CPSC 259
# Data Structures and Algorithms for Electrical Engineers

Introduction

Administration

- Instructor: Geoffrey Tien, gctien@cs.ubc.ca
  Office: ICCS 245 (building inaccessible)
  Office hours (via Collaborate Ultra):
    Tuesday 09:30 – 11:00
    Wednesday 09:30 – 11:00

- Course website:
  - http://www.students.cs.ubc.ca/~cs-259/
  - Lecture slides, labs, homeworks can be downloaded here
  - Administrative policies can be found here also

- Other dynamic course content can be found on Piazza

- Thareja, Reema
  - Data Structures Using C, Second Edition
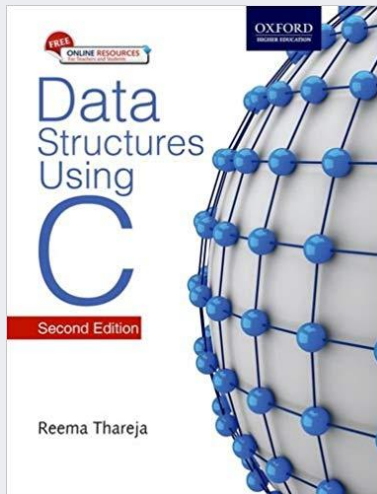
  - Very detailed code examples

- But any data structures book using C will be fine
- I will aim to make lecture materials a fairly complete resource

Piazza

- https://piazza.com/ubc.ca/winterterm12020/cpsc259/home

- The Piazza bulletin board is required reading. It will be used for important material that may not be mentioned in class

- Problems with the CPSC 259 course contents (e.g. lecture, lab, textbook, assignments) can be posted to Piazza

- Students are encouraged to ask questions, and to respond to other students' questions

- Only send e-mail if you have a private issue that you cannot post to Piazza

Lecture

- Lecture slides can be downloaded from the course website
  - 08:00-09:00 session hosted on Collaborate Ultra via Canvas
  - Sessions will be recorded and available in Collaborate Ultra recordings
  - Light use of iClicker (Cloud) (worth 2.5% of grade)
    - Download iClicker Reef app, or sign up to use web app
  - For students in alternate time zones, a separate session for answering iClicker questions will be run starting at 17:30 Vancouver time

Midterms

- Midterm 1
  - Tuesday, October 13
  - Evening – 18:00 – 19:30 (90 minutes)
  - Open-book, invigilated via Zoom

- Midterm 2
  - Tuesday, November 17
  - Evening – 18:00 – 19:30 (90 minutes)
  - Open-book, invigilated via Zoom

- You must notify me of extra-curricular conflicts two weeks ahead of each date (September 29 / November 03)!

Labs

- There are 5 labs (cycles of 2 weeks each), starting week of Sept.14
- Labs hosted on Zoom

- In-lab component:
  - During week 1's lab time
  - Completed in pairs, has deliverables that are due either during your week 1 lab time, or, for certain deliverables, no later than during week 2's lab time

- In-lab programming quiz:
  - During week 2's lab time, the first hour will be a programming test, done individually
  - Timed and invigilated via Zoom

Labs

- There are 5 labs (cycles of 2 weeks each)

- Take-home programming assignment:
  - During each 2-week cycle (minus a day) starting with week 1's lab time, a take-home programming assignment to be completed with your in-lab partner
  - E.g. if your lab time is on Monday, then your "two weeks" end just before midnight on Sunday (>13 days from start of week 1)
  - The last 50 minutes of lab time during week 2 will allow you and your partner to work on the take-home assignment. TAs will be available in the lab to help.

  - Submission of code via either GradeScope or PrairieLearn (pending setup – details will be given in Lab 1)

Written homeworks

- There will be 2 written homeworks to complete during the term
  - These can be done in pairs
  - Purpose is to give you exposure to non-programming problems like those that may appear on the exams

Evaluation

- In-lab component and deliverables: $5 \times 1.5\% = 7.5\%$

- In-lab programming quizzes: $5 \times 2.5\% = 12.5\%$

- Take-home programming assignments: $5 \times 3.5\% = 17.5\%$

- Written homeworks, $2 \times 2.5 = 5.0\%$

- Midterms: $2 \times 12.5\% = 25.0\%$

- Final exam: 30.0%

- iClicker questions: 2.5%

- To pass the course, you must obtain at least 50% overall by these weights, and you must pass the weighted average of the two midterms and final exam.

Prerequisites

- APSC 160

- If you do not have the prerequisite, you will be dropped from the course. Check with a CS advisor if you have a problem

- The instructor cannot change your lab section or add you to the course

  - Contact the department for enrollment issues

Programming environment

- Visual Studio will be used in labs
  - You may use another IDE of your choice, but you should check that your submitted code compiles and runs under Visual Studio

- Details for obtaining a free copy via Microsoft Imagine will be given in Lab 1.

GradeScope

- Written homeworks, and exams will be graded via GradeScope

- Access GradeScope via Canvas (pending setup)
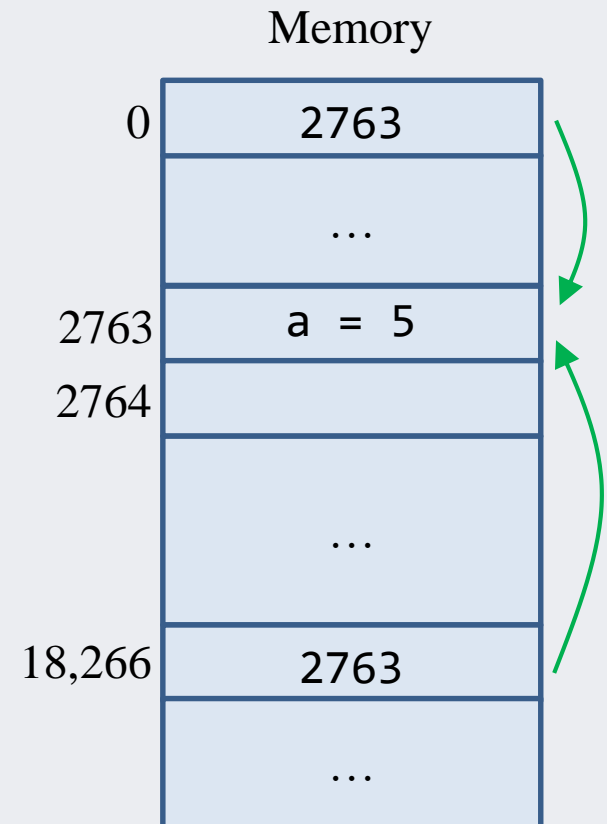
Course Overview

- Briefly review some key programming concepts from APSC 160 (functions, modularity, arrays)

- An object variable residing in memory contains an object, but a pointer specifies *where* an object is located (address)

- Pointers can refer to objects that are allocated on demand

- Pointers can be used for shared access to objects

- Pointers are used in many complex data structures

Memory

0	2763
	…
2763	a = 5
2764	
	…
18,266	2763
	…

Course Overview

- We have used primitive data types
  - `int`, `float`, `string`…
- What if our program requires more complex data types?
  - E.g. a car dealership needs to keep track of its vehicle inventory
    - Year
    - Make
    - Model
    - VIN
    - Colour
    - List of options

- We'll see how we can create our own data types

4 – Complexity

- Suppose I have three functions that all produce the same result (on some given input)

  – How can I compare these algorithms? (in a platform-independent manner?

  – Are there tradeoffs between the algorithms?

 – Based on a specific set of inputs, can I conclude that one algorithm is better than the other algorithms?

5 – Linked lists

- A linked list is a data structure for collecting a sequence of objects, such that addition and removal of elements in the middle of the sequence is efficient.
    - Selling (removing) a car from dealership inventory
    - Adding a new item into a sorted list

- This is not the case with arrays, assuming we keep the data compacted

- Linked lists are constructed using pointers

5 – Linked lists

- A data structure in which elements can be added to or deleted from anywhere

- Nodes are allocated as needed for each element (i.e. no need for extra unused space like arrays)

- Each node points to the location of the next node in the list, thus every node contains:

  – The data stored in the node, and

  – A pointer or link to the next node in the list

5 → 12 → 13 → 19 → 22 →

node
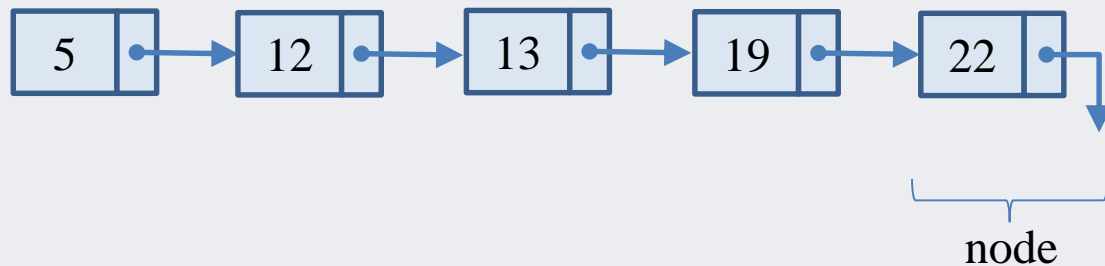
6 – Recursion

- Recursion is a powerful technique to break up complex computational problems into simpler ones.
    - "Recursion" refers to the same computation recurring, or occurring repeatedly, as the problem is solved

    - E.g. factorial
    - $5! = 5 \times 4 \times 3 \times 2 \times 1$
    - $4! = 4 \times 3 \times 2 \times 1$
    - $5! = 5 \times 4!$

- In APSC 160, we learned how to use an array to store some items and then search for them

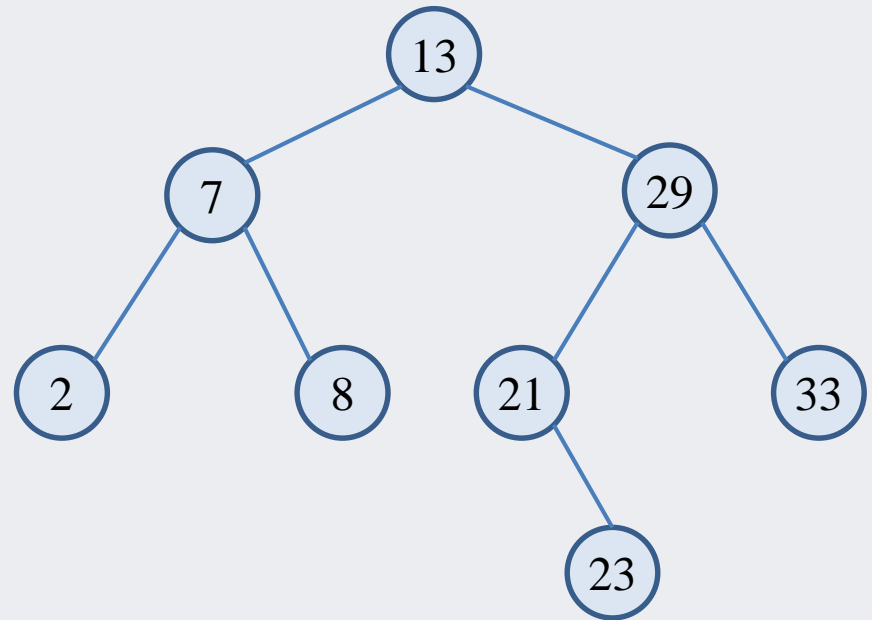| 12	34	89	18	21	–1	

- Using a different data structure, we can potentially find items more efficiently

- A binary tree is a data structure built from nodes (like linked lists)
  - Each node contains a data element, "left" pointer, and "right" pointer

- Advantages: (mostly) quick search, insert, and remove operations

- Disadvantage: complicated removal algorithm

Course Overview

- Suppose I have the following tasks to do for next week:

    6 – Respond to Piazza questions

    6 – Update course website

    8 – Release homework

    2 – Eat lunch

    9 – Make lecture slides

    1 – Sleep

    3 – Replace worn-out pants

    –1 – Bathe

- If the larger numbers represent greater urgency, what kind of data structure can we use to help us (efficiently) pick high priority tasks first?

9 – Sorting

- If I have a collection of values (in an array), how can I put them in order?
  - Is there one algorithm that works well for all the different arrays and permutations I can supply?
  - Study some well-known sorting algorithms
  - Analyze the complexity of these sorting algorithm and look at tradeoffs

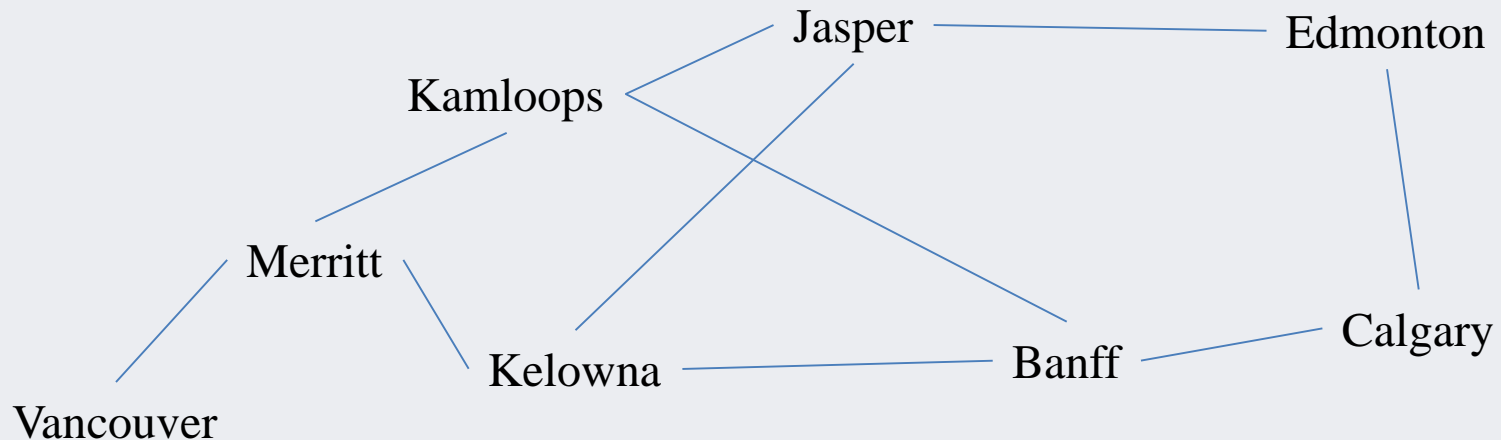- In APSC we learned how to use an array to store some items and then search for them

- Hashing still uses arrays, but uses a hash function to map keys (values) to integers
  - The converted integer is used to quickly find the right spot in the array for a given entry
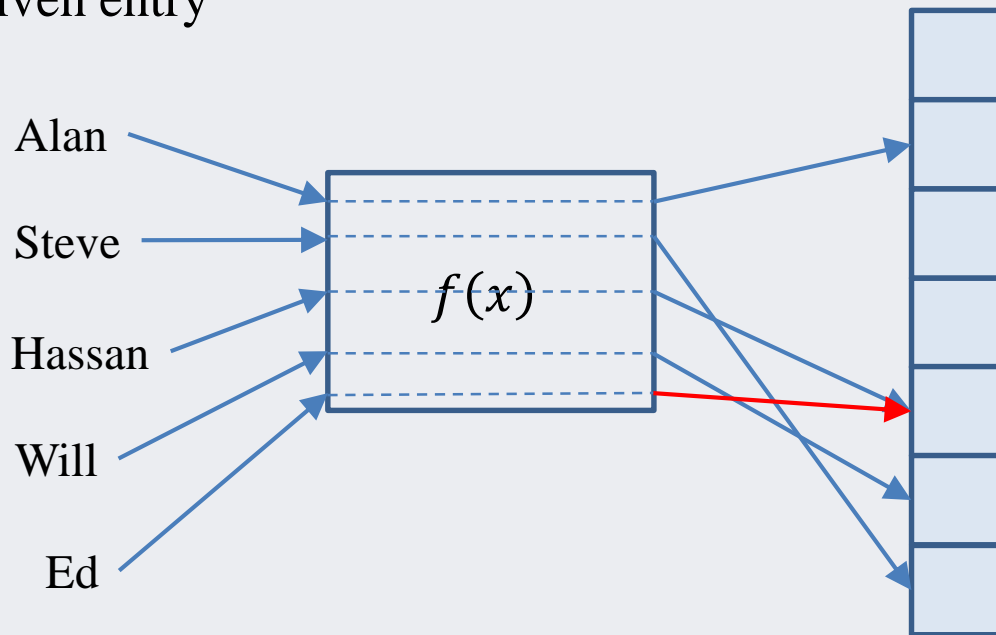
- Graph is a structure for representing network relations
- Many real-world problems can be shown by graphs

- What data structure is suitable for representing a graph (in a computer's memory)?
- What is the shortest path between two cities on a road/flight map?

That's all (for today)!

- Next class we start with APSC 160 review

- Homework:
  - Please ensure that your undergrad CS ID is activated!
  - https://www.cs.ubc.ca/getacct/