Creating Algorithms

Use this resource to reflect on your practice or identify opportunities to integrate algorithms in your classroom. This resource was developed to support middle school science teachers to integrate computational thinking into NGSS-aligned lessons. The content can be adapted for other content areas, grade bands and contexts.

What is an algorithm?

A repeatable process that delivers an expected result.

Look fors:

Students may be creating algorithms when they are:

- Decomposing problems or tasks
- Identifying essential steps
- Testing and debugging
- Considering efficiency

Prompting questions:

Ask students to reflect on their process or progress with these prompting questions:

- What do you need to know to be able to solve this problem/do this task?
- What should be the result of the problem or task?
- Why is each step required to solve the problem/task?
- Would you include additional words or details to explain this process to a partner?
- Does each step have the result you intend it to?
- Does a partner testing your algorithm get the same results as you?
- Are there certain inputs where you do not get the intended result?
- Can your algorithm have the same result with less steps?
- Do you notice any patterns in your procedure?

Activities & Tools

<table>
<thead>
<tr>
<th>Activities that may integrate algorithms into science</th>
<th>You might use this when...</th>
<th>Template</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sort and categorize relationships between objects or processes</td>
<td>Students are identifying distinguishing qualities about items in order to classify them</td>
<td>Template: Creating algorithms to identify, sort, and categorize</td>
<td>Example: Creating algorithms to identify plants</td>
</tr>
<tr>
<td>Develop protocols to troubleshoot a question or design problem</td>
<td>Students have knowledge about a concept, process or tool and want to help others troubleshoot on their own</td>
<td>Template: Creating algorithms to troubleshoot a design problem</td>
<td>Example: Creating algorithms to troubleshoot a circuit</td>
</tr>
<tr>
<td>Code a computational tool to illustrate a scientific phenomenon</td>
<td>Students are learning about something that is difficult to understand because of time, visibility, or size</td>
<td>Template: Creating algorithms to illustrate scientific phenomenon</td>
<td>Example: Creating algorithms to illustrate a food web in Scratch</td>
</tr>
</tbody>
</table>
Creating Algorithms: Identify, Sort & Categorize Objects

An algorithm is a repeatable process that delivers an expected result. One way to use an algorithm is to categorize relationships and characteristics of objects in order to identify them. In the example below, a student collected plants at a park over a year. They created an algorithm to help others identify plants they find in the same area by considering what they look like and when and where they were collected.

Part 1: Identify Characteristics

Collect the object you want to identify. What do you know about them? Create a data table, list, or chart to help you and similarities and differences.

<table>
<thead>
<tr>
<th>Plants</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checker-bloom</td>
<td>Pink with dark green leaves</td>
</tr>
<tr>
<td></td>
<td>Grows March-May</td>
</tr>
<tr>
<td></td>
<td>Found in grasslands</td>
</tr>
<tr>
<td>California Poppy</td>
<td>Orange</td>
</tr>
<tr>
<td></td>
<td>Grows March-November</td>
</tr>
<tr>
<td></td>
<td>Found in grasslands</td>
</tr>
<tr>
<td>Crimson Columbine</td>
<td>Red with yellow center</td>
</tr>
<tr>
<td></td>
<td>Grows March-May</td>
</tr>
<tr>
<td></td>
<td>Found in rocky north facing slopes</td>
</tr>
<tr>
<td>Western Hound’s Tongue</td>
<td>Blue with a white center</td>
</tr>
<tr>
<td></td>
<td>Grows January-March</td>
</tr>
<tr>
<td></td>
<td>Found in rocky north facing slopes</td>
</tr>
<tr>
<td>Blue Eyed Grass</td>
<td>Violet with a yellow center</td>
</tr>
<tr>
<td></td>
<td>Grows March-May</td>
</tr>
<tr>
<td></td>
<td>Found in grasslands</td>
</tr>
<tr>
<td>Bush Moneyflower</td>
<td>Orange</td>
</tr>
<tr>
<td></td>
<td>Grows March-November</td>
</tr>
<tr>
<td></td>
<td>Found in scrub</td>
</tr>
<tr>
<td>Wild Buckwheat</td>
<td>Pink</td>
</tr>
<tr>
<td></td>
<td>Grows June-November</td>
</tr>
<tr>
<td></td>
<td>Found in rocky north facing slopes</td>
</tr>
</tbody>
</table>
How would you help someone filter through all of the objects to identify a single one?

Use the following steps to get started:

1. Identify a single characteristic that sorts all of the objects into categories of approximately equal size.
2. In each category, identify an another characteristic that sorts the objects into additional subgroups.
3. Repeat until each object is in a single category.

List the characteristics you used to sort each group below.

1. Color
2. Time of year
3. Location
4. Number of petals
Part 2: Develop Your Algorithm

Create an algorithm to help someone filter through all of the objects to identify a single one. Use the key below to use the same shapes as the example in your algorithm and create new shapes to show other types of steps. You can also create your own shapes to draft your algorithm on a computer using a tool such as LucidChart, Smartdraw, or Draw.io.
Part 3: Pair Debugging Algorithm

While completing your algorithm, work with a partner to debug — which is to find and fix errors — and improve it:

- Identify the goal of this step?
- Talk with the partner who developed this algorithm to share your process and feedback:
  - What was unclear about this step?
  - How was the intended goal of the step different than what actually happened?
  - What do you need to feel prepared to go to the next step of the algorithm?
- Use this feedback to fix the identified errors.

- Execute the step. Does the goal you identified match what happened?
- Do you feel prepared to do the next step?

There are no more steps in the algorithm to complete.

- Have a new partner use this flowchart to work through your algorithm
- Repeat this debugging algorithm until two partners are able to complete your algorithm without any errors

References:
- Wild Flowers of Golden Gate
- California Plant Finder
- Calscape
Creating Algorithms: Identify, Sort & Categorize Objects

An algorithm is a repeatable process that delivers an expected result. One way to use an algorithm is to categorize relationships and characteristics of objects in order to identify them.

1. **Part 1: Identify Characteristics**

Collect the object you want to identify. What do you know about them? Create a data table, list, or chart to help you find similarities and differences. How would you help someone filter through all of the objects to identify a single one?

First, identify a single characteristic that sorts all of the objects into categories of approximately equal size. Then, identify a characteristic in each subcategory that sorts the objects into additional subgroups. Repeat until each object is in a single category. List the characteristics you used to sort each group below.

1. 
2. 
3. 

2. **Part 2: Develop Your Algorithm**

Create an algorithm to help someone filter through all of the objects to identify a single one. Use the key below to use the same shapes as the example in your algorithm and create new shapes to show other types of steps. You can also create your own shapes to draft your algorithm on a computer using a tool such as LucidChart, Smartdraw, or Draw.io.
While completing your algorithm, work with a partner to debug -- which is to find and fix errors -- and improve it:

Part 3: Pair Debugging Algorithm

Talk with the partner who developed this algorithm to share your process and feedback:
- What was unclear about this step?
- How was the intended goal of the step different than what actually happened?
- What do you need to feel prepared to go to the next step of the algorithm?
Use this feedback to fix the identified errors.

• Have a new partner use this flowchart to work through your algorithm
• Repeat this debugging algorithm until two partners are able to complete your algorithm without any errors

There are no more steps in the algorithm to complete.
Creating Algorithms: Troubleshoot a Design Problem

An algorithm is a repeatable process that delivers an expected result. An engineer might use an algorithm to help others troubleshoot common design problems. In the example below, students completed the circuit arcade activity (https://makerpromise.org/circuit-arcade/). Then, they created an algorithm to help others troubleshoot why a lightbulb in a circuit may not be lighting up.

1 Part 1: Describe Your Design

Sketch and label your design here:

Think about the parts of your design and the purpose of each part (what it does and what makes it important to the overall function). Then, troubleshoot to identify why this part might not work.

<table>
<thead>
<tr>
<th>Part</th>
<th>Purpose</th>
<th>Troubleshoot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output (LED)</td>
<td>The light (LED) turns on, which tells me that my circuit is closed and working.</td>
<td>The switch isn’t closed. The LED isn’t connected correctly (- to - and + to +). The LED burned out.</td>
</tr>
<tr>
<td>Power (Battery)</td>
<td>The battery provides the power needed to turn the light on.</td>
<td>The binder clips handles aren’t touching both the (+) or (-) side of the battery. The battery burned out.</td>
</tr>
<tr>
<td>Input (Switch)</td>
<td>The switch allows my circuit to open and close.</td>
<td>The binder clip handles aren’t overlapping each other.</td>
</tr>
</tbody>
</table>
In this assignment, you are going to create an algorithm to solve a design problem. In the example, there are different shapes that represent different types of steps. Use the key below to use the same shapes in your algorithm and create new shapes to show other types of steps. You can also create your own shapes to draft your algorithm on a computer using a tool such as LucidChart, Smartdraw, or Draw.io.

Start by writing your problem below, and then add clarifying questions and solutions to help someone solve it.
Part 3: Pair Debugging Algorithm

Can you identify the goal of this step?

Talk with the partner who developed this algorithm to share your process and feedback:
- What was unclear about this step?
- How was the intended goal of the step different than what actually happened?
- What do you need to feel prepared to go the next step of the algorithm?
Use this feedback to fix the identified errors.

Execute the step. Does the goal you identified match what happened?

Do you feel prepared to do the next step?

There are no more steps in the algorithm to complete.

- Have a new partner use this flowchart to work through your algorithm
- Repeat this debugging algorithm until two partners are able to compare to complete your algorithm without any errors

Yes

Yes

Yes

No
Creating Algorithms: Troubleshoot a Design Problem

An algorithm is a repeatable process that delivers an expected result. An engineer might use an algorithm to help others troubleshoot common design problems.

1 Part 1: Describe Your Design

Sketch and label your design here:

Think about the parts of your design and the purpose of each part (what it does and what makes it important to the overall function). Then, troubleshoot to identify why this part might not work.

<table>
<thead>
<tr>
<th>Part</th>
<th>Purpose</th>
<th>Troubleshoot</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In this assignment, you are going to create an algorithm to solve a design problem. In the example, there are different shapes that represent different types of steps. Use the key below to use the same shapes in your algorithm and create new shapes to show other types of steps. You can also create your own shapes to draft your algorithm on a computer using a tool such as LucidChart, Smartdraw, or Draw.io.

Start by writing your problem below, and then add clarifying questions and solutions to help someone solve it.
Part 3: Pair Debugging Algorithm

After completing your algorithm, work with a partner to debug -- which is to find and fix errors -- and improve it:

- Have a new partner use this flowchart to work through your algorithm
- Repeat this debugging algorithm until two partners are able to compare to complete your algorithm without any errors

There are no more steps in the algorithm to complete.
Creating Algorithms: Illustrate a Phenomenon

An algorithm is a repeatable process that delivers an expected result. In this assignment, you are going to create an algorithm to illustrate a scientific phenomenon. There are many predictable events that occur in the natural world, such as life cycles, chemical reactions, or Earth’s processes. One way to illustrate these phenomena is by programming an algorithm. If your algorithm represents a mathematical relationship between parts of system, it may be a computational model. If there are no input/outputs or defined mathematical relationships, you are likely simply illustrating a scientific phenomenon. See resources for Creating Computational Models to learn more about how your illustration could be modified to become a computational model. In the example below, a student illustrated a simple food chain with a producer, herbivore and carnivore by programming in Scratch.

1 Part 1: Describe the Phenomenon

Sketch and label the phenomenon you will illustrate here:

![Diagram of a simple food chain with Sun, Producer, Primary consumer, Secondary consumer, and carnivore]
Think about the parts of the phenomenon you will illustrate, the purpose of each part, and if the part will perform an action in your program.

<table>
<thead>
<tr>
<th>Part</th>
<th>Purpose</th>
<th>Action (yes/no). If yes, describe.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun</td>
<td>Provides energy</td>
<td>Appear and disappear on a timer to signal night and day</td>
</tr>
<tr>
<td>Producer</td>
<td>Converts sun’s energy into food through photosynthesis</td>
<td>Grow/reproduce when exposed to the sun</td>
</tr>
<tr>
<td>Primary consumer (herbivore)</td>
<td>Eats producers</td>
<td>If herbivore is pushed, it eats the producer</td>
</tr>
<tr>
<td>Secondary consumer (carnivore)</td>
<td>Eats consumers</td>
<td>If carnivore is pushed, it eats the herbivore</td>
</tr>
</tbody>
</table>

2  Part 2: Create Your Program

Now you will use a computational tool to create your algorithm. There are many tools available to program algorithms, such as coding platforms (e.g., Scratch, Snap, MakeCode) or computational making kits (e.g., Hummingbird Robots, Micro:bit, LegoWedo, Arduino, Raspberry Pi). Your teacher will tell you which tool(s) you may use for this assignment.

Link to Scratch project here
Sun

when clicked
switch costume to sun

when this sprite clicked
forever
switch costume to moon
wait 5 seconds
switch costume to sun
broadcast sunlight
wait 5 seconds

Producer

when clicked
go to x: 24 y: 50
set clone number to 0
delete this clone

when I receive sunlight
if clone number = 0 then
create clone of myself

glide .25 secs to random position

when I start as a clone
set clone number to 1
Primary consumer (herbivore)

- when clicked
  - show
  - go to x: -148 y: -99
  - switch backdrop to backdrop1
  - when I receive hungry
    - hide
  - say I'm full!

Secondary consumer (carnivore)

- when clicked
  - go to x: 118 y: -28
  - glide 1 secs to Hare
  - broadcast hungry
    - say yum!
Part 3: Pair Debugging Algorithm

While completing your program, work with a partner to debug -- which is to find and fix errors -- and improve it:

1. Review the code created for this algorithm. Try an input and observe the results. Does the result match what was supposed to happen?
   - Yes
   - No
     - Talk with the partner who developed this algorithm to share your process and feedback:
       • What input did you use?
       • How was the intended output different than what actually happened?
       Use this feedback to fix the identified issues then go back and start with a new input. If there are no more inputs to test, go to the next questions.

2. Think about the efficiency of the code. Do you observe any repeating patterns in the procedure?
   - Yes
   - No
     - Have a new partner use this flowchart to work through your code
     - Repeat this debugging algorithm until two partners are able to complete your code without errors.

3. Can this code have the same result with less steps?
   - Yes
   - No
     - Talk with the partner who developed this code to share your process and feedback:
       • Where do you observe patterns in the code?
       • In what ways, could you modify the code to be more efficient?
       Use this feedback to fix the identified issues.
       • Have a new partner use this flowchart to work through your code.
       • Repeat this debugging algorithm until two partners are able to complete your code without any errors.
Creating Algorithms: Illustrate a Phenomenon

An algorithm is a repeatable process that delivers an expected result. In this assignment, you are going to create an algorithm to illustrate a scientific phenomenon. There are many predictable events that occur in the natural world, such as life cycles, chemical reactions, or Earth’s processes. One way to illustrate these phenomena is by programming an algorithm. If your algorithm represents a mathematical relationship between parts of system, it may be a computational model. If there are no input/outputs or defined mathematical relationships, you are likely simply illustrating a scientific phenomenon. See resources for Creating Computational Models to learn more about how your illustration could be modified to become a computational model.

1. **Part 1: Describe the Phenomenon**

Sketch and label the phenomenon you will illustrate here:

Think about the parts of the phenomenon you will illustrate, the purpose of each part, and if the part will perform an action in your program.

<table>
<thead>
<tr>
<th>Part</th>
<th>Purpose</th>
<th>Action (yes/no). If yes, describe.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Part 2: Create Your Program

Now you will use a computational tool to create your algorithm. There are many tools available to program algorithms, such as coding platforms (e.g., Scratch, Snap, MakeCode) or computational making kits (e.g., Hummingbird Robots, Micro:bit, LegoWedo, Arduino, Raspberry Pi). Your teacher will tell you which tool(s) you may use for this assignment.

Part 3: Pair Debugging Algorithm

While completing your program, work with a partner to debug -- which is to find and fix errors -- and improve it:

- Review the code created for this algorithm. Try an input and observe the results. Does the result match what was supposed to happen?
- Talk with the partner who developed this algorithm to share your process and feedback:
  - What input did you use?
  - How was the intended output different than what actually happened?
- Use this feedback to fix the identified issues then go back and start with a new input. If there are no more inputs to test, go to the next questions.
- Talk with the partner who developed this code to share your process and feedback:
  - Where do you observe patterns in the code?
  - In what ways, could you modify the code to be more efficient?
- Use this feedback to fix the identified issues.
  - Have a new partner use this flowchart to work through your code.
  - Repeat this debugging algorithm until two partners are able to complete your code without errors.

Attribution 4.0 International (CC BY 4.0)