Algorithms

Problem Solving

- Problem solving is the process of transforming the description of a problem into the solution of that problem.
- We use our knowledge of the problem domain (requirements).
- We rely on our ability to select and use appropriate problem-solving strategies, techniques, and tools.

Algorithms

- An **algorithm** is a step by step solution to a problem.
- Why bother writing an algorithm?
  - For your own use in the future. You won't have to rethink the problem.
  - So others can use it, even if they know very little about the principles behind how the solution was derived.

Examples of Algorithms

- Washing machine instructions
- How to make a peanut butter and jelly sandwich
- A classic: finding the greatest common divisor (GCD) using Euclid's Algorithm

Washing Machine Instructions

- Separate clothes into white clothes and colored clothes.
- Add 1 cup of powdered laundry detergent to tub.
- For white clothes:
  - Set water temperature knob to HOT.
  - Place white laundry in tub.
- For colors:
  - Set water temperature knob to COLD.
  - Place colored laundry in tub.
- Close lid and press the start button.

Observations About the Washing Machine Instructions

- There are a finite number of steps.
- We are capable of doing each of the instructions.
- When we have followed all of the steps, the washing machine will wash the clothes and then will stop.

Refinement of our Algorithm Definition

- Old definition:
  - An algorithm is a step by step solution to a problem.
- Adding to the definition based on our observations:
  - An algorithm is a finite set of executable instructions which directs a terminating activity.

How to Make a Peanut Butter and Jelly Sandwich

- Open a web browser
- Go to `https://www.google.com`
- Search for "harvard how to make pbj sandwich"
- Click any video (or `https://youtu.be/okkIyWhNOiQ`)
- Enjoy

Final Version of the Algorithm Definition

- Our old definition:
  - An algorithm is a finite set of executable instructions that directs a terminating activity.
- Final version:
  - An algorithm is a finite set of unambiguous, executable instructions which directs a terminating activity.

History of Algorithms

- The study of algorithms began as a subject in mathematics.
- The search for algorithms was a significant activity of early mathematicians.
- Goal: To find a single set of instructions that can be used to solve any problem of a particular type (a **general solution**).

Euclid's Algorithm

Problem: Find the largest positive integer which divides evenly into two given positive integers (the **greatest common divisor, GCD**).

Algorithm:

1. Assign M and N the values of the larger and smaller of the two positive integers, respectively.
2. Divide M by N and call the remainder R.
3. If R is not zero, then assign M the value of N, assign N the value of R, and return to Step 2. Otherwise, the greatest common divisor is the value currently assigned to N.

Finding the GCD of 24 and 9

M	N	R
24	9	6
9	6	3
6	3	0

3 is the GCD of 24 and 9.

Euclid's Algorithm (Cont'd)

- Do we need to know the theory that Euclid used to come up with this algorithm in order to use it?
- What do we need in order to find the GCD using this algorithm?

The Idea Behind Algorithms

Once an algorithm behind a task has been discovered, we can just use it.

- We don't need to understand the principles.
- The task is reduced to following the instructions.
- The intelligence needed is "encoded in the algorithm."

Algorithm Representation

Syntax and Semantics

- **Syntax** refers to the representation itself.
- **Semantics** refers to the concept represented.

Contrasting Syntax and Semantics

- Human languages have both syntax and semantics.
- Syntax is the grammar of the language.
- Semantics is the meaning.
- Think about this sentence:
  I walked to the corner grocery store.
  - Is this sentence syntactically correct?
  - Is it semantically correct?

Contrasting Syntax and Semantics

- Human languages have both syntax and semantics.
- Syntax is the grammar of the language.
- Semantics is the meaning.
- Think about this sentence:
  I walked to the corner grocery store.
  - Is this sentence syntactically correct?
  - Is it semantically correct?
- How about:
  I talked to the funny grocery store.

Contrasting Syntax and Semantics

- Human languages have both syntax and semantics.
- Syntax is the grammar of the language.
- Semantics is the meaning.
- Think about this sentence:
  I walked to the corner grocery store.
  - Is this sentence syntactically correct?
  - Is it semantically correct?
- How about:
  I talked to the funny grocery store.
  I grocery store walker corner to.

Semantics (Cont'd)

- Conclusion: A sentence may be syntactically correct, yet semantically incorrect.
- This is also true of algorithms.
- And this is also true of computer code (often called a bug).

Problem Solving

- Decode this sentence: Pdeo eo pda yknnayp wjosan.
- We have just come up with a specific solution to a problem.
- Can this solution be generalized?

Problem Solving

- Decode this sentence: Pdeo eo pda yknnayp wjosan.
- We have just come up with a specific solution to a problem.
- Can this solution be generalized?
- Now that we know what algorithms are, we are going to try some problem solving and write algorithms for the problems.
- We'll start with step-by-step instructions that solve a particular problem and then write a generic algorithm that will solve any problem of that type.

Someone Stole a Cookie from the Cookie Jar

Problem: Momma had just filled the cookie jar when the 3 children went to bed. That night one child woke up, ate half of the cookies and went back to bed. Later, the second child woke up, ate half of the remaining cookies, and went back to bed. Still later, the third child woke up, ate half of the remaining cookies, leaving 3 cookies in the jar. How many cookies were in the jar to begin with?

Specific Solution to the Cookie Problem

First, we solve the specific problem to help us identify the steps.

- 3 cookies left X 2 = 6 cookies left before 3^{rd} child
- 6 X 2 = 12 cookies left before 2^{nd} child
- 12 X 2 = 24 = cookies left before 1^{st} child
  = original number of cookies

A Generic Algorithm

What is a **generic algorithm** for this problem?
An algorithm that will work with any number of remaining cookies

AND

that will work with any number of children.

A Generic Algorithm

What is a **generic algorithm** for a given problem?

An algorithm for solving the problem, that has been generalized to work with a variety of possible input parameters to produce input-specific solutions.

Generic Algorithm for Cookie Problem

- Get number of children.
- Get number of cookies remaining.
- While there are still children that have not raided the cookie jar, multiply the number of cookies by 2 and reduce the number of children by 1.
- Display the original number of cookies.

Flowchart for Cookie Problem

Pseudocode

- When we broke down the previous problem into steps, we expressed each step as an English phrase.
- We can think of this as writing **pseudocode** for the problem.
- Typically, pseudocode is a combination of phrases and formulas.

Pseudocode

- Pseudocode is used in
  - designing algorithms
  - communicating an algorithm to the customer
  - converting an algorithm to code (used by the programmer)
  - **debugging** logic (semantic) errors in a solution before coding (**hand tracing**)
- Let's write the Cookie Problem algorithm using a more formal pseudocode and being more precise.

Improved Pseudocode

```
Display "Enter the number of children:  "
Read <number of children>
Display "Enter the number of cookies remaining:  "
Read <cookies remaining>
<original cookies> = <cookies remaining>
While (<number of children>  >  0)
    <original cookies> = <original cookies> X 2
    <number of children> = <number of children> - 1
End_While
Display "Original number of cookies = ", <original cookies>
```

Observations

- Any **user prompts** should appear exactly as you wish the programmer to code them.
- The destination of any output data should be stated, such as in "Display", which implies the screen.
- Make the data items clear (e.g., surround them by $<$ and $>$) and give them descriptive names.
- Use formulas wherever possible for clarity and brevity.
- Use keywords (such as Read and While) and use them consistently. Accent them in some manner.

Observations

- Use indentation for clarity of logic.
- Avoid using code. Pseudocode should not be programming language-specific.
- Always keep in mind that you may not be the person translating your pseudocode into programming language code. It must, therefore, be unambiguous.
- You may make up your own pseudocode guidelines, but you MUST be consistent.

Brian's Shopping Trip

Problem: Brian bought a belt for $9 and a shirt that cost 4 times as much as the belt. He then had $10. How much money did Brian have before he bought the belt and shirt?

Specific Solution to Shopping Problem

$$
\begin{aligned}
Start\$ &= Belt\$ + Shirt\$ + \$10 \\
Start\$ &= Belt\$ + (4 X Belt\$) + \$10 \\
Start\$ &= 9 + (4X9) + 10 = \$55
\end{aligned}
$$

Generic Algorithm for Shopping Problem

- Now, let's write a generic algorithm to solve any problem of this type.
- What are the inputs to the algorithm?
  - the cost of the first item (doesn't matter that it's a belt): <item1 price>
  - the number to multiply the cost of the first item by to get the cost of the second item: <multiplier>
  - the amount of money left at the end of shopping: <amount left>

Generic Algorithm for Shopping Problem

- Now, let's write a generic algorithm to solve any problem of this type.
- What are the inputs to the algorithm?
  - the cost of the first item (doesn't matter that it's a belt): <item1 price>
  - the number to multiply the cost of the first item by to get the cost of the second item: <multiplier>
  - the amount of money left at the end of shopping: <amount left>
- What are the outputs from the algorithm?
  - the amount of money available at the start of the shopping trip: <start amount>
- Note that we may end up needing some intermediate variables.

Pseudocode

```
Display "Enter the price of the first item:  "
Read <item 1 price>
Display "Enter the multiplier:  "
Read <multiplier>
Display "Enter the amount left after shopping:  "
Read <amount left>
<item2 price> = <multiplier> X <item1 price>
<start amount> = <item1 price> + <item2 price> +
                              <amount left>
Display "The starting amount was ", <start amount>
```

Control Structures

Any problem can be solved using only three logical **control structures**:

- Sequence
- Selection
- Repetition

Sequence

- A series of steps or statements that are executed in the order they are written.
- Example:

```
Display "Enter two numbers:  "
Read <number1>
Read <number2>
<sum> = <number1> + <number2>
Display "sum = ", <sum>
```

Selection

- Defines one or more courses of action depending on the evaluation of a condition.
- Synonyms: **conditional**, **branching**, **decision**.
- Examples:

```
if (condition is true)
    do something
end_if
```

```
if (condition is true)
    do this
else
    do that
end_if
```

Selection

A more complex example:

```
DISPLAY "Enter number to invert"
READ <my_num>
If (<my_num> < 0)
    DISPLAY "Don't like negative numbers"
    if (<my_num> < -999)
        DISPLAY "... but I guess you really do!"
    End_if
Else_if (<my_num>  == 0)
    <result> = 0
Else
    <result> = 1 / <my_num>
End_if
```

Repetition

- Allows one or more statements to be repeated as long as a given condition is true.
- Synonyms: looping, iteration
- Example:

```
While (condition is true)
    do this
End_while
```

- Notice the repetition structure in the Cookie Problem pseudocode on Slide **??**.

Repetition

More complex example (with mistakes)

```
DISPLAY "Enter number to compute factorial for"
READ <my_num>

While (<my_num> > 0)
    <factorial> = <factorial> * <my_num>
    <my_num> = <my_num> - 1
End_while

DISPLAY "The factorial of", <my_num>, " is ", <factorial>
```

Repetition

More complex example (with mistakes)

```
DISPLAY "Enter number to compute factorial for"
READ <my_num>
<factorial> = 1

While (<my_num> > 0)
    <factorial> = <factorial> * <my_num>
    <my_num> = <my_num> - 1
End_while

DISPLAY "The factorial of", <my_num>, " is ", <factorial>
```

Repetition

More complex example (with mistakes)

```
DISPLAY "Enter number to compute factorial for"
READ <my_num>
<factorial> = 1
<saved_my_num> = <my_num>

While (<my_num> > 0)
    <factorial> = <factorial> * <my_num>
    <my_num> = <my_num> - 1
End_while

<my_num> = <saved_my_num>
DISPLAY "The factorial of", <my_num>, " is ", <factorial>
```

Writing Algorithms from Scratch

- Given a problem statement, we are going to write the corresponding generic algorithm for the solution.
- We will use the following procedure:
  - Determine the algorithm inputs and outputs
  - Complete the pseudocode

Warm-up: Simple Tip Calculator

Problem: Write an interactive program to calculate the dollar amount of tip on a restaurant bill at the standard 15% rate.

You should allow for changes in the total price of the bill.

Tip: Whenever getting user input, always check for invalid input.
Sometimes bad input is accidental, sometimes it's malicious behaviour.

Drawing a Rectangle

Problem: Write an interactive program that will draw a solid rectangle of asterisks (*) of user-specified dimensions. The program must also display the dimensions of the rectangle at the end.

General Tip Calculator

Problem: Write an interactive program to calculate a table of dollar amounts of tip on a restaurant bill. You should allow for changes in the total price of the bill. You should also ask the user for the range of tipping rates to calculate (ex: high or low amount, or 10%, 15%, or custom %).

Error checking should be done to be sure that the amount of the bill is greater than 0, that the tip amount is between zero and one.