Algorithms and Algorithm Analysis

The “fun” stuff
What is an algorithm?

A finite set of steps that specify a sequence of operations to be carried out in order to solve a specific problem.
Properties of Algorithms

1. Finiteness
2. Absence of Ambiguity
3. Definition of Sequence
4. Feasibility
5. Input
6. Output
What is Programming?

- Phases of Programming
  1. Design
  2. Implementation
  3. Testing
  4. Repeating
Algorithm vs. Programs

- A computer program is one concrete implementation of an algorithm using a particular computer language.
- The design phase should produce an algorithm.
- The implementation phase should produce a program.
- The design phase is typically much longer than the programming phase.
Algorithm Performance

- It would be great if you could code up an algorithm and then run it with a bunch of input.
- Take a look at the clock and decide how well it ran.
- There are many problems with this approach and it suffices to say that this is bad.
A better way

- What is often done is to approximate or estimate the performance of an algorithm
- Estimation is an important skill to learn and to use
- Question: How many hotdogs tall is the Empire State Building?
Example

- **Question:** How many hotdogs tall is the Empire State Building?
- **Answer:** The ESB is 1250 feet tall.
- **Assuming that a hotdog is 6 inches from end to end, you would need, 1250 * 2 = 2500 hotdogs.**
Complexity Analysis

- An objective way to evaluate the cost of an algorithm or code section.
- The cost is computed in terms of space or time, usually.
- The goal is to have a meaningful way to compare algorithms based on a common measure.
- Complexity analysis has two phases,
  - Algorithm analysis
  - Complexity analysis
Algorithm Analysis

- Algorithm analysis requires a set of rules to determine how operations are to be counted.
- There is no generally accepted set of rules for algorithm analysis.
- In some cases, an exact count of operations is desired; in other cases, a general approximation is sufficient.
- The rules presented that follow are typical of those intended to produce an exact count of operations.
Rules

1. We assume an arbitrary time unit.
2. Execution of one of the following operations takes time 1:
   1. assignment operation
   2. single I/O operations
   3. single Boolean operations, numeric comparisons
   4. single arithmetic operations
   5. function return
   6. array index operations, pointer dereferences
More Rules

3. Running time of a selection statement (if, switch) is the time for the condition evaluation + the maximum of the running times for the individual clauses in the selection.

4. Loop execution time is the sum, over the number of times the loop is executed, of the body time + time for the loop check and update operations, + time for the loop setup.
   † Always assume that the loop executes the maximum number of iterations possible

5. Running time of a function call is 1 for setup + the time for any parameter calculations + the time required for the execution of the function body.
Example

Sum = 0;
In >> Value;
while ( In )
{
    if ( Value < 0 )
    {
        Sum = -Sum;
        Sum = Sum + Value;
    } else {
        Sum = Sum + Value;
    }
    In >> Value;
}