.
  - Go ***beyond*** lectures (e.g., CodingBat, LeetCode).
  - Be *curious* about why things work the way they do.
  - Always *reflect* yourself on *how things are connected*.
  - Be *happy* about doing work **not** associated with marks ☺

General Tips about Success



SOURCE: `https://a.co/d/aQ13fR1`

Professional Engineers: Code of Ethics

- *Code of Ethics* is a basic guide for ***professional conduct*** and imposes duties on practitioners, with respect to **society**, **employers**, **clients**, **colleagues** (including employees and subordinates), the **engineering profession** and him or herself.
- It is the duty of a practitioner to act at all times with,
  1. ***fairness*** and ***loyalty*** to the practitioner's associates, employers, clients, subordinates and employees;
  2. ***fidelity*** (i.e., dedication, faithfulness) to public needs;
  3. devotion to ***high ideals*** of personal honour and professional integrity;
  4. ***knowledge*** of developments in the area of professional engineering relevant to any services that are undertaken; and
  5. ***competence*** in the performance of any professional engineering services that are undertaken.
- Consequence of misconduct?
  - ***suspension*** or ***termination*** of professional licenses
  - civil ***law suits***

Source: PEO's Code of Ethics

What is this course about?

- ***Data Structure*** [WHAT]

  Systematic way of organizing and accessing data
  e.g., arrays, linked-lists, stacks, queues, maps, trees, graphs, *etc.*

- ***Algorithm*** [HOW]

  Step-by-step procedure, using the appropriate data structure(s), for solving a computational problem
  e.g., inserting, deleting, sorting, searching

- ***Analysis*** [HOW GOOD?]

  Determining, mathematically, the **correctness** and **efficiency** of algorithms

Example (1): A Searching Problem

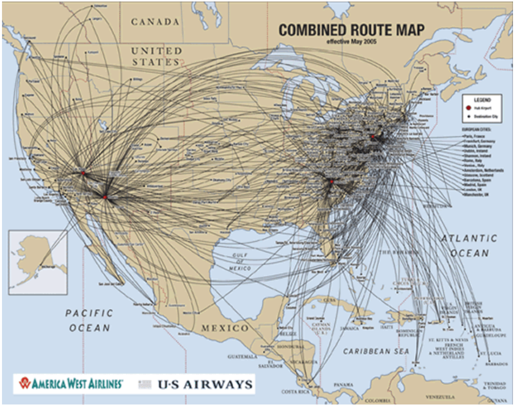
Problem: How would you save the records of a megacity with ***10 million residents***? Given a particular resident's social insurance number (ID), how ***fast*** can you locate his/her record?

```
ResidentRecord find(int sin) {
  for(int i = 0; i < database.length; i ++) {
    if(database[i].sin == sin) {
      return database[i];
    }
  }
}
```

- How many times will you have to run the loop?
  Best case? [1]
  Worst case? [10 million]
- You will learn about the appropriate data structure and algorithm to solve this problem (i.e., ***searching***), in the ***worst case***, within **24 iterations** of a loop!

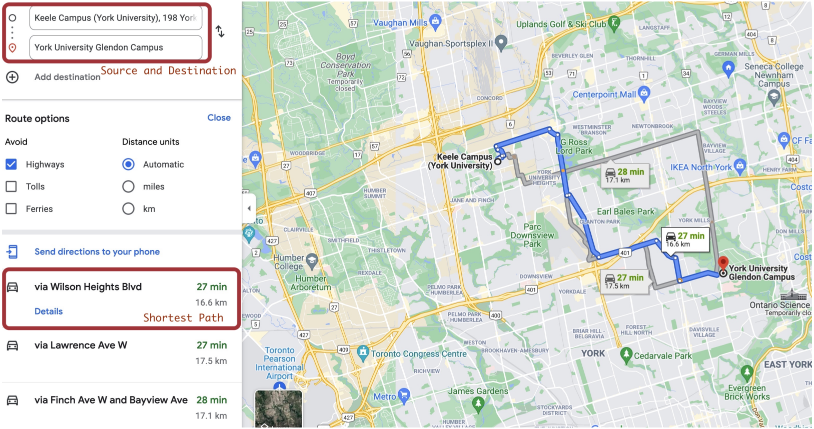
Example (2a): Flight Routing

Problem: Given the point-to-point connections of several airline companies, how do you plan an ***itinerary*** of flying from one city (origin) to another (destination)?



Example (2b): Car Routing

Problem: Plan a driving route which takes the ***minimum*** amount of time to arrive.



○ Keele Campus (York University), 198 York

York University Glendon Campus

Source and Destination

⊕ Add destination

Route options — Close

Avoid

☑ Highways

□ Tolls

□ Ferries

Distance units

● Automatic

○ miles

○ km

Send directions to your phone

via Wilson Heights Blvd — 27 min — 16.6 km

Details

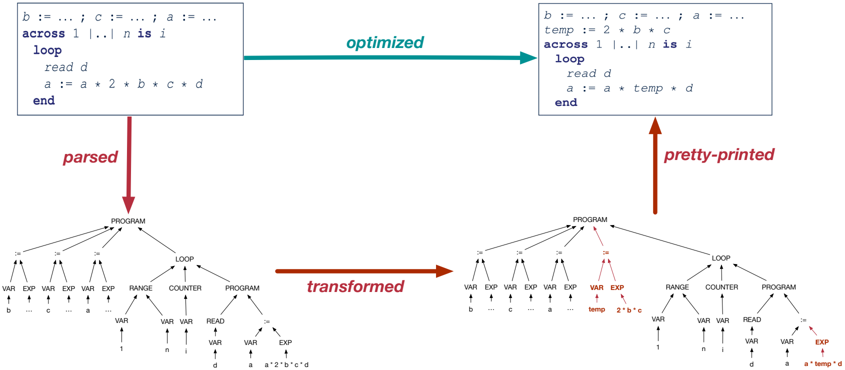
Shortest Path

via Lawrence Ave W — 27 min — 17.5 km

via Finch Ave W and Bayview Ave — 28 min — 17.1 km

Example (3a): Program Optimization

Problem: Given a user-written program, ***optimize*** it for best runtime performance.

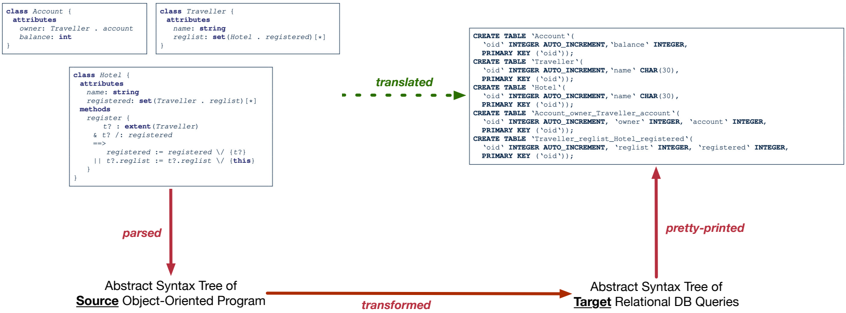


```
b := ... ; c := ... ; a := ...
across 1 |..| n is i
  loop
    read d
    a := a * 2 * b * c * d
  end
```

```
b := ... ; c := ... ; a := ...
temp := 2 * b * c
across 1 |..| n is i
  loop
    read d
    a := a * temp * d
  end
```

Example (3b): Program Translation

Problem: Given a user-written object-oriented program, ***translate*** it into SQL tables/queries for persistent storage in a relational database.



```
class Account {
 attributes
   owner: Traveller . account
   balance: int
}
```

```
class Traveller {
 attributes
   name: string
   reglist: set(Hotel . registered)[*]
}
```

```
class Hotel {
 attributes
   name: string
   registered: set(Traveller . reglist)[*]
 methods
   register {
        t? : extent(Traveller)
     & t? /: registered
     ==>
         registered := registered \/ {t?}
     || t?.reglist := t?.reglist \/ {this}
   }
}
```

```
CREATE TABLE `Account`(
  `oid` INTEGER AUTO_INCREMENT,`balance` INTEGER,
  PRIMARY KEY (`oid`));
CREATE TABLE `Traveller`(
  `oid` INTEGER AUTO_INCREMENT,`name` CHAR(30),
  PRIMARY KEY (`oid`));
CREATE TABLE `Hotel`(
  `oid` INTEGER AUTO_INCREMENT,`name` CHAR(30),
  PRIMARY KEY (`oid`));
CREATE TABLE `Account_owner_Traveller_account`(
  `oid` INTEGER AUTO_INCREMENT, `owner` INTEGER, `account` INTEGER,
  PRIMARY KEY (`oid`));
CREATE TABLE `Traveller_reglist_Hotel_registered`(
  `oid` INTEGER AUTO_INCREMENT, `reglist` INTEGER, `registered` INTEGER,
  PRIMARY KEY (`oid`));
```

translated
parsed
pretty-printed
Abstract Syntax Tree of Source Object-Oriented Program
transformed
Abstract Syntax Tree of Target Relational DB Queries

Index (1)

Index (2)