Algorithm

- **Algorithm**: a procedure for solving a mathematical problem (as of finding the greatest common divisor) in a finite number of steps that frequently involves repetition of an operation; *broadly*: a step-by-step procedure for solving a problem, or accomplishing some end, especially by a computer.  [Merriam-Webster Dictionary]

- **Algorithm**: any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output. An algorithm is thus a sequence of computational steps that transform the input into the output.”  [Introduction to Algorithms, 2nd Edition by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein]
Algorithm

- An **algorithm** is a well-ordered collection of unambiguous and effectively computable operations that when executed produces a result and halts in a finite amount of time. [Schneider & Gersting. An Invitation to Computer Science, 1995]
Algorithms: Characteristics

• Well-Order, Step-by-step

• Unambiguous operations

• Effectively computable operations

• Input transformed into output

• Finite number of steps, Halts in a finite amount of time
Algorithm

- Recipes are examples of algorithms although they frequently lack the precision we require of algorithms (recipes include ambiguities)
1 cup steel-cut oats
3 cups water
Bring water to a boil in a saucepan, and stir in your oats.
Reduce heat to a simmer and cook oats until soft, 20 to 30 minutes, stirring occasionally. If all the liquid has reduced before your oats are tender, stir in a bit more water or milk and continue to cook.
Algorithms

oats = 1 cup
water = 3 cups
Pour water into saucepan
Place saucepan on burner
heat = high
temp = CheckTemp()
while temp is not Boiling
  temp = CheckTemp()
Pour oats into saucepan
  heat = heat – 1
  temp = CheckTemp()
  while temp is not Simmer
    heat = heat - 1
    temp = CheckTemp()
    while areNotSoft(oats)
      Stir
      if waterLevelTooLow()
        water = 1 oz
        pour water into saucepan
Algorithms

This “recipe” uses the basic building blocks of algorithms:

- Variables
- Sequencing
- Selection
- Iteration
Variables

- Oats
- Water
- Heat
- temp
Sequencing

• The order of the instructions matter

• The following order does something different

  Pour oats into saucepan
  Place saucepan on burner
  heat = high
  temp = CheckTemp()
  while temp is not Boiling
    temp = CheckTemp()
  Pour water into saucepan
Selection

- Choose to execute an instruction based in a condition
  
  if waterLevelTooLow()
  
  water = 1 oz
  
  pour water into saucepan
Iteration

- Repeat instructions based on a condition
  
  while temp is not Simmer
  
  heat = heat - 1
  
  temp = CheckTemp()
Methods and Functions

• Functions are not required but their use can simplify the development of algorithms.

• Think of a function as a way to name a group of instructions. The function can be given initial values (parameters) and return a value.

• `CheckTemp()`
  – returns a number

• `areNotSoft(oats)`
  – returns a boolean value
Algorithm Building Blocks

- Variables and expressions
- Instruction sequences
- Selection instructions
- Iterative instructions
- Functions
Variable

- Stores or hold a single value

- The meaning of single value depends on the data type of the variable
  - Int, float, string, list, ...

- Instructions can use the current value of a variable or change the value of a variable
Example use of numeric variables

• $x = 2$
• $x = 2*(3 + 7)$
• $x = x / y + 2 * z$
• $x = x + 1$

  – This statement might be confusing given your algebra background
  – There is a difference between assignment and equality
Sequencing
The following sequences result in two different program states
Order Matters!

\[
\begin{align*}
x &= 2 \\
y &= 10 \\
temp &= x \\
x &= y \\
y &= temp \\
\end{align*}
\]

\[
\begin{align*}
x &= 2 \\
y &= 10 \\
temp &= x \\
y &= temp \\
x &= y \\
\end{align*}
\]
Selection

if it is sunny then
  I will go for a walk
else
  I will wear sunscreen

if it is raining then
  I will bring an umbrella
Selection (if statements)

If $x > 10$ then

$y = x$

if $x > y$ then

$z = x$

else

$z = y$
Iteration (loops)

- sum = 0;
- i = 1;
- while i <= 10
  sum = sum + i;
  i = i + 1;
Sorting Example

• What characteristic must be defined for a data set so that it can be sorted?

•
Sorting

• Selection Sort
## Selection Sort Ascending Order

<table>
<thead>
<tr>
<th>Initial</th>
<th>Pass 1</th>
<th>Pass 2</th>
<th>Pass 3</th>
<th>Pass 4</th>
<th>Pass 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>14</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>7</td>
<td>14</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>14</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>14</td>
<td>20</td>
</tr>
</tbody>
</table>
Selection Sort

Suppose x is a list or array of n integers that can be indexed by position. Positions begin at 0 so there are elements in x at positions 0 through n -1

\[ p = 0 \]

Repeat the follow process \( n -1 \) times

- find the location of the smallest value in positions \( p \) through \( n -1 \)
- call the position of the smallest value \( s \)
- swap the values at position \( p \) and \( s \)
- \( p = p + 1 \)
Selection Sort

Let \(x\) be a list or array of \(n\) integers and let \(x[k]\) references the \(k\)-th integer in the array. Legal values for \(k\) are 0 through \(n-1\).

\[
\text{for (p = 0; p < n-1; p++)} \quad \{ \\
\quad \text{s = p;} \\
\quad \text{for (j = p+1; j < n; j++)} \quad \{ \\
\quad \quad \text{if (x[j] < x[s])} \\
\quad \quad \quad \text{s = j;} \\
\quad \} \\
\text{temp = x[p];} \\
\text{x[p] = x[s];} \\
\text{x[s] = temp;} \\
\} \\
\]