COMP251: Algorithms and Data Structures

Jérôme Waldispühl, Roman Sarrazin-Gendron
School of Computer Science
McGill University

About Me

- Jérôme Waldispühl
- Associate Professor of Computer Science
- Research in Bioinformatics & Human-Computing
- 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 Me (2)

- Roman Sarrazin-Gendron
- PhD student in Computer Science
- Research in Bioinformatics & Human-Computing
- 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)

Where to get announcements & updates?

Official channel:

- Course web page: http://www.cs.mcgill.ca/~jeromew/comp251.html

Other channels

- MyCourses
- Ed: https://edstem.org/us/ (Login to stem via mycourses)
- Emails

Remote learning

- Lectures with Zoom
- Discussion board on Ed
- Assignments with codepost
- Exam grading using crowdmark
- We welcome your feedback to make the best use of these tools and improve, when possible, your overall experience.

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.

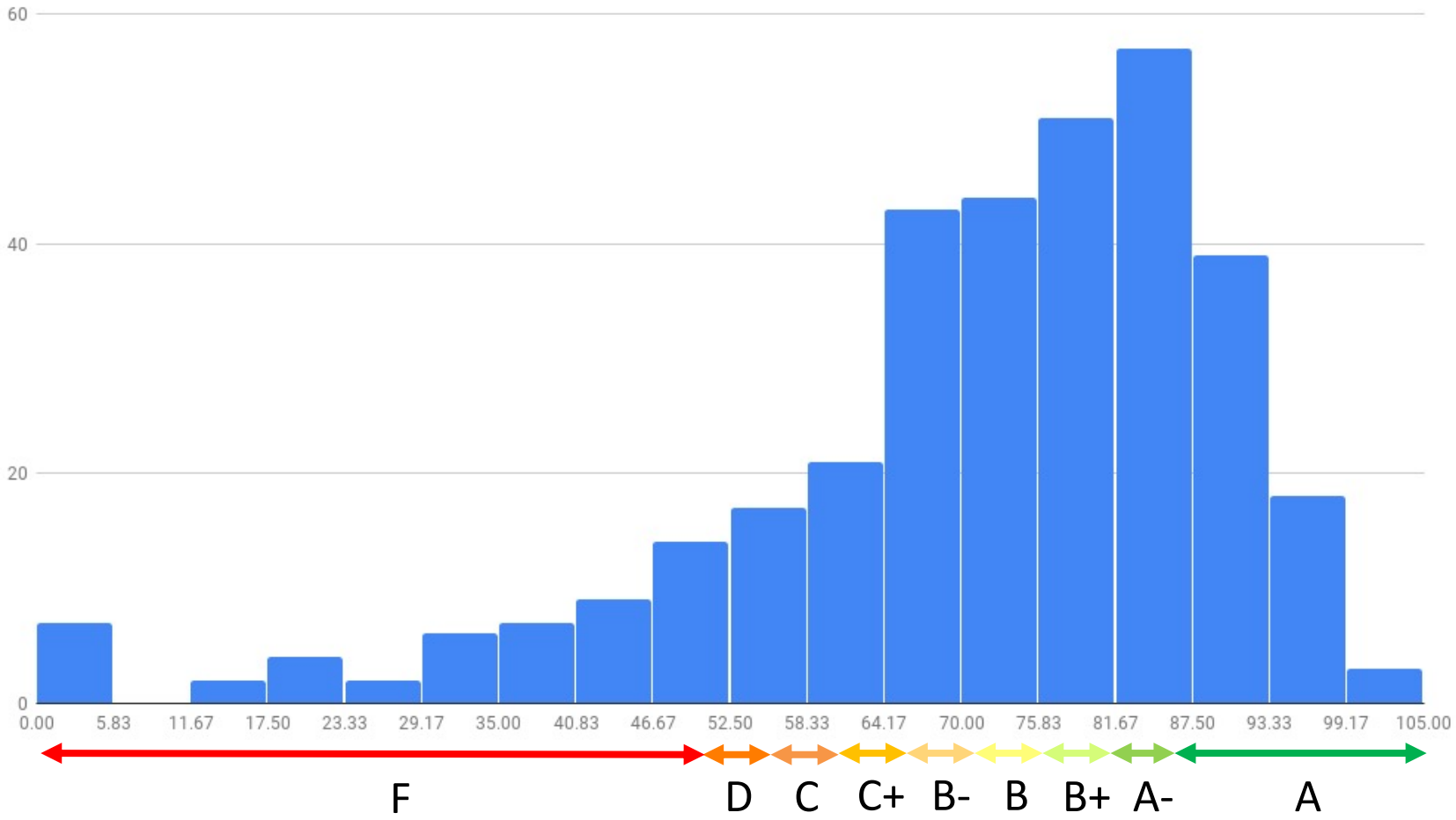
Evaluation Scheme

- 20% for 4 programming assignments (5% each)
- 20% for 1 mid-term exams
- 60% for the final exam

Notes:

- There will be no modification of this scheme
- The mid-term is **NOT** optional (as well as the final…)

Grading


60
40
20
0
0.00
5.83
11.67
17.50
23.33
29.17
35.00
40.83
46.67
52.50
58.33
64.17
70.00
75.83
81.67
87.50
93.33
99.17
105.00
F
D
C
C+
B-
B
B+
A-
A

Schedule

- Classes start… Today and end on Dec 2.
- Everything is... Online (except the final exam).
- Tutorials (Unix & Java): September
- Assignments:
  - End of September, mid October, beginning and end of November
  - released 2 weeks before the due date. Start early!
- Midterm: End of October (during regular class hours)
- Final: Exam week **(In person!)**

Online classes

- Lectures are recorded and available for streaming but cannot be downloaded.
- Questions can be asked in the chat. Raise (virtually) the hand and type your question.
- Slides will be available on the course webpage.
- We will try with the live format (with recording). If it does not work well, we will switch to pre-recording and live Q&A (a.k.a. flipped classroom).

Office hours

- Check schedule on the course webpage for the instructors and TA's office hours.
- The instructors will cover questions about the course material and administration.
- Programming questions (i.e. assignments) can be asked to TAs, but do not expect them to debug your code.
- We will organize Java tutorials.
- How to reach us? cs251@cs.mcgill.ca

Outline

- Sep 7 - 14: Background & COMP 250 Review
- Sep 16 - 30: Dictionaries (Tree ADT & Hash tables)
- Oct 5 - Oct 7: Intro to Algorithm design (Greedy algorithm)
- Oct 12 - Nov 2: Graph algorithms
- Nov 4 - Nov 18: Algorithm design & Algorithm analysis
- Nov 23 - 25: Advanced topics
- Nov 30 - Dec 1: Review

Note: Subject to changes

COMP251 vs COMP250

Topics:

- Algorithm design
- **Proofs** (correctness)
- Analysis of performance

Requirements:

- Basic understanding of probabilities and discrete math
- Programming in Java

Textboooks

[CLRS2009] Cormen, Leiserson, Rivest, & Stein, *Introduction to Algorithms*.
(*Available as E-book at the McGill library*)

[KT2006] Kleinberg & Tardos, *Algorithm Design*.

Textbooks are recommended but not mandatory.

Assignments

- Programming questions (+ optional proof not graded)
- Programming Language: Java
- Read **carefully** the formatting guidelines.
- **Strictly** follow the formatting guidelines.
- We will use Codepost (subject to change)
- You can replace your files before the deadline.
- Discuss but **do not share/copy solutions** (this is plagiarism).
- Indicate the name of persons with whom you discussed with (including teaching staff).
- We cannot guarantee to answer any question sent less than 24h from the deadline.
- We do not debug your code.
- 20% late submission penalty if less than 24 after the deadline. Refused otherwise. This is a strict policy.

Rules for programming questions

- Indent and comment your code!
- Use the template provided.
- Do **not** use custom libraries (unless specified).
- Follow the syntax of the command line provided in the question.
- Use the tests but recall it does not guarantee that your program is 100% correct.
- As safety net, check that your files compile on SOCS servers (Note: Create an account if you do not already have one)

Automated grading

"His assignments were not incredibly difficult, but you would get a 0 for tiny, unimportant details. I misnamed a file by one letter, automatic zero. After spending 3 weeks on the assignment, they made no exceptions, had no mercy or understanding."

(Anonymous comment on ratemyprofessor.com)

1. *Warning: Assignments were not including proofs*
2. *We (staff) only can decide what is important or not*
3. *In 2 weeks you have plenty of time to check the guidelines (a.k.a. do not wait the last minute)*
4. *No exception is the only way to be fair with everyone*

Note: we also get nice comments sometimes

Plagiarism

1. We run programs to automatically detect possible cases of plagiarism in programming assignments.
2. We manually review each case.
3. If we consider there is a case of plagiarism, we report directly to a disciplinary officer. We send email notifications after.
4. The rest of the procedure is out of our hands. Still, you can consult a description of the disciplinary procedure at: https://www.mcgill.ca/students/srr/honest/staff/student

Midterm

- What? Quizzes, application of algorithms, and **proofs**.
- When? During the regular class hours.
  - Designed to be fully completed in less than 1h30
  - You are not expected to have any conflict with another midterm
  - Dates will be announced ASAP
- Where? Crowdmark

Note: we will also closely monitor for plagiarism cases

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:
  - Counting primitive operations
  - Dealing with loops: $\Sigma^{n}_{i=1}\ i\ = n\ (n+1)/2$ is $O(n^2)$
  - Worst-case vs average-case vs best-case
- Big-Oh notation:
  - Mathematical definition
  - 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