Algorithms

CSC131 The Beauty & Joy of Computing

10 September 2018

- an algorithm is a "recipe"
- an algorithm is a sequence of logical and arithmetic operations
- an algorithm can be expressed in some combination of English, mathematical notation, and pseudo-code
- pseudo-code looks like a programming language but we are not so fussy about conforming to exact rules of a programming language's grammar
- a computer program is an algorithm translated into some particular programming language
- there are many, many programming languages!
- Church-Turing Thesis:
  - every computer is equivalent to every other computer
  - every programming language is equivalent to every other programming language
  - "equivalent" means that if I can write a program on computer A that produces output O when given input I, then I can write a program on computer B that also produces output O when given input I
- what can a computer do?
  - arithmetic: addition, multiplication, subtraction, division
  - logic: and, or, exclusive or, not
  - compare: less than, greater than, equals (¡, ¡=, ==, !=, ¿, ¿=)
  - branch (jump): conditionally execute an instruction
- kinds of programming languages
  - machine language: numerical codes corresponding to elementary states
  - assembly language: short abbreviations (letters) corresponding to machine instructions

  - high-level language: looks as much like English and mathematics as possible

- programs must be translated before they can be run on the computer
  - translate high-level language to assembly language with a compiler
  - translate assembly langage to machine language with an assembler
  - compiler or assembler translates and also checks for conformity with language's grammar rules

- some important algorithms
  - searching
  - sorting

- other algorithms
  - hidden surface removal
  - scheduling
  - for us—machine learning

- principal resources that are available to programmers
  - time
  - space
  - (and now, energy)

- methods of solving problems
  - we solve many problems by guessing, evaluating the guess, and then guessing again, until we have an answer that is "good enough"

- simple example of getting an answer by making successively better estimates—Newton's Method
  - guess that the square root of 2 is 1
  - next guess is the average of 1 and 2/1 (call it nextBestGuess)
  - next guess is the average of nextBestGuess and 2/nextBestGuess
  - keep going until the difference 2 - nextBestGuess * nextBestGuess is very small

- complexity
  - care about number of instructions that computer must execute as a function of the size of the input

- P =? NP: there appear to be 2 kinds of problems
  - those that can be solved in a practical amount of time
  - those that cannot be solved in practical amount of time