COMP251: Algorithms and Data Structures

Giulia Alberini, Jérôme Waldispühl
School of Computer Science
McGill University
About Me

- Jérôme Waldispühl
- Associate Professor of Computer Science
- Bioinformatics, Human-Computer Interactions & Video games!
- How to reach me?
  - Office hours (TBA; See online schedule)
  - By appointment (email me to schedule a meeting)
  - Email: cs251@cs.mcgill.ca

(Note: This will be the only email address you should use and from which you can expect an answer)
About (2)

• Giulia Alberini
• Lecturer in Computer Science
• Cryptography and CS Education

• How to reach Giulia?
  o Office hours (TBA; See online schedule)
  o By appointment (email me to schedule a meeting)
  o Email: cs251@cs.mcgill.ca

(Note: This will be the only email address you should use and from which you can expect an answer)
Objectives

• Classical tree & graph algorithms
• Techniques to efficiently solve computational problems
• Estimate the efficiency of an algorithm
• **Prove the correctness of an algorithm**

**THIS IS NOT A PROGRAMMING CLASS!**

(But you will learn a LOT of techniques that will make you a better programmer)
Course Material

Course web page:
• Slides of lectures
• General information & rules
• Schedule
• Announces

MyCourses:
• Grades
• Video recording of the lectures
• Access to discussion forum (Ed)
Communication

General inquiries:
Use the forum (https://edstem.org/us/). The answer may be help to your peers too!

Private matters:
Email us at cs251@cs.mcgill.ca
• Both instructors receive the email simultaneously.
• If the question is a general request, we will ask you to post it on Ed to answer it publicly.
• If the question has been answered on the forum, we will redirect you there.
Evaluation Scheme

- 30% for 3 assignments (10% each)
- 20% for 1 mid-term exams
- 50% for the final exam

Notes:
- There will be no modification of this scheme
- The mid-term is **NOT** optional (as well as the final...
Schedule

• Classes start... Today and end on Dec 1.

• Assignments:

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Released</th>
<th>Due on</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sep. 20</td>
<td>Oct. 6</td>
</tr>
<tr>
<td>2</td>
<td>Oct. 18</td>
<td>Nov. 8</td>
</tr>
<tr>
<td>3</td>
<td>Nov. 15</td>
<td>Nov. 29</td>
</tr>
</tbody>
</table>

Assignments are released 2 weeks before the due date. This is much more that needed. It will allow you arrange you schedule. Start early!

• Midterm: November 1 (in-person during regular class hours)

• Final: Exam week
Lecture

• Lectures are recorded and available for streaming but **cannot be downloaded**.

• When possible, the slides will be available on the course webpage shortly before the lecture.

• Questions are encouraged!
Office hours

• Check schedule on the course webpage for the instructors and TA’s office hours (OHs).
• By defaults OHs are in-person, but occasionally we will also organize Zoom sessions.
• The instructors will give priority to questions about the course material and administration.
• Question about assignments should be ask in priority to Tas
• How to reach us? cs251@cs.mcgill.ca
Pandemic

• Wearing a mask is no longer mandatory but still recommended

• Back to the normal mode (i.e., pre-pandemic), yet if the situation should change, we will adjust.
Outline

• Sep 6 - 15: Background & COMP 250 Review
• Sep 20 - 29: Dictionaries (Tree ADT & Hash tables)
• Oct 4 - Oct 6: Intro to Algorithm design (Greedy algorithm)
• Oct 11 – Oct 27: Graph algorithms
• Nov 3 - Nov 24: Algorithm design & Algorithm analysis
• Nov 29 - Dec 1: Advanced topics (subject to change)
Prerequisites

COMP 250:
• Data structures
• Recursive algorithms

MATH 240:
• Graph theory
• Combinatorial methods
• Basic proof techniques

What will useful too?
• Basic understanding of probabilities
Textbooks

(Available as E-book at the McGill library)


Textbooks are recommended but not mandatory.
Assignments

• Mostly proofs and theoretical problems
• Relatively short, but start as early as possible
• Read carefully and strictly follow the formatting guidelines.
• Submitted on MyCourses using Crowdmark
• Discuss but do not share/copy solutions (this is plagiarism).
• Write down the name(s) of person(s) with whom you discussed the answers (including teaching staff).
• We cannot guarantee to answer any question sent or posted less than 24h from the deadline (but we will try to).
• 20% late submission penalty if less than 24 after the deadline. Refused otherwise. This is a strict policy.
Additional Material

• We will release programming assignments that will help you to practice the implementation of the algorithms covered in class

• Optional and Not graded

• You can visit the TA to check your solution
Academic integrity

- If we identify a case of plagiarism, we will report it directly to a disciplinary officer and send email notifications after.
- For all other rules and processes, you can consult [https://www.mcgill.ca/deanofstudents/students/student-rights-responsibilities/code](https://www.mcgill.ca/deanofstudents/students/student-rights-responsibilities/code)
Midterm

• What? Quizzes, application of algorithms, and **proofs**.

• When? During the regular class hours.
  ▪ Designed to be fully completed in 1h15
  ▪ You are not expected to have any conflict with another midterm or class
  ▪ November 1

• Where? In our regular classroom.
Final Examination

• What? Same format as the mid-term but it covers all topics (including advanced topics!)
  ▪ Same format as the mid-term but it covers all topics (including advanced topics!)

• When?
  ▪ Designed to be fully completed in 3h00
  ▪ Exam week

• Where?
  ▪ In-Person!
Next (four) classes

• Review of COMP 250 material
  • Reccurences
  • Proofs
  • Big Oh notations
  • Trees and graphs

• Basic probability (expectation, indicator)
• Binary numbers
Prerequisite from COMP 250: Data Structures

- **Array**
  - Running time for insert, delete, find...

- **Single-linked list**
  - Better than arrays:
    - Easier to insert and delete
    - No need to know size in advance
  - Worse than arrays:
    - Finding the n-th element is slow (so binarySearch is hard)
    - Require more memory (for the "next" member)

- **Doubly-linked list**
  - Allow to move backward
  - Makes deleting elements easier

- **Stacks and queues**
  - You should understand all applications we saw
Recursions

• Definition (recursive case & base case)
• Binary search
• Fibonacci
• Merge Sort
• How to write a function describing the running time of a recursive algorithms.
• Estimate the number of recursive calls.
• Dividing original problem into roughly equal size subproblems usually gives better running times.
Running time and big-Oh

• Running time:
  o Counting primitive operations
  o Dealing with loops: $\sum_{i=1}^{n} i = n(n+1)/2$ is $O(n^2)$
  o Worst-case vs average-case vs best-case

• Big-Oh notation:
  o Mathematical definition
  o Big-Oh is relevant only for large inputs. For small inputs, big-Oh may be irrelevant (remember integer multiplications)

• Big-Theta, Big-Omega
• Unless mentioned otherwise, big-Oh running time is for worst-case.
• You need to know and understand the big-Oh running time of all algorithms seen in class and in homeworks.
ADT (Abstract Data Structure)

What it is?

Description of the *interface* of a data structure. It specifies:

- What type of data can be stored
- What kind of operations can be performed
- Hides the details of implementation

Why it is important?

Simplifies the way we think of large programs
Trees

- treeNode representation
- Vocabulary: node, leaf, root, parent, sibling, descendants, ancestors, subtree rooted at x, internal and external nodes, ordered, binary, proper binary
- Depth and height
  - Definition
  - How to compute it.
- Tree traversal
  - Pre-order, In-order, Post-order
Dictionary ADTs

- Stores pairs (key, info)
- Operations: find(key), insert(key, info), remove(key)
- Cases where array implementation is bad (with complexity)
- Cases where linked-list implementation is bad (with complexity)
Dictionary ADTs with Binary Search trees (BST)

• Property: for any node x,
  – keys in the left subtree of x have keys smaller or equal to key(x) and
  – keys in the right subtree of x have keys larger or equal to key(x)

• Algorithm to find a key and its running time $O(h) = O(\log n)$ if the tree is balanced.

• Inserting a new key. Running time $O(h)$. Sequence of insertion that can lead to bad running times.

• Removing a key.

• You need to be able to execute these algorithms by hand on examples.
Dictionary ADTs with Hash Tables

• Implements a dictionary
• Idea:
  – map keys to buckets
  – Each bucket is itself a dictionary
• Hash functions:
  – Goal: minimize collisions
  – Easy to compute
• Best case:
  – keys are distributed uniformly among the buckets. Each bucket contains few keys
• Worst case:
  – All keys end-up in the same bucket
Priority queues

• Heap property:
  – key(x) is smaller or equal to the keys of children of x.
  – All h-1 first levels are full, and in the last level, nodes are packed to the left

• Operations:
  – findMin(). Algorithm. O(1)
  – insert(key). Bubbling-up. O(log n)
  – removeMin(). Bubbling-down. O(log n)

• Array representation of heaps

• HeapSort
  – insert keys one by one
  – removeMin() one by one
Graphs

• All the terminology
• Data structures for representing graphs:
  – Adjacency-list
  – Adjacency-matrix
  – Running time of basic operations with each data structure
• Graph traversal
  – Depth-first search
    • Recursive
    • Iterative using a stack
  – Breadth-first search
    • Iterative using a queue
• IMPORTANT:
  Applications of DFS and BFS