Journal of Technology-Integrated Lessons and Teaching
(2025) Volume 4, Issue 1; DOI: 10.13001/jtilt.v4i1.9205

Puzzling Our Way into Computational Thinking

David J. Mulder, Dordt University

Overview

This unplugged activity is a brief, practical introduction to computational thinking that uses an accessible and decidedly low-tech approach: solving a jigsaw puzzle. The skills needed to collaboratively solve a jigsaw puzzle illustrate the key concepts of computational thinking in a straightforward way that makes the basics of decomposition, pattern recognition, abstraction, and algorithmic thinking come to life. The learning representation included here was taught as part of a professional development workshop for PK-12 teachers but could easily be adapted to use with learners from upper elementary grades through middle school, high school, or university.

Topics: Collaborative Learning, Computational Thinking, Hands-on Learning

Time: 30 minutes

Materials

- Children's jigsaw puzzles (48-piece puzzles work well) – one for each group of 3-5 participants
- Zip-top storage bags
- Whiteboard and markers
- You are not a computer presentation

Setup

A classroom setup with tables works best. Arrange each table for 3-5 participants. Each participant should be able to see the whiteboard where ideas will be shared. To prepare the puzzles, remove each puzzle from its box and place the pieces in a zip-top storage bag. Keep the boxes for storage, but groups should receive pieces of their puzzle without the box or any picture of how the puzzle should appear.

Context-at-a-Glance

Setting
A professional development workshop for PreK-8 teachers at a non-public school located in the upper Midwestern United States.

Modality
Face-to-face

Class Structure
This lesson was implemented as part of a series of workshops related to computational thinking and educational robotics. This lesson was part of the first workshop in the series.

Learner Characteristics
Twenty-four PreK-8 educators participated in the workshop. Their experience ranged from first-year teachers to veteran educators with over 30 years of classroom experience. Only a handful of participants had any previous experience with coding or computational thinking.

Instructor Characteristics
A university professor with over 25 years' experience teaching in PreK-12 and higher education. He previously taught middle school and high school computer science courses, and currently teaches educational robotics and other STEM education courses. He provides STEM professional development opportunities to PreK-12 educators.

Development Rationale
This activity was developed to introduce computational thinking in a way that is accessible, meaningful, and relevant by using non-digital materials that were already familiar to the participants.

Design Framework
Social Constructivism (see Brau, 2022; Palinscar, 1998)

Context and Setting

Emphasis on computer science education and computational thinking skills is on the rise in K-12 education (Klein, 2023, Vegas & Fowler, 2020). As such, I am regularly invited to provide professional development (PD) opportunities for teachers in PK-12 schools, particularly related to topics in STEM education and educational technology, which are my areas of expertise. My education and experience have prepared me to do this sort of training, and I thoroughly enjoy it! I spent 11 years serving as a middle school math and science teacher, 3 years as a technology director and K-8 computer science teacher, and 13 years in higher education, where I teach courses in educational technology, STEM education, and educational foundations.

Through grant funding, my department was recently able to purchase some materials to support our students' learning about computer science education and computational thinking. Having access to these resources got me thinking about outreach opportunities in our area as well. I contacted local school districts to share that I am willing and able to provide PD on these topics for their teachers. The email I sent to administrators offering my services explained what computational thinking is, the possibilities of using computational thinking strategies across the curriculum, and the enjoyable aspects of moving from "novice to knower" with regard to learning about educational robotics—a good reminder for teachers of how it feels to be brand-new at learning something. These workshops were designed to help remind teachers how much courage it can take as a learner to try something new; something teachers sometimes forget, because we are experts in our content.

As part of this outreach, I was invited by a local non-public PK-8 school to provide a series of PD experiences to help introduce their faculty to some topics in computer science education that could have broad, cross-curricular appeal. Working with their Director of Learning in the planning process, we decided to focus on computational thinking, an introduction to block-based coding, and educational robotics. This lesson was a 30-minute segment of the full PD that focused on understanding computational thinking skills.

Each session of the full PD was scheduled for an hour-long block at the end of a Wednesday school day when the students had an early release so the staff could focus on PD. Twenty-four PK-8 teachers and paraeducators participated, representing a variety of grade levels and content areas. The participants in these PD opportunities had a range of experience as teachers. Some were in their first year, and several had over thirty years of experience as professional educators. None of the participants were currently teaching computer science as a primary part of their content responsibilities, though two of the teachers were tasked for teaching computer applications as part of their assigned curricula. Additionally, one of the paraeducators had some experience with block-based coding and educational robotics through an enrichment activity she helps facilitate.

The lesson described here came about serendipitously as I was preparing for our first session in this series of PD opportunities. I was reflecting on how I could best introduce the concepts of computational thinking as I was walking through the student lounge adjacent to our Education department offices. We always have a jigsaw puzzle out on one of the tables for students (and faculty) to work on collaboratively when they need a break from their academic work. Seeing the puzzle illustrated in Figure 1, I was suddenly struck with an idea for introducing computational thinking through an unplugged, readily accessible approach.

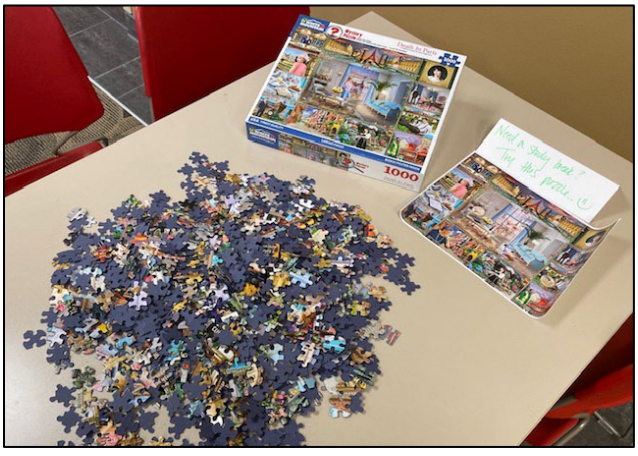

Figure 1. A photo of a communal jigsaw puzzle in our student lounge. The sign next to the puzzle box says, "Need a study break? Try this puzzle."

This lesson, described in the following section, illustrates the introduction to a series of PD workshops conducted with this same group of participants. The first workshop, including this

activity, introduced the idea of computational thinking, and gave a variety of opportunities for the participants to make connections to their grade levels and content areas. The second workshop in the series introduced block-based coding using Sphero BOLT+ robots, and participants learned the basics of how to use pattern recognition and algorithmic thinking particularly to address beginner-level coding challenges. The third and final workshop used Sphero BOLT+ robots to reinforce the things participants had previously learned and utilize all four computational thinking skills to achieve more demanding coding challenges.

Learning Representation

During this lesson, italic text identifies questions or prompts for the learners.

Introduction

Computational thinking is a structured approach to attacking complex problems (Wing, 2006). While generally considered a subset of computer science, computational thinking can be broadly applied to many different disciplines (Hunsaker, 2020; Lodi & Martini, 2021). In a seminal article from 2006, Wing suggests that "Computational thinking is a way humans solve problems; it is not trying to get humans to think like computers" (p. 35). Instead of thinking like a computer, computational thinking is a key part of the creative work that humans do to attack authentic problems—the sort of "wicked problems" that crop up in almost every discipline.

While there are different definitions of what skills should be thought of as part of computational thinking (Cansu & Cansu, 2019), four cognitive skills are included in depictions of computational thinking across the literature (Campbell & Heller, 2019; Cansu & Cansu, 2019; Hunsaker, 2020; Lodi & Martini, 2021; McNicholl, 2019; Yin et al., 2020). These four key computational thinking skills are:

1. Decomposition
2. Pattern recognition
3. Algorithmic thinking
4. Abstraction (Campbell & Heller, 2019)

In this learning segment, the instructor leveraged the strengths of social constructivism as an approach for learning, prioritizing active engagement with other learners and facilitated discourse about the learning experiences (Palincsar, 1998). The idea of computational thinking was introduced through a hands-on activity, and then a discussion was facilitated with the learners to help them understand what these four component skills look like in practice. This approach illustrates the tenets of social constructivism, drawing on the strength of the group along the lines of Vygotsky's approach of learning by doing and by interacting with others in support of the learning process (Brau, 2022). The role of the instructor in such a social constructivist setting shifts to that of facilitator, asking questions, providing support, and dialoguing with learners throughout the learning process (Brau, 2022).

Opening the Session

The session began by inviting participants to sit down in groups of about five people at each table. They were given the opening prompt, "*When you hear the phrase 'computational thinking,' what ideas, concepts, and pictures come to mind? Think out loud with your tablemates about what 'computational thinking' might look like…"* Participants discussed their ideas, for about two minutes, after which they were invited to share the things that had come up in their table conversations. Their associations for "computational thinking" included things like computers, programming, and artificial intelligence. The instructor affirmed their ideas, and suggested that by the end of this session, they would have a clear definition for what computational thinking is.

As the participants moved into the activity for this session, the instructor shared three learning targets that would guide their work together:

1. *I can define and give examples of computational thinking.*
2. *I can commit to playfulness, collaboration, and celebration of learning.*
3. *I can take reasonable risks that will help me practice new skills.*

Hands-on, Minds-on Collaborative Investigation

At this point, the instructor gave each group a 48-piece jigsaw puzzle in a clear zip-top bag.

Participants did not receive the box or any indication of what the finished puzzle should look like.

They then received this instruction: "*Let's play! Grab your jigsaw puzzle and work with your group to complete it as quickly as possible.*" Figure 2 illustrates how a group began completing the puzzle.

Figure 2. A group early in their work putting together the puzzle. This group rapidly sorted pieces by color, which was a helpful strategy for them.

In each group, someone opened the bag, poured out the pieces onto the table, and the hands flew into action. Several groups joked about not receiving a picture of what they were trying to compose as they worked. But they quickly began flipping the pieces face-up, sorting them out, and snapping the pieces into place. There was a flurry of activity and lots of conversation and encouragement as they worked. Figure 3 illustrates the quick progress a group made as they collaborated on putting the puzzle pieces together.

Figure 3. A group making quick progress. This group looked for edge pieces, and had a few members work on the "sky" while others worked on the "grass."

Most groups had their puzzles completed in about three minutes, as seen in Figure 4. Two groups, however, needed a little more time, as they had lenticular puzzles (the kind where the picture shifts as you move your field of vision from side-to-side.) These ended up being much more challenging to solve as compared to the more basic puzzles. All the groups were finished with their puzzles in five minutes or less.

As groups finished putting the pieces together, they were encouraged to think through *exactly* what they did to solve the puzzles, as that would be the next topic of discussion.

Figure 4. A group finishing their puzzle. This group quickly recognized the characters in the puzzle, which they noted helped give them context to figure out what the finished puzzle should look like.

Processing the Puzzling

After all the groups had their puzzles completed, they were invited to get specific about the strategies they used for solving their puzzles quickly. Some ideas that emerged included:

- Finding the corner pieces and edge pieces and sorting them out from the non-edge pieces.
- Flipping all of the pieces face-up so the colored sides were showing.
- Sorting the pieces by color.
- Using clues from the pieces to make sense of what the picture of their puzzle was.
- Using the shapes of the pieces to determine where they actually fit into the puzzle.
- Starting from the outside and working towards the inside of the puzzle.

- Intuitively subdividing the work so different people focused on different parts of the task.

As they shared their responses, the suggestions were affirmed and written down on the whiteboard. When they finished suggesting strategies for solving their puzzles, the participants had a few minutes to examine the list and see if they had other strategies to add. Hearing none, the instructor asked a clarifying question to the group: *"How did you know if you were successful in this task?"*

This question took a little time for them to answer. The immediate response for most groups was something like, "We just knew we were done." But this was probed a bit further: *"What does 'done' mean for this task?"* Eventually the participants agreed that they knew they were successful when they had used their pieces to complete a picture, and they had no leftover pieces. This was added to the whiteboard as well, which can be seen in Figure 5.

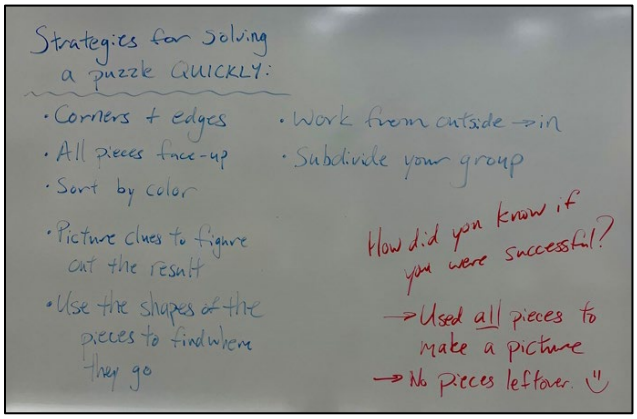

Figure 5. A photo of the whiteboard, including the ideas the group shared about their strategies for quickly solving their puzzles, and how they would know if they were successful.

The idea of this final question was to help them keep the focus on the main goal of the whole activity: solving an actual problem. This is central to the idea of computational thinking: using creative and critical thinking strategies to solve authentic problems by "drawing on concepts fundamental to computer science" (Wing, 2006, p. 33).

So…What is Computational Thinking?

At this point, the content took a turn to some direct instruction. The instructor explained to the participants that what they had been doing as they solved their puzzles was, in fact, applying computational thinking skills.

The instructor reiterated that the goal of computational thinking is not only to solve computer science problems, but to use structured, strategic thinking to attack authentic problems, including putting together a jigsaw puzzle. The four computational skills posited by McNicholl (2019) was also explained. The definitions presented to the participants for these four computational skills were:

1. Decomposition: Breaking the problem down into parts
2. Pattern Recognition: Seeing similarities and differences, and making connections
3. Algorithmic Thinking: Devising step-by-step processes to solve the problem
4. Abstraction: Identifying what actually needs to be done to solve the problem and filtering out noise

After this explanation, participants were encouraged to see if they could find examples of all four of these thinking skills in the way they approached solving their puzzles.

Groups immediately named pattern recognition as a key skill for solving the puzzle, which did not surprise the instructor, as so much of solving a jigsaw puzzle depends on finding patterns in colors and shapes of the pieces.

Decomposition was also quickly identified. Participants named the ways they had worked to break down the big task of "solve the puzzle" into component parts. This led most groups to immediately explain the algorithms, the steps, they used to solve the puzzle. Based on the strategies used for solving their puzzles, which were still listed on the whiteboard, most of the groups' algorithms were similar:

1. Flip all the pieces to be picture-side up.
2. Sort the pieces by color and/or shape.
3. Put the edge pieces together.
4. Fill in the middle pieces to complete the puzzle.

Abstraction, identifying what actually needs to be done to solve the problem and filtering out noise, was the most difficult of these four computational thinking skills for the participants to name, but eventually the instructor helped them list things like:

- Instead of trying to solve the whole problem all at once, focus on just one part of the problem, like finding the edge pieces
- Don't worry about sorting out every single piece before you start putting pieces together.

In this particular example of using computational thinking, the main goal of "use all the pieces to make a picture" was a helpful general rule, and that helped the participants conceptualize what abstraction is about: generalizing the particular. This sort of instructor facilitation is an important part of social constructivist learning theory (Brau, 2022).

A question several participants wanted answered was, "It seems like we're saying some parts of solving the puzzle are examples of several of these components of computational thinking at the same time. Is that right?" My response to them was that this is true; a component task like "start with the edge pieces" could be part of their algorithm for solving the problem, and it was *also* part of the way they decomposed the broader problem, and *also* one of the patterns they recognized in the pieces. That answer satisfied the adult learners in this group.

Wrapping Up the Lesson

To close out this lesson, the instructor reiterated that computational thinking is a skillset for solving complex problems, and while computational thinking is an essential part of computer science, it has broad application to many other disciplines. The participants were also encouraged to not think of computational thinking as "thinking like a computer," but rather a way of:

- Developing thinking skills to analyze complex problems.
- Learning to clearly articulate a plan.
- Designing a "good enough" plan to attack the problem.
- Refining that "good enough" plan through iterations.

The instructor also affirmed that these thinking skills are used by programmers as they are coding solutions to problems, which is the general intent of computational thinking within computer science. It was suggested that in future sessions the groups might work on making explicit connections between computational thinking skills and their own content areas. Hunsaker (2020) has a great many ideas for how computational thinking can be applied across a wide variety of disciplines, from more "obvious" options like math, engineering, and science as well as perhaps less-obvious content areas, including language arts, foreign language, social studies, music, family and consumer science, and physical education.

To close out the lesson, the learning targets for the session were revisited, and participants were asked to rate themselves on a 3-point scale (thumbs-up, thumbs-sideways, thumbs-down) of how they thought they did on each of the targets. Thumbs-up was the predominant response all around!

Critical Reflection

Overall, I was very pleased with how this PD experience unfolded for the participants. They came away with a practical, first-hand experience that helped them understand the fundamental principles of computational thinking at an intuitive level. The participants were able to apply the things they learned in this first session into other activities in later sessions as they learned about coding and educational robotics. We also continued to refer back to this shared puzzle experience as a way of making sense of the other things we did later on in this series of PD workshops. We continued to use the language of computational thinking skills (decomposition, pattern recognition, algorithms, and abstraction) throughout our work together.

Since using this approach of introducing computational thinking with puzzles, I have used this activity with several other groups of educators and found similar results. I have also used it with a group of 16 middle school aged students at a summer camp to help introduce the idea of computational thinking as part of an experiential learning opportunity of programming robots. I did not share that experience of puzzling with middle school students as the focus of this lesson because a colleague was the one actually leading that session, and I just received permission to try out this activity in the first 20 minutes with the middle schoolers to see how it went with a group of young adolescents. I did not substantially change anything about the way I presented the activity to the young adolescents from the way I presented it to the adult learners who participated in the activity described above. I used the same instructional moves to engage them in

completing the puzzles and facilitating a brief discussion with them afterward. I was encouraged how well it worked with the middle schoolers, and I think it merits more exploration. I'm convinced based on these experiences with both adults and young adolescents that this activity could work well in a wide variety of contexts and different age groups. After successfully implementing this learning representation multiple times, I continue to reflect upon the importance of connecting this experience to further learning about computational thinking. Instructors using it in a similar setting might be able to use it as described, and those adapting it to use with other age-groups or in specific content areas might be able to use it similarly. But in either case, having a clear connection to future instruction would be valuable. This learning representation will not do all the work of teaching learners about computational thinking.

That said, I do believe this jigsaw puzzle strategy is a useful way to begin learning about computational thinking, and to understand the key vocabulary related to computational thinking skills: decomposition, pattern recognition, algorithmic thinking, and abstraction. The unplugged activity described here should be considered just that, an intentionally low-tech introduction to computational thinking that should be robustly supported and expanded by a variety of other experiences with computing and programming.

References

Brau, B. (2022). Constructivism. In R. Kimmons (Ed.), *Education research: Across multiple paradigms* (Section 3.3). EdTech Books. https://edtechbooks.org/education_research/constructivismy

Campbell, L. O., & Heller, S. (2019). Building computational thinking: Design and making in teacher education. In J. Leonard, A. C. Burrows, & R. Kitchen (Eds.), *Recruiting, Preparing, and Retaining STEM Teachers for a Global Generation* (pp. 163-189). Brill.

Cansu, F. K., & Cansu, S. K. (2019). An overview of computational thinking. *International Journal of Computer Science Education in Schools*, *3*(1), 17-30. https://doi.org/10.21585/ijcses.v3i1.53

Hunsaker, E. (2020). Computational thinking. In A. Ottenbreit-Leftwich, & R. Kimmons (Eds.), *The K-12 Educational Technology Handbook* (Section 2.3). https://edtechbooks.org/k12handbook/computational_thinking

Klein, A. (2023, November 1). *Computer science courses are on the rise—But girls are still half as likely to take it.* Education Week. https://www.edweek.org/teaching-learning/computer-science-courses-are-on-the-rise-but-girls-are-still-half-as-likely-to-take-it/2023/11

Lodi, M., & Martini, S. (2021). Computational thinking, between Papert and Wing. *Science & Education*, *30*(4), 883-908. https://doi.org/10.1007/s11191-021-00202-5

McNicholl, R. (2019). Computational thinking using code.org. *Hello World, 4*, 36-37. https://issuu.com/raspberry314/docs/helloworld04

Palincsar, A. S. (1998). Social constructivist perspectives on teaching and learning. *Annual review of psychology*, *49*(1), 345-375. https://psycnet.apa.org/doi/10.1146/annurev.psych.49.1.345

Vegas, E. & Fowler, B. (2020, August 4). *What do we know about the expansion of K-12 computer science education?* The Brookings Institution. https://www.brookings.edu/articles/what-do-we-know-about-the-expansion-of-k-12-computer-science-education/

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Yin, Y., Hadad, R., Tang, X., & Lin, Q. (2020). Improving and assessing computational thinking in maker activities: The integration with physics and engineering learning. *Journal of Science Education and Technology, 29*(2), 189-214. https://doi.org/10.1007/s10956-019-09794-8

About the Author

David J. Mulder serves as Professor of Education at Dordt University, where he teaches courses in educational foundations, STEM education, and educational technology in both the undergraduate Teacher Preparation Program and the Master of Education program. His research interests include technology integration, social presence in online learning, and digital citizenship. He can be contacted about this lesson at david.mulder@dordt.edu.

Acknowledgement

The development of this learning representation and the materials included herein are based upon work supported by the National Science Foundation under Grant No. (DUE-2243334). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Sharing & Modification Permissions