

Puzzling Procedure

Grade Levels: 3-5

Duration: 40 min

Computational thinking is a problem-solving process that is used in everyday life as well as on computer programs. In this lesson, students apply their computational thinking skills to jigsaw puzzles. They'll create an algorithm or set of instructions to help others solve jigsaw puzzles.



Outline

Frame the Activity	15 min total
Engage Student Interest	5 min
Introduce Computational Thinking	5 min
Introduce the Activity	5 min
Activity	25 min total
Create Algorithms	10 min
Test and Share	10 min
Debrief	5 min

Grade Levels: 3-5

Duration: 40 min

Computational Thinking Focus

Algorithms

Objectives

Students will...

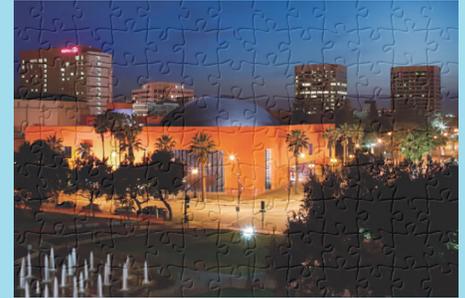
- Analyze their jigsaw puzzle-solving process.
- Develop an algorithm that describes a similar process.

Materials

- Paper, notecards or sticky notes
- Writing utensils
- *Optional:* Jigsaw puzzles
- *Optional:* [Computational Thinking Elements posters](#)



Try one of our digital jigsaw puzzles on our [Tech at Home website](#). Change the number of pieces to make it simpler or more challenging!



Prep

1. Work through the activity and try to create your own algorithm.
2. *Optional:* Hang up The Tech's Computational Thinking Elements posters.
3. Divide the class into pairs or small groups of three for the activity. Working with others, students will likely discover steps that their classmates may have missed.



Frame the Activity

Engage Student Interest (5 min)

1. Tell students: *Today, we're going to think about solving puzzles, specifically jigsaw puzzles.*
2. Hook student interest by making it relevant to them. Options include:
 - **Prior Knowledge:** Have students share personal experiences with puzzles. Perhaps they have taught a younger sibling how to do puzzles. Maybe they do large puzzles as a family, or perhaps they find them frustrating. Getting a sense of their prior experience can help you connect the concepts to the real-world for them.
 - **Puzzle:** Have students do a jigsaw puzzle on their own. Use a resource provided in the materials, or have physical puzzles for them to solve. After doing the puzzle, have learners interview each other about their process or participate in a class discussion about the experience.
3. Briefly introduce the design problem: *Today we're going to develop a strategy and instructions for solving puzzles using computational thinking.*

Introduce Computational Thinking (5 min)

1. Ask students if they have heard of **computational thinking**: *When I say computational thinking, what does that make you think of?*
 - Write down key words and ideas used to describe computational thinking.
 - There will be a variety of answers and ideas. Lead them to the following understanding:
 - *It is a problem-solving process that is important for computer science but is also used more broadly. Since puzzles are problems, computational thinking can be used when solving them.*

2. *Optional:* If you have the Tech's Computational Thinking Elements posters up, refer to them at this time.
3. Explain to students that for this activity they are going to focus on one computational thinking element: **algorithms**.

 The Tech's **Computational Thinking Tech Tip** can be a good resource here.

- Share the definition and the real-world example so that they can see how they already use algorithms every day.
- It may be useful to post this definition and example up on the board.

1

Algorithms

Step-by-step instructions to solve a problem. When solving a problem, it is important to create a plan for your solution. Algorithms are a strategy that can be used to determine the step-by-step instructions on how to solve the problem. Algorithms can be written in plain language, with flowcharts, or pseudocode.

2

3

Real-world examples of algorithms

We use algorithms daily, normally in the form of step-by-step instructions. Recipes, listed instructions for making furniture or building blocks sets, plays in sports, and online map directions are all examples of algorithms.



Introduce the Challenge (5 min)

1. Use a narrative or story to make a real-world connection for students.

Some examples:

- *We need to teach someone who has never solved puzzles before how to do it.*
- *Your job is to teach an artificial intelligence program in a computer how to solve puzzles. Give them the instructions they should follow.*

2. The activity is outlined as a design problem, with criteria and constraints below. Explain it to students and address any questions they might have.

Design Problem	Develop a strategy for solving jigsaw puzzles.	
Criteria	<ul style="list-style-type: none"> • Write an algorithm (instructions) for how to solve a puzzle that has at least 5 steps. • Must be understandable by someone who has never solved a puzzle. 	 
Constraints	You have 10 minutes to create your algorithm.	

- 
3. Before students begin, walk through the process with the class together.
 4. Ask students: *When you get a puzzle, what's the first step?*
5. Take suggestions from the class. Write them on the board or on sentence strips so you can add, rearrange and sort like ideas later.
 - Their suggestions are likely to be helpful during idea generation, even if they're not correct for this exact answer.
 - The first step should be something like: "Open the box and dump the pieces out," or "Flip over all of the pieces so you can see the image you're working with."
 6. If you haven't already, place students in pairs or small groups of three for the activity.
 7. Tell students: *Now you're going to work with your partner/group to write step-by-step instructions for solving a puzzle. Remember that a set of step-by-step instructions is called an algorithm.*
-

Activity

Create Algorithms (10 min)

1. Students should discuss their own strategies for solving a jigsaw puzzle with their partner.
 - At this point, it may be helpful to have a puzzle out on the table for them to refer to, but NOT be trying to solve.
2. As they discuss, students should write down each instruction and put them in order. Adjust the level of specificity you require of students based on the grade level, understanding of puzzles, or comfort with language.
 - They should write each step on a separate card or a sticky note so they can get reordered, edit or add to.
 - Remind students that they are writing these instructions for someone with no jigsaw puzzle experience, so they should be as specific as possible.
3. Encourage students to do their best. They may not finish the task within the time limit, but the goal is to begin thinking about the steps that might be needed to create a set of instructions.

Test and Share (10 min)

1. Once time is up, ask a group to volunteer their algorithm by writing the steps on the board, in order, leaving space in between each one.
2. Try running through the algorithm and ask the class for input that will make the algorithm more precise.
 - Have others contribute steps that aren't represented and insert them in the correct spots in order to build a classroom master algorithm.

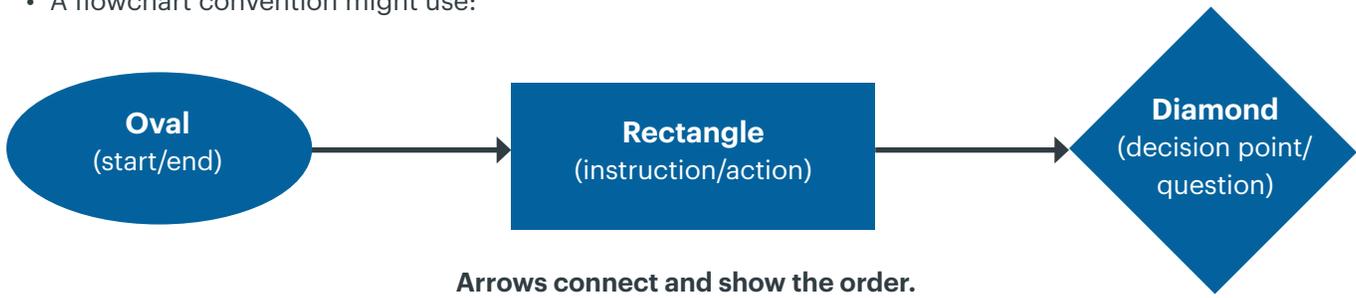


Tips

- One technique is to have one student describe their steps for solving a puzzle while their partner writes down the algorithm, then switch roles. Differences in their respective algorithms may lead to more iteration!
- If students finish early, they can show their algorithms to another group and get feedback.
- If you have access to jigsaw puzzles, students can "test" their algorithms on another group by having them try to follow the directions exactly. Encourage the students to take instructions literally and to kindly point out if something needs more clarity. This process would be similar to "debugging" in a computer program.

Adaptations for Advanced Students

- 1 Flowcharts:** Students who are more familiar with computational thinking may benefit from organizing their instructions in a flowchart.
- A flowchart is another way to write an algorithm.
 - A flowchart convention might use:



- 2 If/Then Logic:** Use the step of “test a piece” to introduce If/Then logic. The concept of If/Then falls under conditionals. Conditionals are programming concepts that evaluate if something is true or false.
- Ask students: *When you’re trying a piece to see if it connects, what happens? What do you do if it doesn’t match?*

	Listen for students to give responses like: “You try it and...”	Within a computer program, this statement of action would be written in an if/then format.
1	If it works, you move on and try to find a new piece to connect.	If the piece fits properly, then you can move on to find a new piece.
2	If it doesn’t match, you rotate the piece and try all possible sides.	If the piece doesn’t match, then you rotate the piece to try all the possible sides or orientations.
3	If it still doesn’t match, you return it to the pile and keep looking.	If the piece still doesn’t match, then you return the piece to the pile and keep looking.

Point out to students that they are already using if/then logic in their work. In order to transfer this to a computer program, in many cases they would just need to add the “if/then” terminology. Have students try to create their own if/then statements for their algorithm.

- 3 Brute Force Algorithms:** When programmers are writing algorithms they often consider the “brute force algorithm” which could be used to solve a problem. A brute force algorithm is a method that checks every possibility until a solution is found. In puzzles, the brute force algorithm might be to check each puzzle piece against the other until the puzzle is solved. Considering the brute force algorithm can provide a comparison for more advanced students looking at simplifying their algorithms.

Debrief (5 min)

1. Lead a short debrief with students which focuses on their use of computational thinking and algorithms.
2. Help students make connections between the skills they used in this activity and the rest of their lives. Build their confidence in their ability to use computational thinking in any setting.
3. Students may also notice that there are multiple ways to get to the same solution. A variety of different algorithms might give someone the instructions they need to solve a puzzle. This aligns with The Tech's Computer Science Education Principle VII: "There are many solutions to a given problem. Different computer programmers might write algorithms differently, but still come up with the same result."
4. Possible debrief questions include:
 - *What was hard about writing an algorithm?*
 - *What would you do differently next time?*
 - *What is useful about writing an algorithm?*
 - *What was most important to remember when writing instructions?*Discussion may include the precision of language needed to write an algorithm or how information needed to be organized or thought through in advance.
5. To deepen the connection with computer science, use debrief questions like:
 - *How do you think this relates to computer programming or Artificial Intelligence (AI)?*
 - *We focused on algorithms, but what other computational thinking elements do you use when solving puzzles?*
6. During the discussion students may notice:
 - In a computer program, we need to write precise instructions in the correct order for a computer to execute a solution properly.
 - Artificial intelligence (AI) often builds upon algorithms. For instance, if teaching an AI to solve a puzzle, you would give it a set of instructions to solve a puzzle and it would get better at completing puzzles over time. There have been several famous examples of teaching AIs to play games like Go and chess.
 - Sorting the pieces by shape or color or type is an example of **pattern recognition**.
 - Breaking the image down into identifiable objects or areas, or even into a frame and an inside is an example of **decomposition**.



Extensions

Have students continue to iterate on and improve their algorithm. This can be done within class time or as an at-home extension.

- Have students test the algorithm with more difficult or challenging puzzles. What would they need to adjust or change? What steps are missing?
- Have students test their algorithm at home on family members. Does it work the same way on people of different ages? Did older or younger people need different instructions?



Tech Tips

See our [educator guides and videos](#) for more design challenge facilitation techniques.

- Computational Thinking
- Sharing Solutions

Standards Connections

California Computer Science Standards

Grades 3-5	AP.10	Compare and refine multiple algorithms for the same task and determine which is the most appropriate. (P3.3, P6.3)
------------	-------	--

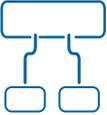
Common Core State Standards

English Language Arts

College and Career Readiness Standards: Production and Distribution of Writing

Grades K-12	4	Produce clear and coherent writing in which the development, organization, and style are appropriate to task, purpose, and audience.
	5	Develop and strengthen writing as needed by planning, revising, editing, rewriting, or trying a new approach.

Vocabulary

	Computational Thinking	A problem-solving process that can be broadly applied across content areas and everyday life.
	Decomposition	Breaking down problems into smaller problems.
	Pattern Recognition	Recognizing if there is a pattern and determining the sequence.
	Algorithms	Step-by-step instructions to solve a problem.
	Abstraction	Generalization of a problem – focus on the big picture and what’s important.