CMPT 120
Intro to CS & Programming I

WEEK 1 (Jan. 6-10)

Lecture 1: What is an algorithm?
(and how to make lunch for your kid brother)

http://www.sfu.ca/~jlumbros/Courses/CMPT120/

Quick Check

How many of you have an iClicker?

A iClicker 1

B iClicker 2

C Another device

(iClicker GO seems to be difficult to operate… will keep you posted)

About the Paper Survey

Thank you so much! I read them all (4 hours): very interesting read!

- Why did you choose this course?

  Credits; requirement; recommended by friends; someone in my entourage has a CS career, "accident"

- What is your experience with computers?

  "My computer always gets hurt because of my puppy"

- How would you describe programming?

  I don't know; "Changing or customizing programs on a computer"; "Programming = computer 'whispering'"; "**Love it!**"; "Programming = 0001111000011010 ???"; "Afraid of it"; "Always reminds me of high IQ"; "**Sore eyeballs […] Temper tantrums**"

(Results on the "Your expectations?" will be used later…)

For a Quicker Answer

- Contact me personally for "personal" stuff
  jeremie.lumbroso@sfu.ca
- Contact the help address (read by me + TAs) for "general" questions:
  cmpt-120-help@sfu.ca
- **Please begin subject line with CMPT 120**
- Course website (in construction!)
  http://www.sfu.ca/~jlumbros/Courses/CMPT120/

How to think about problems a computer can solve

INTRO: ALGORITHMS?

Important Distinction

- Algorithmic: how to think about/deconstruct problems in terms appropriate for computers

- Programming language: to express *algorithms* in a way understandable by computers

You may not know what these things are, but they will be introduced right now and in the next lectures

What is an "Algorithm"?

First definition

- An algorithm is a set of instructions that can be used to systematically solve a problem
- Concept is fundamental to CS and prog.

Second definition

Algorithms explain, without going into the details of a programming language, "how to break down problems in chunks that that can be understood by a computer?" (or a very "dumb" person)

Origin of word "algorithm"

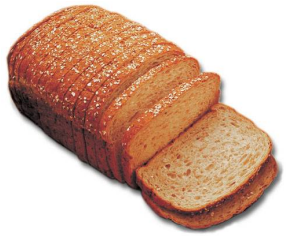
- From "Muḥammad ibn Mūsā al-Khwārizmī"
- Persian mathematician 780-850

What else did he do?

- Introduced Indian numerals (and zero) and decimal system to Western world
- Also introduced "algebra" (comes from Arab word "al-jabr" involved in solving quadratic equations)
- More info on Wikipedia: http://goo.gl/Xyvam7

Sandwich Example

borrowed from Diana Cukierman



How to prepare a Peanut Butter & Jelly sandwich?

Do it yourself 

- Take a piece of paper out
- Write a set of instructions explaining to a very young child how to make a P&J sandwich

When you are done, use iClicker:

A Too easy/dumb! I hate peanut butter! Bad instructor!

B Learning exceptional skills here! You are brilliant!!

Problems that (should) arise

- What do I start with?

  loaf of bread? slices of bread? what kind of bread (and does it matter)? what utensils do I need? is there a difference between ingredients and utensils?

- What do I end with?

  does the sandwich need to be a special way? how many sandwiches?

- Additional requirements

  if this is for a restaurant, additional hygiene requirements; any dietary considerations (gluten-free?, peanut-free?!!)

Observations

- Precision
  - need to define **every** step
  - need to know level of detail and language to use
- Input/output
  - need to know what our inputs are
  - need to know, be clear what the output is (expected output if steps followed clearly)
- Systematic instructions
  - result should always be the same
  - alternatives, repetition, set of instructions should be reusable
  - big picture first and then details

How to make a P&J sandwich

- Version 1
  1. put peanut butter on top of bread
  2. put jam on top of peanut butter

- Version 2
  1. clean hands, put gloves on
  2. get a slice of bread
  3. put peanut butter on top of bread slice
  4. then put jam on top

P&J: a Third Version

1. clean hands, put gloves on
2. prepare a board to work on
3. if package is closed open package of bread
4. take a slice of bread from package and put on board
5. if peanut butter jar is not opened
6. firmly hold lid of peanut butter jar with right hand
7. firmly hold peanut butter jar with left hand
8. turn right hand holding lid clockwise until jar opened
9. take a knife
10. introduce knife in peanut butter jar
11. spread peanut butter on top of slice a couple times until covering the slice of bread with a 0.25cm thick layer
12. if jam jar is not opened,
13. ….

Version 4

- **Parameter:** *N* (the number of sandwiches)
- **Input:** sliced bread (*2N* slices per sandwich), jar of peanut butter, jar of jam, two knives, plate
- **Output:** *N* P&J sandwich

Make sandwich (*N*)

1. prepare instruments on table (2) # 2 = nb. of knives needed
2. do *N* times
    1. get a slice of bread
    2. spread peanut butter on slice
    3. spread jam on top of peanut butter
    4. put on plate # sandwich is ready
3. end

Sub-procedures 1

Prepare instruments on table (*K*)

1. put board on table
2. put *K* knives on table
3. put jars on table

Get one slice of bread

1. if package is closed
    1. open package
2. take one slice of bread from package
3. place slice on board

Sub-procedures 2

Spread (XXX)

1. if XXX jar is closed
    1. open jar (XXX)
2. take a knife
3. introduce knife in XXX jar
4. …

Open jar (XXX)

1. firmly hold lid of jar XXX with right hand
2. firmly hold XXX jar with left hand
3. turn right hand holding lid clockwise until jar is opened

Reusable "Blocks" of Instructions

- We have created a set of functions:
  - Make sandwich (*N*)
  - Prepare instruments on table (*K*)
  - Get one slice of bread
  - Spread (XXX)
  - Open jar (XXX)

- Such "blocks" will later be called functions
- Some "blocks" have parameters (in parenthesis)
- **Important:** not "tied to" P&J sandwiches, can be used for anything else

Wake-up!

How do you feel about this material?

A I understand, and you are going too slow

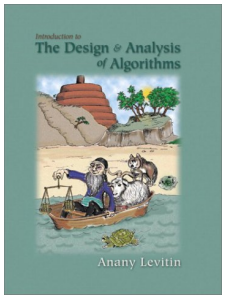
B I understand, and you are going just right

C A bit confused, but I think rereading the slides later will help

D I do not understand this concept at all!!! Help!!!

E "*This talk of food is making me nauseous!*"
or "*I am allergic to peanuts, you insensitive idiot!*"

Formal Definition of an Algorithm



DEFINITION (Levitin 2003, p. 3)

"*An algorithm is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.*"

Sounds complicated?

It "simply" summarizes what we just saw!!

Formal Definition of an Algorithm

"*An algorithm is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.*"

What notion seems most complicated?

- **A** Sequence
- **B** Unambiguous instructions
- **C** Required output
- **D** Legitimate input
- **E** Finite amount of time

"Sequence"

- Algorithms give a sequence of instructions
- Notion of "coming one after the other"
- Numbering of instructions

Unambiguous Instructions

- When you read an algorithm there should be no question about what should be done

Examples of ambiguity

- "Heat oil up in a pan" (supposes prior knowledge of the person doing it)
- "Look at the man with binoculars" (linguistic ambiguity)
- "Then take a left" (GPS ambiguity)
- "Set *k* to the remainder of *m* divided by *n*"
  (mathematical ambiguity, because Euclidean division of negative integers is not well defined)
- …

Solving Problem/Required Output

- What we **promise** the result will be if the person or computer follows instructions (= guaranteed)
- The output must **always be the same**
- With "*peanut butter jar, jam jar, slices of bread…*" algorithm will provide "*sandwich*"

Counter-examples

- "*Sandwich*" is not "*sandwich everybody who eats will like*" or "*gluten-free sandwich*", etc.
- *If you are getting assaulted*, algorithm: "*Scream for help*"; output: "*Someone will come*" cannot be promised!

Legitimate Input

- An algorithm might require some input (or parameters) and there might be constraints
- For example:
  - our "algorithm" Make sandwich (*N*)
  - *N* is the number of sandwiches
    - *could N be a negative number?*
    - *could N be a fraction?* ***in real life? for our "algorithm"?***
    - *could N be square root of 2?*
    - *could N be the lyrics to the song "Oh Happy Day!"*

Finite Amount of Time

- The algorithm should **stop** (and give its output) in a **finite** amount of steps

- Finite ≠ infinite, takes forever

- Example: "**write out 1/3 in decimal form**"
- Output: 0.3333333333333333… forever

Recap

- A "recipe" is an example of an algorithm
- Algorithmic is how to break down problems in a way that can be understood by a computer
- Requires absolute precision of instructions
- Takes input/parameters and must work with different input/parameters (different number *N* of sandwiches, different type of jelly)

Pacing and Understanding

How did you feel about this lecture?

A. Too slow

B. Just fine

C. Too fast, but I can catch up of my own

D. Too fast, and I need you to slow down

E. I think this is hopeless, I am leaving this class

Small Assignment

- For next lecture, describe instructions for **one** of these:
  - how to draw a circle?
  - how to draw a simple image? (Pictionary)

  - how to make tea? (teabag? bulk tea? teapot?)
  - how to take a bath? how to do the dishes?
- I will pick up two next class and read them