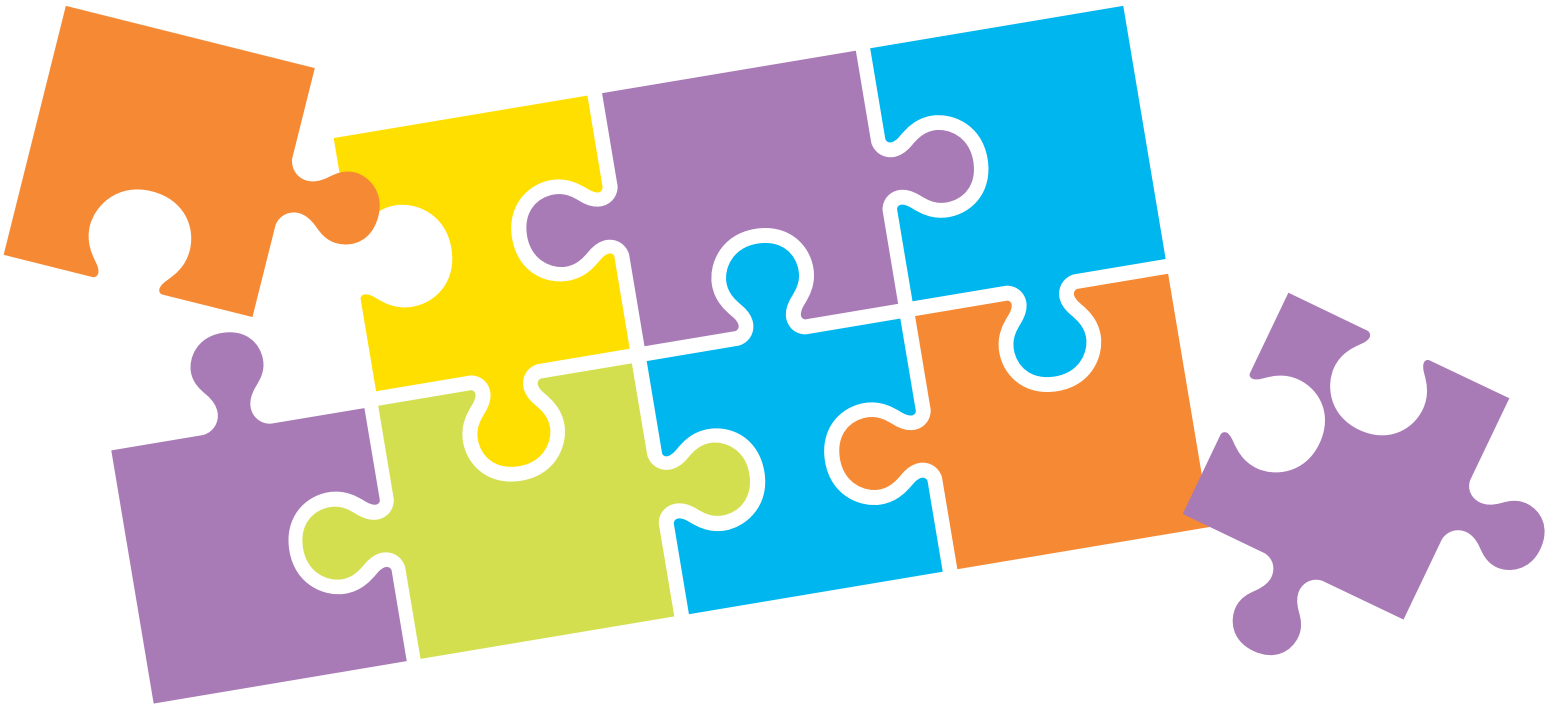
The Tech Interactive at Home

Puzzling Procedure

Who says all the fun has to happen at The Tech Interactive? This computational thinking activity will expand your problem-solving skills and can be done anywhere!



Introduction

Have you ever tried to think like a computer? Guess what? You probably do it all the time without even knowing! Computational thinking is a problem-solving process that is used in everyday life as well as by computer programs. In this fun activity you'll apply your computational thinking skills to jigsaw puzzles. In addition to doing a puzzle, you'll create an algorithm or set of instructions to help someone else solve jigsaw puzzles.

Design Challenge

Create a step-by-step algorithm to describe your puzzle-solving strategy.

Subject:
Computational Thinking

Age:
7+

Time:
30+ minutes

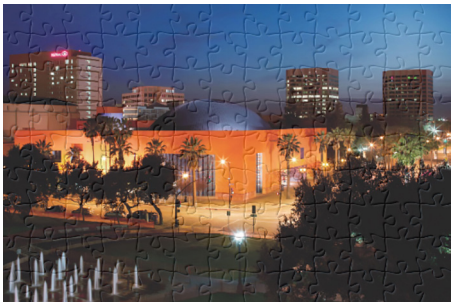
Key Concepts:
Algorithms, computational thinking, computer science

Materials

- Paper, notecards or sticky notes
- Writing utensils (paper, pencil, marker, etc.)
- Jigsaw puzzle

Puzzle Options

- Use a physical jigsaw puzzle.
- Try one of our digital puzzles on our Tech at Home website! Change the number of pieces to make it simpler or more challenging!
- Feeling creative? Make your own puzzle by drawing a picture or using an old photo and cutting it out into random shapes!



Instructions

Do a puzzle

1. Refresh your puzzle-solving skills! Start by doing a jigsaw puzzle. Any size will do!
2. As you're solving the puzzle, pay attention to the details that you think about. If you can, jot down a few quick notes along the way.
   - What do you do first? *Perhaps you turn all the pieces over? Or maybe you spread them out so you can see them.*
   - Does your strategy stay the same for the whole puzzle? Does it change as you complete certain steps?
   - What do you look for at each step?

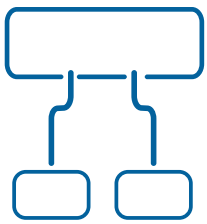
Tip: If it's hard to keep track of your train of thought, ask someone else to solve a puzzle and interview them about their thought process.

Write an algorithm

Look at your notes from solving the puzzle. Write each step down on a notecard, sticky note or a slip of paper.

Tip: Writing each step on a separate sheet allows you to rearrange steps and insert new ones as your algorithm develops!

1. Start broad and then fill in more specific instructions.
2. It can help to start by breaking down the puzzle-solving process into smaller parts. This is called **decomposition** in computational thinking.
   - What do you do in any set-up?
   - What do you do first?
   - Do you sort the pieces? How?
   - How do you find two pieces that fit together?
   - What do you do next?
3. Organize the steps into the correct order.
4. Then, step back and evaluate your algorithm. Are you missing any steps?



Algorithm: Step-by-step instructions to solve a problem. Algorithms are an important part of computer programming and computational thinking.

Real-world examples: Recipes, instructions for making furniture or building blocks sets, diagrams of sports plays, and online map directions.

Consider the **brute force algorithm** which could be used to solve a puzzle. This can help you compare and simplify your own algorithm. A brute force algorithm is a method that checks every possibility until a solution is found. Perhaps in puzzles, the brute force algorithm would be to check each puzzle piece against the others until the puzzle is solved (without looking at other clues like shape, color or pattern).

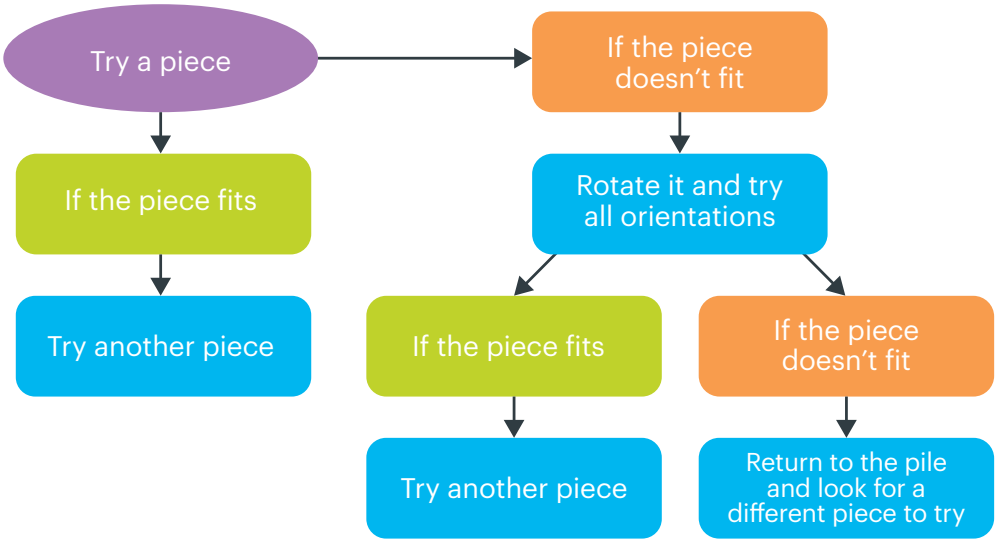
Test your algorithm

1. Don't forget to test and iterate! Try to have someone else solve a puzzle using your algorithm. Have them follow your instructions exactly and don't give them any extra hints.
2. Make observations and reflect on how it went.
3. "**Debug**" your algorithm.
   - Where does your user get stuck?
   - Should the order of your instructions change at all?
   - Does your algorithm need more detail?
4. Have fun revising and testing your algorithm.
5. Remember with a complex problem like a puzzle, there are usually multiple solutions and not necessarily a "right" answer.
6. Think about how it felt to use computational thinking and algorithms.
   - What is useful about writing an algorithm? Where did it get in the way?
   - What was most important to remember when writing instructions?

Software **bugs** are errors in a computer program that result in incorrect or unexpected behavior.

Computer Science Tips

- As you get more comfortable, try using **If/Then logic** to describe the decision-making process.
  - Map out this logic using a flowchart with simple symbols and graphics to show the decision-making process.



Did you notice? This guide is our own algorithm for you! How did we do? What steps were we missing?

Share Your Results! Keep us posted about your progress on social media with **#TheTechatHome**.

Explore More

- Try your algorithm with more family members and friends. You can even send your instructions to someone far away. Does it work the same way for everyone? Test your algorithm with more complicated puzzles. What do you need to adjust?
- You see and use algorithms all the time. Once you've created an algorithm for puzzles, try creating an algorithm for another task, problem or process in your daily life.

The Tech Interactive at Home

thetech.org/athome