Administrative Issues

EECS2011 X:
Fundamentals of Data Structures
Winter 2023

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!
- There will be a bonus opportunity for you to fill out an informal, anonymous ***midterm course survey*** during the reading week.
- 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 the study materials.
- Still complete assignments and tests (***no extension***).

Mask Policy

- As of now, the university does **not** require us to wear masks.
- However, for the safety of your instructor and classmates, please **do** consider:
  - Wearing a mask
  - ***Minimizing*** talking if you decide **not** to wear a mask
  - ***Minimizing*** the consumption of food or drink
- ***When you visit my office in-person, I'd be grateful if you can wear a mask.***

Class Protocol

- Talking
- Using mobile phones

  $\Longrightarrow$ ***distracting***, ***disrespectful*** to everyone
- If you cannot stop talking or using mobile, please ***leave to do it***.
- Slides are ***self-contained***, so I may not read them off.
- I will focus on core concepts, 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
```

Course Information

- A single eClass site:
  - ***LE/EECS2011 X - Fundamentals of Data Structures (Winter 2022-2023)***
    - Announcements
    - Assignments [instructions only]
    - Written Tests [instructions & submissions]
- Check your emails regularly!

Required Study Materials

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

  `https://www.eecs.yorku.ca/~jackie/teaching/lectures/index.html#EECS2011_W23`

- The *course syllabus* is posted in the above lectures site.

Course Syllabus

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

Need Accommodation?

- Please contact me via email as soon as possible, so we can make proper arrangements for you.
- We will work out a way for you to gain the most out of this course!

Becoming a Software Engineer

- One useful mindset is to treat this course as a training course for ***programming interviews***.
- How a real ***software developer*** works:
  - Programming ***problems*** are explained via the expected methods' ***API*** (input and output types) and some ***use cases***, without visualization!
  - A set of ***tests*** must be *re-run automatically* upon changes.
- Thinking *abstractly* without seeing changes on a physical device is an important skill to acquire before graduating.

  e.g., Watch *interviews at Google*: Given problems described in English, solve it on a whiteboard.
- Take advantage of the ***problems*** I assign in class.

Study Tips

I attempt 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.

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***

  $\Rightarrow$

  - 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*.

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

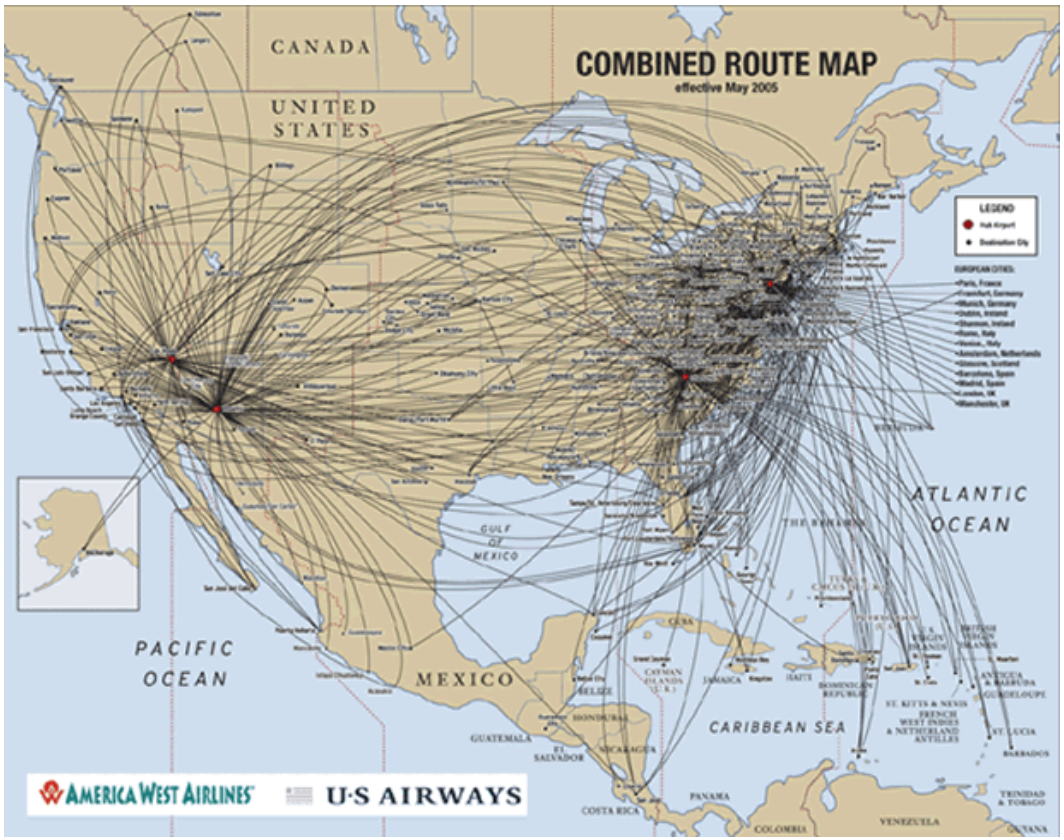
Problem: How would you save the records of a megacity with ***10 million residents***? Given a particular resident's social insurance number (ID), how ***fast*** can you locate his/her record?

```
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 the loop!

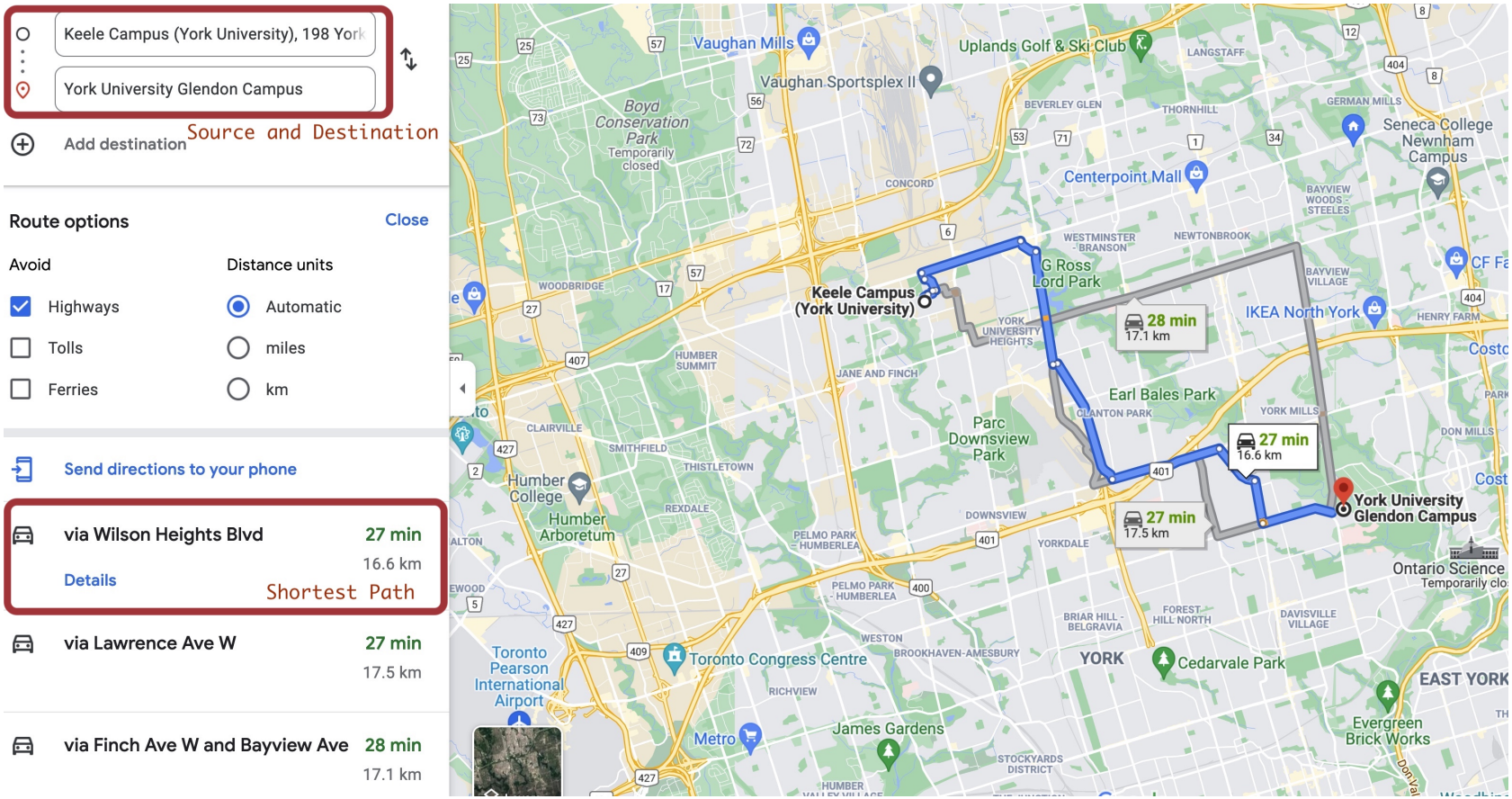
Example (2a): Flight Routing

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)?



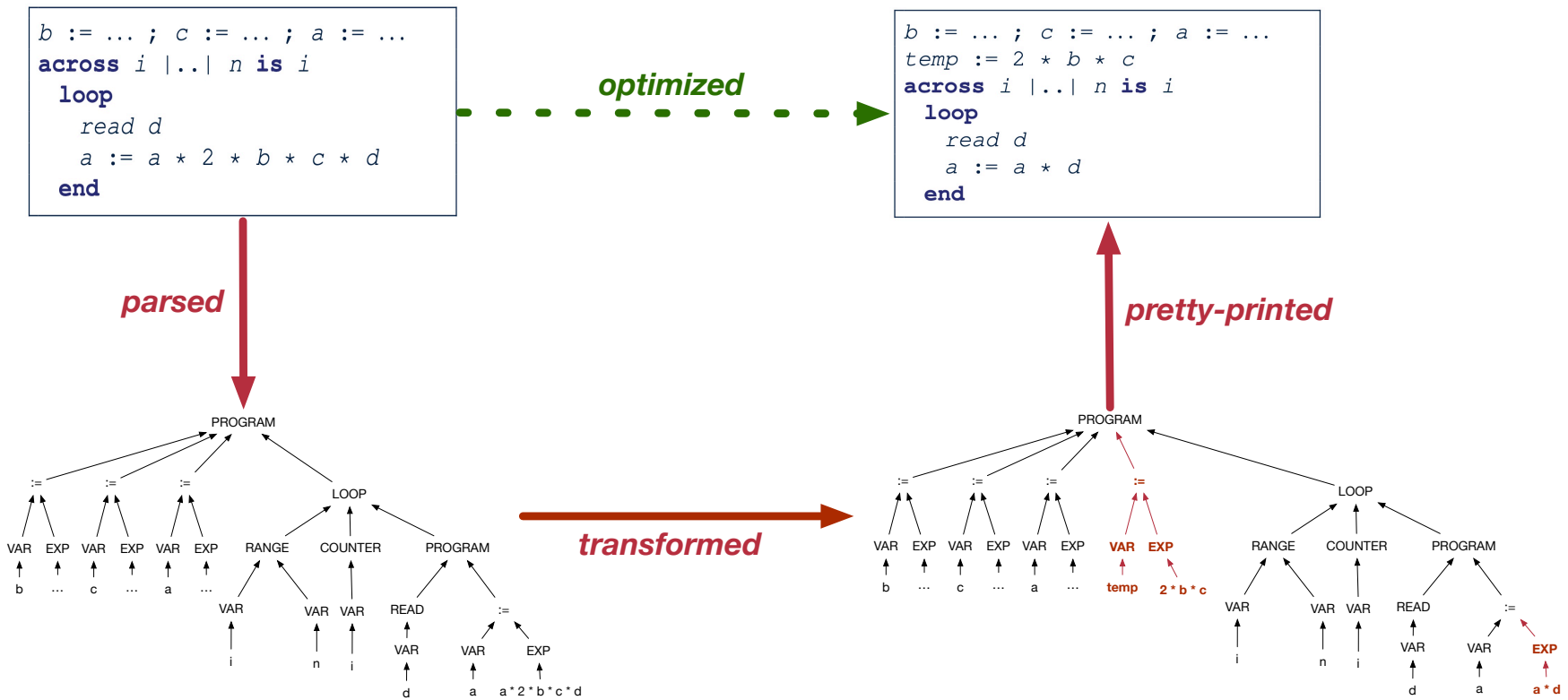
Example (2b): Car Routing

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



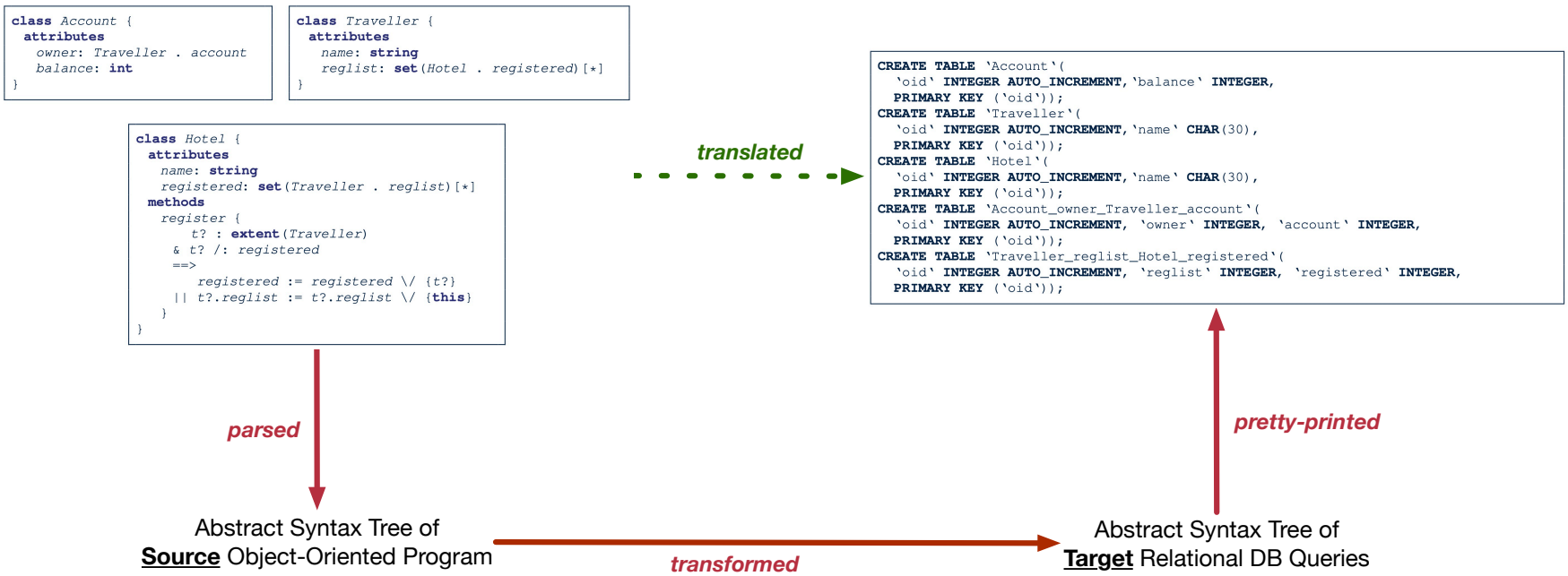
Example (3a): Program Optimization

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



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.



Index (1)

Index (2)