Algorithms in Computational Thinking

McKenna, J., Smith, A.

In the 1760s, European mapmakers pasted maps onto wood, cutting them into small pieces. With exploring the world as a universal fascination, these puzzles quickly became an educational tool both frustrating and delighting minds ever since.

Some puzzles are more challenging than others; the size and number of pieces, the intricacies of the image and the color range are all influential. And the solving process often involves patterns when progress is both linear and non-linear. We've all met those moments when even a daunting puzzle seems to miraculously solve itself, and we've faced those nights when we wonder what we've gotten ourselves into by dumping a thousand pieces of cardboard out in disarray.

Then there's the far greater challenge the brave attempt, constructing the puzzle without the box. Present day educators find themselves somewhat box-less when confronting a great instructional challenge of our time - teaching computational thinking amidst widely differing visions of just what that concept means.

According to Jeanette Wing, Computational Thinking is:

the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out.

According to the Computer Science Student Teachers Association (CSTA) and International Society for Technology in Education (ISTE), Computational Thinking is:

a problem-solving process that includes (but is not limited to) the following characteristics:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them.
- Logically organizing and analyzing data. Representing data through abstractions such as models and simulations
- Automating solutions through algorithmic thinking (a series of ordered steps) Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
- Generalizing and transferring this problem solving process to a wide variety of problems

According to Google, Computational Thinking is:

Computational thinking (CT) is a problem solving process that includes a number of characteristics, such as logically ordering and analyzing data and creating solutions using a series of ordered steps (or algorithms), and dispositions, such as the ability to confidently deal with complexity and open-ended problems. CT is essential to the development of computer applications, but it can also be used to support problem solving across all disciplines, including math, science, and the humanities. Students who learn CT across the curriculum can begin to see a relationship between subjects as well as between school and life outside of the classroom.

Just what is Computational Thinking?

According to Carnegie Mellon University's Center for Computational Thinking, Computational Thinking is:

a way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science.

There comes a time in a very difficult puzzle journey when, just as we consider calling it all off, a tiny thread reveals itself. Dazzling like a strand of saffron in a perfect dish, a link of similar colors or shapes is spotted and suddenly, we've illuminated the beginnings of an excellent big-picture strategy.

If constructing the coastline is the key to solving a puzzle of a mysterious map, perhaps the common thread of defining the elusive concept of computational thinking is about accurately and comprehensively identifying its undercurrent – the algorithm.

CT's Still the New Student

With its relatively new inclusion in the traditional classroom, the subject of computational thinking (CT) has yet to benefit from a collective perception that often develops over time, or the learning milestones that define proficiency in subjects like math and reading. The knowledge and experience teachers have with these conventionally taught areas, as well as the immense research and subsequent best practices developed from such studies, have manifested very useful rules that predict when a stumbling block in the learning process is a problem. Moreover, this allows teachers the invaluable opportunity to make an intervention at the right time, to ensure educational development continues and students thrive.

These milestones can be as simple as sight words that illustrate reading gains or the deadline of the end of third grade as the time marker for knowing basic multiplication facts. In addition to aiding in the identification of delays in learning, milestones such as "reading grade levels" can help teachers challenge students who are getting a little too comfortable with where they're at in their cognitive development as a reader. With computational thinking occupying more and more space and place in the lives of students and educators, it's time to offer up these same milestones and interventions, at least in an area – algorithms - we can all agree is core to the CT model.

Algorithms in Computational Thinking

By deconstructing complex concepts like the algorithm, we can identify manageable chunks that can then be used to create milestones and scaffold instruction to offer a reliable path to proficiency in computational thinking.

To assist in the scaffolding of the instruction of algorithms, the AP Computer Science Principles course has helped to identify and define the building blocks of algorithms[1]:

- Sequence
- Selection

- Iteration

Sequencing refers to identifying the steps needed to accomplish a task and the correct order of those steps. Both the act of brushing one's teeth, as well as the complicated task of flying a fighter plane require a specific set of steps that must be followed in the correct order to achieve completion.

Selection refers to a condition. These are best understood by considering them as "if" statements – if *this*, then do *that*. A basic example of successful selection would be the learned and understood responses a driver makes to a traffic light. If the light is red, this signals a certain action as appropriate for this circumstance, in this case, to stop. The color of the street light and its accepted meaning is the condition to which a driver will respond to.

Iteration refers to loops. Iteration also uses a condition, but while selection will only check the condition once, with iteration the condition can be checked repeatedly. To continue with the traffic light example, iteration is what causes you to stop at *every* red light, not just the first one you meet. This is very important with robot behavior - as opposed to human – as it illustrates the need for programming a process of continually checking for an understood condition throughout the journey of completing a task.

Establishing the concepts of sequence, selection and iteration as the foundational aspects of the algorithm process is widely considered the first step for computer science and robotics teachers in developing and assessing students' abilities to build and successfully apply algorithms to problems and tasks. Some exciting ways of utilizing the building blocks of sequence, selection and iteration as milestones in the student comprehension of algorithms are emerging in innovative curriculum models from several prominent curriculum developers.

The building blocks of algorithms as milestones in computational thinking development

The Introductory Robotics Curriculum, developed by Carnegie Mellon and Robomatter, has found that by creating a curriculum based on a solid understanding and application of these cornerstones of algorithms, students can gain the thorough understanding of algorithms needed to engage in best practices when creating and applying them. Specifically, this has helped to avoid relying upon unhelpful techniques like "guess and check," a method often turned to by students[2] because of a lack of comprehensive instruction and organized scaffolding between sequence, selection and iteration.

The potential organization of instruction this approach enables is a great start to developing a universal way of teaching algorithms. To reach the goal of student proficiency in applying algorithms, the use of robotics itself is also a powerful teaching tool.

Students are exposed to performing the step of sequencing as they program robots to solve challenges in the Basic Movement unit of the curriculum (see Figure 1), giving educators a chance to put the teaching of algorithms into practice.

When people communicate through their natural, spoken language, meaning is often derived through context and inference. However, a robot cannot understand context and inference, or other nuanced aspects of communication, like looking directly at someone while speaking to them or changing the inflection of your voice for emphasis, both of which are understood by young children and accepted as cultural norms well engrained and understood. Robots of today can only do exactly what it is instructed, and needs those instructions delivered with a level of detail students and teachers alike may find, at first, difficult to grasp. For instance, while we may assume giving the instruction to "stop" at every red light enough to control the flow of traffic, the instructions that create that "stop" are quite numerous – the robot's motor power must be reduced, and the length of the stop must be clearly and explicitly defined, just to name a few. Teachers must prepare themselves for this, the great challenge of sequencing - the ability to break a task down into the smallest part as possible, and then to place those parts in the correct order. Beginning students have a notoriously difficult time overcoming this challenge[3]. Yet with a robot, this task becomes more manageable because the student is given the immediate feedback of the physical actions of the robot. If the robot is supposed to move three rotations forward, but instead move three rotations backwards, actually seeing the robot move provides the student with immediate, and concrete, feedback.

Fig. 1 The Basic Movement unit of the Introduction to Robotics curriculum. Movement and Turning are each broken down into separate chapters, with each chapter concluding in a

culminating challenge. The numbers that follow Moving Forward and Turning represent the lessons that the students receive that emphasize how to sequence their commands while programming. This structure is repeated with every unit in the curriculum.

The sensors used with a robot also provide an authentic and concrete connection for students as they begin to learn selection with the following unit of the curriculum, titled Sensors. One of the programming challenges contained within this unit is called the Dynamic Maze Challenge. In this challenge, students are required to program their robot to move from the starting area through a maze with tall vertical walls. Using the Distance Sensor, the robot has to navigate through the maze without touching any walls. The distance between the walls can change as the students work through different attempts, therefore the use of the sensor is required. This required use of the sensor introduces the students to selection. The input from the sensor is utilized for creating conditions, as it measures the distance between the robot and the wall, which can then be used by the student to create additional programming logic. For example, if the robot is less than 80 millimeters away from the wall, it can be programmed to stop, and then turn. The sensors unit introduces students to a variety of different sensors, thus allowing the students to practice applying the concept of selection to multiple challenges; challenges that gradually increase in complexity as more skills are mastered.

Introducing students to the building blocks of algorithms continues in the next curriculum unit titled Program Flow. Here, students familiarize themselves with the concept of iteration. Again, teachers can utilize the robot to provide context and authenticity to students as they begin to learn this new idea. The universal application of this method of teaching algorithms can be understood as one considers the primary functions of robots in everyday life. We often turn to robots to perform repetitive and monotonous tasks. The curriculum showcases a real-world application of this by showing a video of an autonomous container-handling robot that repetitively moves large numbers of containers. The students are then challenged to use programming logic to create an efficient way for their own robot to move objects that are placed in front of it, a task that relies on the use of iteration for navigation purposes (see Figure 2). Besides grasping and applying algorithms, the curriculum enables students to see they are capable of incredible feats with the use of programming skills and robotics.

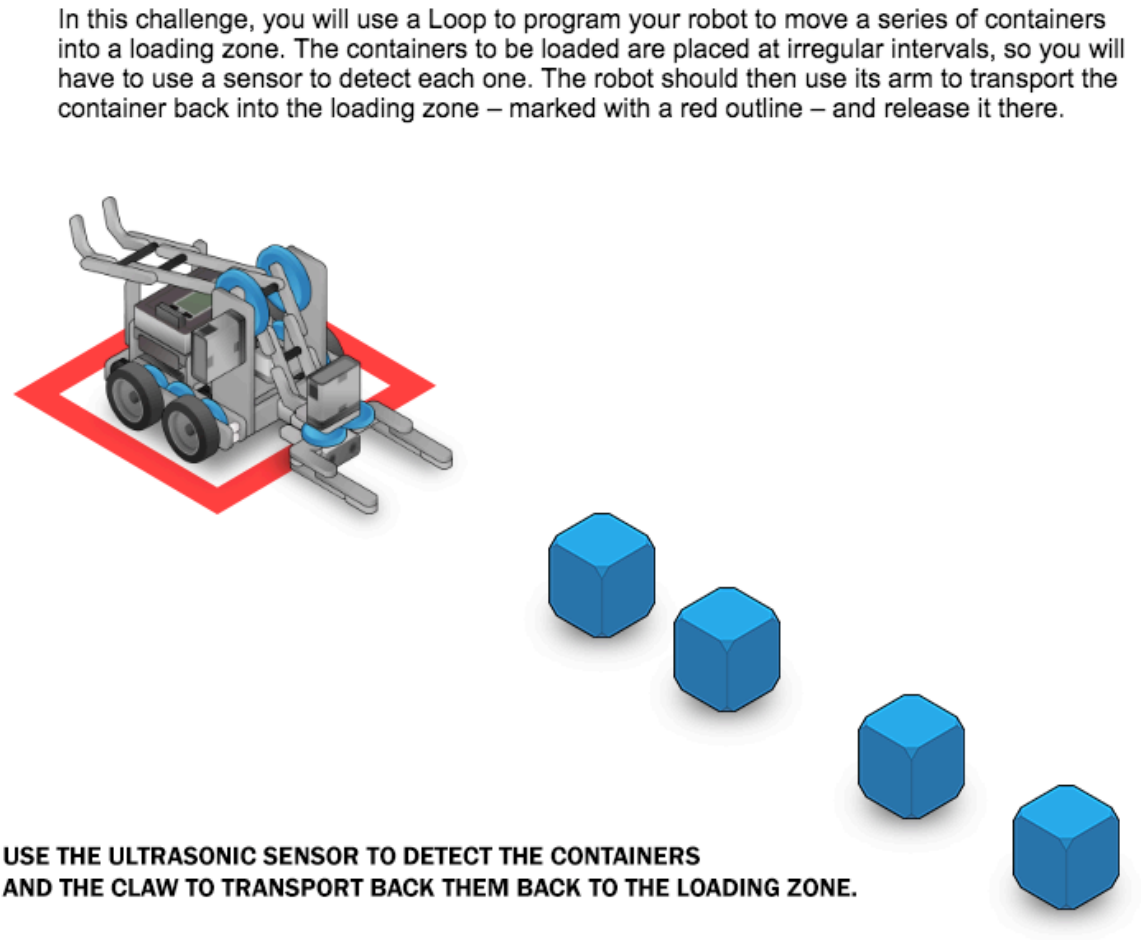

Fig. 2 The student description of the container handling challenge in the Introduction to Robotics curriculum.

To review, the Movement, Sensors and Program Flow units in the Introduction to Robotics curriculum is each devoted to one of the widely-accepted building blocks of creating an algorithm, as identified by the AP Computer Science Principles course. Within each unit, students are provided with worked examples and numerous programming challenges that gradually increase in complexity to scaffold their learning of these concepts, which function as traditional learning milestones. The use of educational robotics also aids in the student learning process for the robots provide authentic examples to learn and apply the ideas while also providing immediate and hands-on feedback to the student.

Other curriculum providers are also creating instructional materials around computational thinking practices. Code.org is a non-profit that both works with schools and conducts teacher training in order to expand access to computer science in schools, while additionally trying to increase participation by women and underrepresented minorities[4]. Currently, Code.org has over 700,000 teachers signed up to teach their introduction courses in computer

science[5]. Code.org also emphasizes learner-centered pedagogy and have found great success in implementing their "Hour of Code" activities to educate teachers, schools and communities about the importance of, and the fun that can be found in computer science.

Code.org offers curriculum for students in elementary, middle and high school. The K-5 curriculum is called CS Fundamentals, and it emphasizes Computational Thinking and Algorithms. The CS Fundamentals course has three different segments. One of the course segments, called Courses A-F, is identified by Code.org as being the preferred pathway. Course A begins with kindergarten students, Course B for 1st graders, and so on (see Figure 3). Each course can be its own entry point. For example, a fifth grade students that has no previous experience can begin with Course F.

The course structure of CS Fundamentals dictates that the students are introduced to computational thinking practices with unplugged activities before students begin working on programmatic solutions. Students then move to "bridging activities," that combine online and unplugged activities. These activities allow students more practice with abstract concepts, like iteration, before they move on to lessons that are completely online. When students begin the online coding activities, the students are asked to solve puzzles that gradually increase in difficulty. As students complete their programming activities, they are encouraged to keep a journal. The purpose of the journal is so the student can write down thoughts, answer questions, use emoticons to communicate to the teacher how they are feeling about an activity, and answer reflection questions. The use of the journals shows that the Code.org curriculum shares the same goals as the Carnegie Mellon Robotics Academy and Robomatter curriculum of moving classroom activities from ones that focus on a surface-level knowledge of computational thinking to a deeper level of understanding as journals provide opportunities for students to reflect on what they have learned so they can explain the content. Receiving the content through some form of direct instruction, and understanding it enough to apply it, is only one aspect of learning it. Explaining the content is a comprehension-deepening step that requires analyzing what has been learned and then communicating it through words. All of this is greatly aided by the use of journals in chronicling the experience of interacting with the material. At the end of the course, students are asked to create an original project that showcases their creativity. These big-picture educational ideas are applied to teaching algorithms by Code.Org and exploring their methods helps elucidate just what is possible in developing a uniform way of teaching the relatively new subject of algorithms.

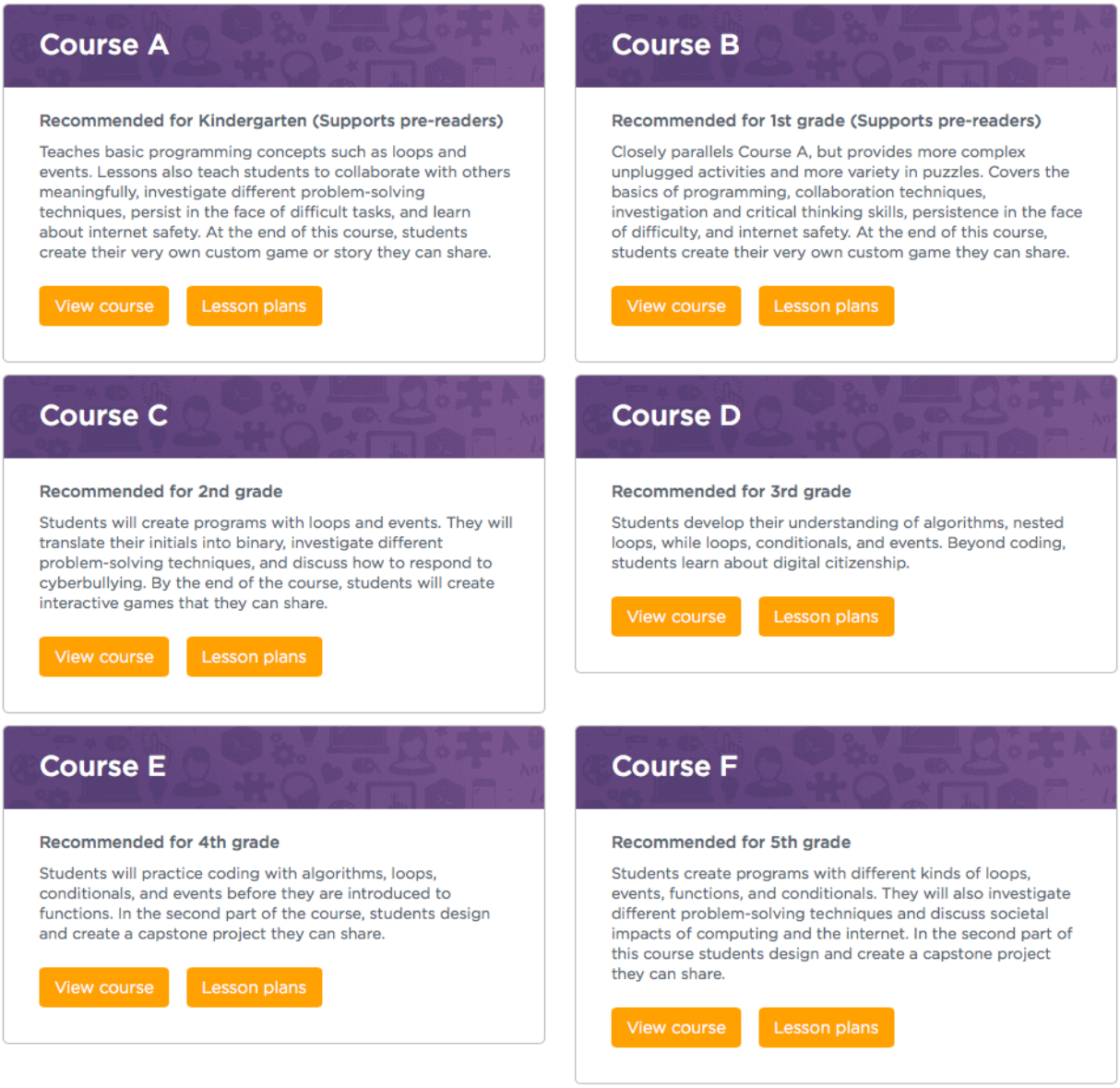

Fig 3. Screenshot of the breakdown of courses A-F in the CS Fundamentals course. Each course contains a description of the learning outcomes contained within it and recommendations of what age level the course is appropriate for.

Course A of the CS Fundamentals course begins the study of algorithms, and, interestingly, introduces the subject with an unplugged activity that relates sequencing to the real-life activity of planting a seed. This analogue is explored in a worksheet given to students. The worksheet includes nine pictures, six of which, illustrate the correct steps needed in planting a seed. The students are tasked with choosing the correct six steps and then placing those steps in the correct order. The students then trade their algorithm with another student or student group, and use each other's directions to plant the seed.

Students get more practice in sequencing in another unplugged activity where they have to use arrows on a map to direct a character to a goal area on the map. This is then followed by an online activity, featuring characters from the game Angry Birds, where students will apply their sequencing skills to get the bird to the pig without crashing into walls or TNT.

After completing the scaffolded activities and lessons around sequencing in Course A, students are next introduced to iteration. Students again begin with an unplugged activity, in this case, "Happy Loops" - another extension of the unplugged map activity. In Happy Loops, the size of the map has increased, but the amount of arrows that they can use has not. This constraint allows students to use the utility of loops when creating an algorithm. After Happy Loops is completed, students next apply the use of Selection in an online programming challenge.

Selection is introduced to students in Course D of CS Fundamentals. Students first complete an unplugged activity, entitled Conditional with Cards, that introduces them to the concept of making a decision within an algorithm and gives them an opportunity to practice iteration. The students are prompted that if a player chooses a particular card (e.g. the queen of hearts), everyone else will perform an action (e.g. clap their hands). If the player does not choose the queen of hearts (e.g. any other card in the deck), then the students would perform a different action (e.g. stomp their feet). This is different than selection as the number of responses has increased beyond a singular reaction. This game introduces students to the basic structure of selection (see Figure 4):

If a condition is met
do something
    else
do something different

Main Activity (20 min)

Conditionals with Cards Sample Program - Teacher Prep Guide

Directions:

- Create a few programs with your class that depend on things like a card's suit, color, or value to award or subtract points. You can write the program as an algorithm, pseudocode, or actual code.

Here is a sample algorithm:

```
if (CARD is RED)
    Award YOUR team 1 point

Else
    Award OTHER team 1 point
```

Here is a sample of the same program in pseudocode:

```
If (card.color == RED){
    points.yours = points.yours + 1;
}

Else {
    points.other = points.other + 1;
}
```

Fig. 4 Screenshot of the teacher lesson plan for teacher conditionals with the unplugged activity, Conditionals with Cards. The lesson plan shows that teacher that the condition is the card's suit and how students can build an algorithm with that information.

Once students have completed the unplugged activity, they can then move on to applying iteration to a programming challenge. Up until this point, when a student has written a program, the program is executed the same way every time the program was run. With selection, this is no longer the case. Similar to the experience of programming a robot with sensors, the algorithm will now function differently depending upon the condition the program encounters.

Both the CS Fundamentals and the Introduction to Robotics curriculum break the teaching of algorithms down into the teaching of sequence, selection and iteration - three established concepts with widely-accepted definitions. It is important that teachers and schools understand this structure so they can provide their students with the same individualized instruction that is strived for in other subjects. Much like offering reading assignments at a student's individual instructional level, teachers can use these three terms as milestones to recognize students struggling to learn sequencing - and in need of extra instructional aids, and students that have mastered the material quickly, and should be provided with an opportunity to create more complex algorithms. This is a great example of how teachers can use these milestones to apply differentiated instruction in their classrooms.

Additionally, understanding the parts of an algorithm will help teachers give feedback and guidance to students as they create more complex programs. Decomposition, the ability to

break down problems into small steps, is a critical part of computational thinking. Understanding the parts of an algorithm will help teachers and students as they decompose problems and build solutions.

Quality curriculum is much more than just the packing of information to facilitate teacher delivery and student consumption. Quality curriculum should create a framework that allows teachers and student to interact at different levels of proficiency and thus create meaningful learning experiences. It is especially helpful, efficient and more comprehensive if these concepts are understood with widely-accepted definitions that can be applied as milestones. In addition to preparing students on what to know and do in a computational world, curriculum has to provide teachers with the necessary tools to reach students at all levels of the learning spectrum.

REFERENCES:

[1] *AP Computer Science Principles*[PDF]. (2016, September). New York: College Board.

[2] Flot, J., Higashi, R., McKenna, J., Shoop, R., Witherspoon, E. (July 2016) *Using Model Eliciting Activities to Engage Students in Computational Thinking Practices* Presented at the High Impact Technology Exchange Conference (2016 HI TEC), Pittsburgh, Pennsylvania

[3] Guzdial, M. (2016). *Learner-centered design of computing education: Research on computing for everyone*. San Rafael, CA: Morgan & Claypool.

[4] About Us. (n.d.). Retrieved March 21, 2018, from https://code.org/about

[5] About Us. (n.d.). Retrieved March 21, 2018, from https://code.org/about