Unit 6: Algorithms

Content Area: **Mathematics**
Course(s): **Generic Course, AP Comp Sci A**
Time Period: **Semester 2**
Length: **3 weeks**
Status: **Published**

Standards

CS.9-12.8.1.12.AP.1	Design algorithms to solve computational problems using a combination of original and existing algorithms.
CS.9-12.8.1.12.AP.2	Create generalized computational solutions using collections instead of repeatedly using simple variables.
CS.9-12.8.1.12.AP.3	Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
CS.9-12.8.1.12.AP.4	Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
CS.9-12.8.1.12.AP.5	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
CS.9-12.8.1.12.AP.6	Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
CS.9-12.8.1.12.AP.7	Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users.
CS.9-12.8.1.12.AP.8	Evaluate and refine computational artifacts to make them more usable and accessible.
CS.9-12.8.1.12.AP.9	Collaboratively document and present design decisions in the development of complex programs.
	Individuals evaluate and select algorithms based on performance, reusability, and ease of implementation.
	Trade-offs related to implementation, readability, and program performance are considered when selecting and combining control structures.
	Complex programs are developed, tested, and analyzed by teams drawing on the members' diverse strengths using a variety of resources, libraries, and tools.
	Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. Modules allow for better management of complex tasks.
	Programmers choose data structures to manage program complexity based on functionality, storage, and performance trade-offs.

Essential Questions

- How can performing a Binary Search save time?
- What makes an algorithm efficient?
- How do we know if a problem is Undecidable?
- How do Parallel and Distributed Computing add to efficiency?

Enduring Understanding

- The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.
- There exist problems that computers cannot solve, and even when a computer can solve a problem, it may not be able to do so in a reasonable amount of time.
- Parallel and distributed computing leverage multiple computers to more quickly solve complex problems or process large data sets.

Knowledge and Skills

Students learn to design and analyze algorithms to understand how they work and why some algorithms are considered more efficient than others. This short unit is entirely unplugged, and features hands-on activities that help students get an intuitive sense of how quickly different algorithms run and the pros and cons of different algorithms. Later in the unit, students explore concepts like undecidable problems and parallel and distributed computing.

Transfer Goals

Not all problems can be solved.

Algorithms have different levels of efficiency.

There are trade offs to making programs "more efficient".

Resources

1. Various YouTube videos that visually explain concepts and ideas.

2. Various widgets found on code.org.

3. Test banks created on Edulastic and code.org

4. Use of Google Classroom, Google Slides, Google Docs and Google Sheets