SA403 – Networks

Lesson 3: Algorithms

Notes

Book acknowledgment:

> "Introduction to Algorithms" by Cormen, Leiserson, Rivest, and Stein. *Third Edition*

Goals

- Introduce definition, purpose, and outline of algorithms

1 What is an Algorithm?

How would you define an algorithm?

Definition:

"Informally, an *algorithm* is any well-defined computational procedure that takes some value, or set of values, as ***input*** and produces some value or set of values as an ***output***. An algorithm is thus a sequence of computational steps that transform the input into the output."

What algorithms can you think of?

An algorithm can also be viewed as a tool for solving a well-specified computational problem. The statement of the problem specifies in general terms the desired input/output relationship. The algorithm describes a specific computational procedure for achieving that input/output relationship.

Example:

Assume that you need to sort a sequence of numbers into non-decreasing order. This is commonly referred to as the Sorting Problem.

What is the input for this problem?

What is the output for this problem?

In your own words, what is an algorithm?

What types of problems can be solved using algorithms?

Parts of an algorithm:

- Variables
  - What values can change?
- Parameters
  - What values must remain the same?
- Sequencing
  - In what order show tasks be performed?
- Conditionals
  - If X occurs, then Y must be performed.
    - *If*-statements
    - *While*-statements
    - *Do*-statements
- Repetition
  - When some set of tasks must be repeatedly performed.
- Subroutines
  - You can think of these as subtasks.

2 Try writing out an algorithm for solving the following problems...

Tying your shoes

Finding a minimum value among a set of numbers

Service Selection Assignments

3 Algorithm Running Time

The *running time* of an algorithm on a particular input is the number of primitive operations or “steps” executed. Assuming that each computer takes a constant amount of time to execute each line of code.

Worst-case Running Time:

The longest running time for *any* input of size n. This gives us an upper bound on the running time for any input. By knowing the worst-case running time, we can guarantee that the algorithm will never take any longer.

4 Written Steps

Determine the maximum among values x, y, and z.

1. If $x > y$ then proceed to step 2 Otherwise, proceed to step 3.
2. If $x > z$ then return x as the maximum value. Otherwise, return z as the maximum value.
3. If $y > z$ then return y as the maximum value. Otherwise, return z as the maximum value.

5 Pseudocode

You can think of pseudocode is a transition between writing out the steps and the code of an algorithm.

Algorithm 1 Determine maximum among values x, y, and z

```
if $x > y$ then
  if $x > z$ then
    Maximum value = $x$.
  else
    Maximum value = $z$.
  end if
else
  if $y > z$ then
    Maximum value = $y$.
  else
    Maximum value = $z$.
  end if
end if
```