COMP-250 - Introduction to Computer Science

Doina Precup
Artificial Intelligence / Machine Learning
http://www.cs.mcgill.ca/~dprecup/courses/comp250.html
Lectures / assignments / solutions
My Courses - bulletin board, assign. submit / grades
Lectures are recorded
45% assignments (5)
5% quizzes
In-class midterm - 15% - Wed before March break
Final - 35%
Office hours - Mon & Fri 10:30-12, MC 111N

1) "Basics"
- Algorithms
- How to measure alg. complexity
- Min, Sorting ...
- <u>Prove</u> complexity, <u>correct</u>

2) "Programming" - Java
Variable, loops, conditionals
Inheritance, objects, pre-definable classes

3) Data structures
- Stacks, Queues Lists
- Trees of different kinds
- Graphs

Bonus: Special topics

Algorithms

Recipe 250 g flour, 100 ml milk] inputs

simple Mix in a bowl until smooth

Pancakes!] output

Simple algorithm!

Eg. 2 Adding fractions

$$\frac{a}{b} + \frac{c}{d} = \frac{ad + cb}{bd}$$ Math

Fraction = pair of integers (a, b)

Integers → basic "type"

Fraction → Object

Pseudocode: Intermediate between program and our idea in your head

→ easy to translate in code

→ specify the logic without all details

Algorithm: Add Fractions

Input: Two fractions (pairs of ints)
$(a, b), (c, d)$

Output: Fraction: (x, y)

$x = ad + bc$

$y = bd$

Return (x, y)

constants-size input, output no. of operations

Generic → Variables

$O(1)$

~~Natural Language~~

Pseudocode → Programming Languages (Formalized) → Machine Language

01101100...

(COMP-273)

Properties of algorithms

- Space complexity
- Time complexity

] as a function of size of input

- Correctness: Input → "Desired" output
  - Testing: Think up input examples → try out
    - Debug
    - "Typical" example
    - → "Corner" cases: array sorted
  - Prove (mathematically) alg is correct
    - Conditions that input satisfies
    - that output should satisfy
    - (possible in simple algs)
- "Simplicity" → others would use it/understand
  - → less chance of mistakes
  - ↓ Comments

Basic programming constructs for pseudocode

- Assignments max ← a[i]
- Conditionals if a[i] > max then
  - . . . .
  - else . . . (optional)
- Loops: for (while, repeat ... until)

Finding the maximum in an array

a[n]: 1 15 2 30 9 (n=5)

↑ i

max "candidate"

max ← ~~1~~ a[1] (1...n) ←

(0...n-1) Java

"For each" element ≈ n operation ⇒ "O(n)"

for i=2 to n

if a[i] > max

max ← a[i]

i ← i+1

Return max

nr. of operations depends on size of the input

O() → depends on n

How to measure how complicated an alg is

1) Time it takes to execute

≠ Actual clock-time of execution

⇒ Number of operations needed (O() Big-Oh)

2) Memory - quite important