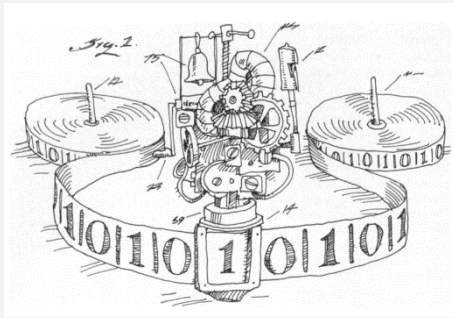
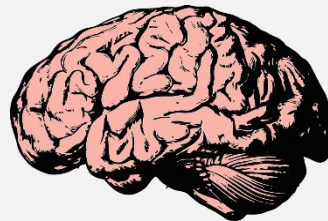
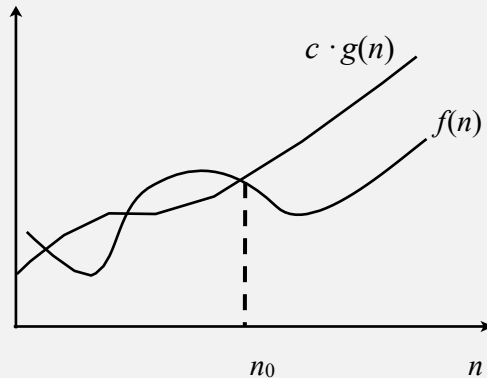


COMP 3602

Intro to Theory of Computation

Lecture 1



Course info

- Lectures MWF 4:00pm-4:50pm in EN 2040
- Communication:
 - Main page: D2L
 - Subscribe to notifications to get announcements to your email
 - Email: kol@mun.ca
 - Please try not to send me large attachments



Marking scheme



- In-class quizzes/tests
 - Based on posted practice problems
 - 3 best out of 4, at 25% each?
- Auto-graded multi-attempt drills?
 - On D2L, 5 drills at 1% each
- Presentaton at 15%?
reading group
- Seminar/reading group attendance?
 - 1% each, up to 5%?
- Oral exam instead of one of the quizzes?
1st written, 2nd oral
- Bonus bug bounty. *2nd IP*

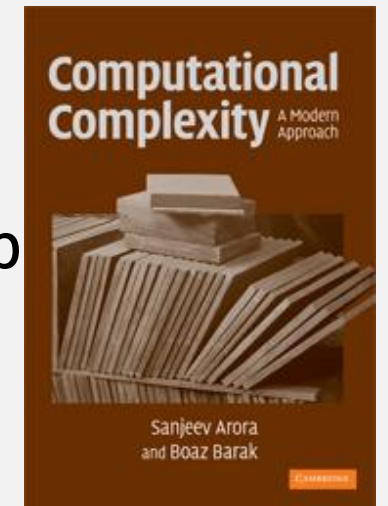
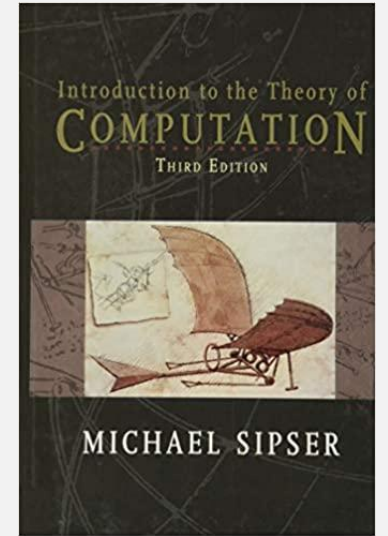
All evaluation except
drills is in person!

Logistics

- Office hours times: before/after lectures?
 - Reading group times? *10-11 MWF?*
1-2 MT?
- I plan to be away for two workshops during the semester:
 - Jan 11-16 (next week)
 - Apr 3-10 (last week of classes)

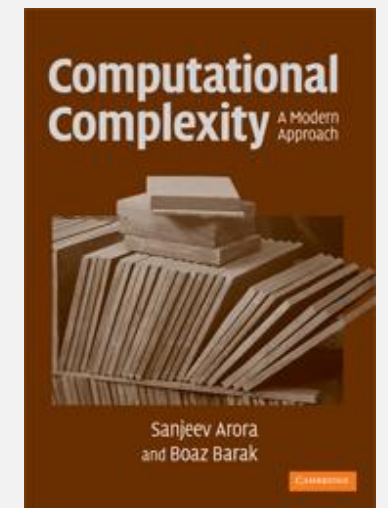
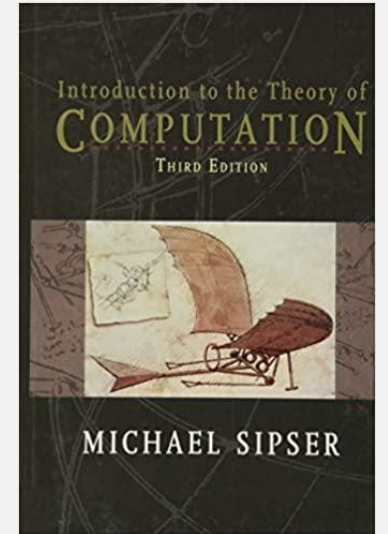
Notes/textbooks

- No official textbook
 - I will post slides
- Reference books:
 - Sipser “Intro to Theory of Computation”
 - Arora and Barak “Computational complexity: A modern ap



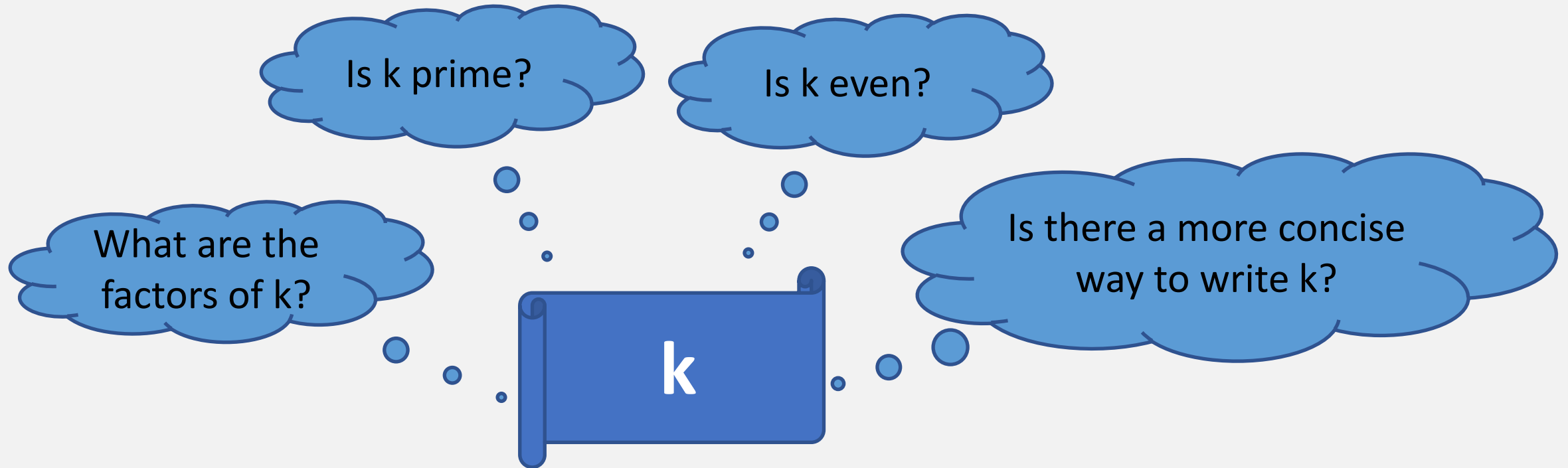
Topics

- Basic: what have you seen already?
 - Finite automata / regular languages
 - Context-free grammars / pushdown automata
 - Turing machines
 - Undecidability
 - NP-completeness
- Advanced: suggestions?
 - Recursion theory / Kolmogorov complexity / Gödel's incompleteness?
 - Circuits complexity / space complexity/ proof complexity/ randomness / barriers / interactive proofs?..
- Advanced topics for your presentations?



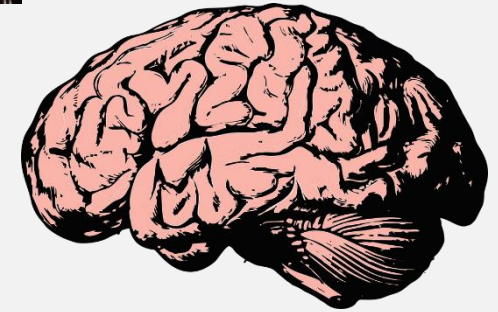
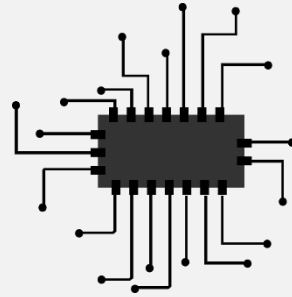
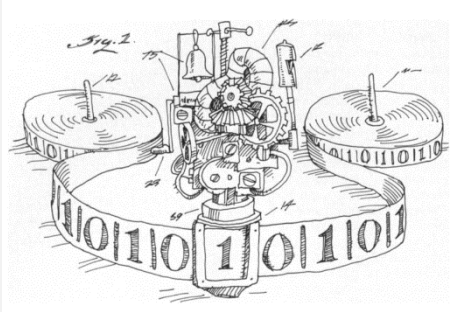
Questions/comments?

How hard is it to solve each of these problems?



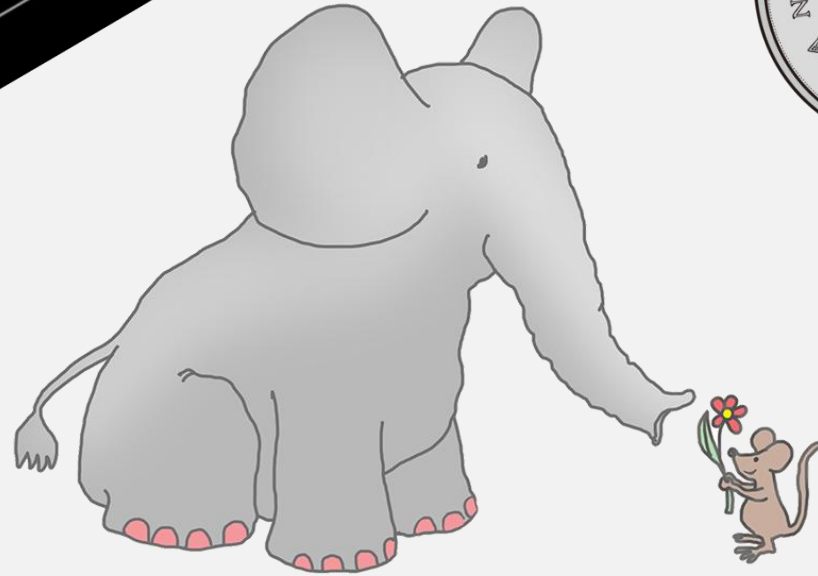
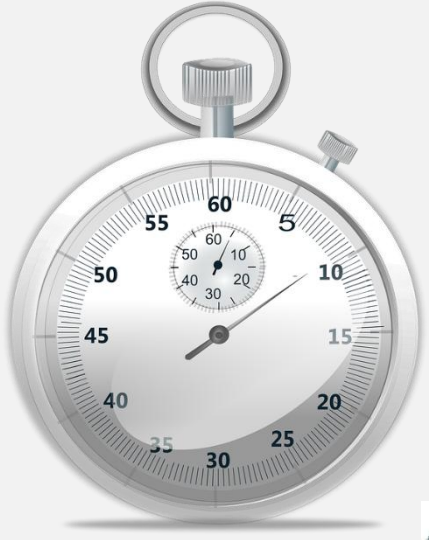
- Here, $k \in \mathbb{Z}$

Hard for whom?



We will look at different computational models, and see what they can do and how much resources they need.

Hard in what way?



We will consider several types of resources, though mostly focus on time.



How hard is it... ?

- For now, consider a “normal pseudocode” for a computer with random-access memory, and let’s look at time as the main resource.
 - Time: number of elementary operations
 - We will give more precise definition later.
 - Hardness may depend on how the input is represented



How hard is it... ?

- First, is the problem solvable at all?
 - If not, prove it!
- If it is solvable, try to give upper and lower bounds on the time needed to solve it as a function of input length.
 - For upper bounds, give an algorithm: use O -notation
 - For lower bounds, give a proof: use Ω –notation
 - If have matching upper and lower bounds, use Θ -notation.

Computability

Complexity

We want to know the complexity of the **problem**, not just one specific algorithm!