Notes, 3(a)

ECE 606

What is an algorithm?

A step-by-step procedure to solve a problem.

1. What is a "procedure?" What is a "step?"
2. What is a "problem?" What does it mean to "solve a problem?"

Question (2) first:

For this course, and more broadly, in many cases for which we devise algorithms:

- "Problem" = "function" (the mathematical notion)
- "Solve a problem" = "Compute a function"
- E.g., what is an algorithm that, given input an array of distinct integers, outputs the maximum?

  Same as: an algorithm to compute the function $f : \mathcal{A}_{\mathbb{Z}} \to \mathbb{Z}$, where $\mathcal{A}_{\mathbb{Z}}$ is the set of all arrays of distinct integers, and $f : [z_1, \ldots, z_n] \mapsto \max_{i \in \{1,\ldots,n\}} z_i$
- Special cases of interest to us:
  - "Decision problem": co-domain of function is $\{\mathsf{true}, \mathsf{false}\}$, or $\{0, 1\}$.
  - "Optimization problem" : biggest, smallest, longest, shortest,...

Question (1): "procedure," "step" in a procedure.

Recall our syntax for pseudo-code from Lecture 2(a).

Each such sub-routine is a procedure. We hypothesize a computer that can run such pseudo-code one "instruction" at a time. Examples of instructions:

- assign a value to a variable
- add two integers
- Compare values in two variables
- return an output

Each such "instruction" is a step.

Important note: each such "instruction" does not necessarily take the same amount of time to run. We clarify this in Lecture 3(c) on how we measure the time an algorithm takes to run.

E.g.,comparing two three-digit integers ~~will not take as long as~~ *may take longer than* comparing two ~~ten~~ *two* -digit integers.

(I've changed this to "two" from "ten" because the original version was indeed correct with "ten," and I mistakenly read "two" here in the lecture video.)

Properties of interest in an algorithm:

- Termination: given any (valid, finite) input, is the algorithm guaranteed to eventually stop running?
  - We will consider finite inputs only, and not worry about basic error-checking. Our focus is correct logic in the algorithm.
- Correctness: at the moment the algorithm terminates, if it indeed does, does it output what we expect it to, i.e., what is specified by the function it is intended to compute?
- Efficiency:
  - Time-efficiency: does the algorithm run sufficiently fast in time?
  - Space-efficiency: does the algorithm consume (i.e., allocate) sufficiently small amounts of space?