Exhaustive Search for Nuggets

Colin Fitzpatrick and Ellen Gaffney

Purpose

This lesson is designed to connect several areas of coding for students, introduce a higher level math concept, and look forwards towards algorithmic problem solving by starting with a basic first step known as exhaustive search. For coding, the students will practice basic syntax in defining functions, working with variables, and executing loop statements. For the math concept, students will work with Diophantine equations, or polynomial equations for which each unknown is an integer. Finally, algorithms represent a wide area of computer science; in this lesson, we introduce an exhaustive search algorithm, one of the first typically taught.

Overview

The lesson begins with a juxtaposition of something that is probably unfamiliar to students, Diophantine equations, and something that is probably familiar, Chicken McNuggets. To engage students, Diophantine equations will be introduced and then an example of how they work (ordering chicken McNuggets). The question guiding the lesson is: if you can order nuggets in boxes of 6, 9, and 20, can you place an order for 51 nuggets? We will work through the easiest case, asking students to develop algorithms for solving if an order can be placed if only boxes of 6 are available and using test cases to see if their function works. (This will lead to a discussion of modulo function and/or exhaustive search.) Then, they will work in pairs creating a new function/algorithm for cases where boxes of 6 and 9 and available. (This will lead to a discussion of exhaustive search, as modulo algorithms becomes ineffecient as number box sizes increases.) Finally, they will be left with following up with boxes of 20 for the next lesson.

Student Outcomes

- Students will be able to develop and implement an algorithm for solving math equations.
- Students will be able to create a function and use testing to check if their code works.

Standards Addressed

MS-ETS1-1 - Define the criteria and constraints of a design problem with sufficient precision to ensure a successful solution, taking into account relevant scientific principles and potential impacts on people and the natural environment that may limit possible solutions.

MS-ETS1-2 - Evaluate competing design solutions using a systematic process to determine how well they meet the criteria and constraints of the problem.

MS-ETS1-4 - Develop a model to generate data for iterative testing and modification of a proposed object, tool, or process such that an optimal design can be achieved.

Time

2 class periods

Reach for the Stars is a GK-12 program supported by the National Science Foundation under grant DGE-0948017. However, any opinions, findings, conclusions, and/or recommendations are those of the investigators and do not necessarily reflect the views of the Foundation.

Level

Seventh grade advanced learning program, intro to programming

Materials and Tools

Laptops (1 per two students), one for instructor
Projector, for projecting instructor's laptop
Python 2.7 installed
IDLE (comes with Python) or another text editor (I prefer Sublime Text)
mcNuggets.py file, mcNuggetsSampleAnswer.py file, and MIT open courseware assignment file, for reference. These files can be downloaded here.
(https://northwestern.box.com/s/h55alrzi2yxgmtjl8e7a19qzec5tj29d)

Preparation

To prepare, the instructor should share the mcNuggets.py file with the students. No additional prep is needed.

Prerequisites

This lesson is part of an ongoing unit around intro to programming. Students should already be familiar with variable declaration, print statements, loops, conditionals, and writing functions.

Background

In order to be successful in this lesson, you should know basic programming skills (variable declaration, print statements, loops, conditionals, and writing functions) but you must also be willing to try out ideas. Students who discuss and share ideas with each other have a better chance at implementing a successful solution. That said, use this as an opportunity to practice explaining your process to another student and your reasoning behind it.

Teaching Notes

I implemented the start of this lesson on two sides of a whiteboard. When I asked if a student could describe or has heard of a Diophantine equation on the board, I wrote it up on the left side of the board. Then, I followed up immediately after with asking if they had ever heard of Chicken McNuggets and wrote that up on the right hand side of the board. We then went back and forth between the "high level" concept of the equations, and the "low level" concept of orders of Chicken McNuggets.

As a warm up, I asked them if they could order 36, 47, and 50 nuggets given boxes of 6, 9, and 20. Then, for the students who answered, I asked how they arrived at their answer. As a class, we discussed the things we would need to know in order to answer the question, and the procedure we would follow to come to that answer.

The assignment file is split into three parts, asking students to write functions to solve for whether or not an order can be made given the different boxes that are available. We started as a class and solved the simplest case (a McDonald's just opened and they only have boxes of 6 available). Typically, students will implement this solution using a simple modulo conditional (if the number wanted is divisible by 6 with no remainder, then you can order it). (This is a workable solution throughout all three parts (box of

6; box of 6 and box of 9; and box of 6, 9, and 20) but can become overly convoluted with the conditionals.)
For the second part, with boxes of 6 and 9 nuggets, we had students work in pairs. They were free to adopt the modulo solution we did as a class or pursue other options. After working for ten minutes, we came back together as a class and discussed what worked and what did not. We then worked together on compiling one "class answer" drawing from the work the pairs did.

For the last part, students should either become curious or frustrated with their current strategies, as the modulo solution for boxes of 6, 9, and 20 is complex. I recommend saving the last part for the second day of the lesson and brainstorming alternative strategies for finding solutions (either in pairs or as a class). They should arrive at some version of exhaustive search, which should then be implemented in the first and second part. Students should then complete the third part in their pairs again.

The lesson should close with the "so what" of algorithm design and thinking. One option is to have students pick sides: modulo solution vs. exhaustive search solution, and have they articulate why they prefer that strategy. End the second day of the lesson by previewing other types of algorithms, such as sorting algorithms (bubble sort, quicksort, etc). The takeaway should be that there are multiple strategies to arrive at workable solutions, but that each present decisions around implementation strategy, efficiency, robustness, etc. (For example, exhaustive search is not efficient necessarily, but it is trivial to implement and is quite robust if McDonald's were to add another size box.)

Assessment

Assessment will happen throughout the lesson. While working on the first part together as a class, students will be assessed on their responses and contribution to generating code, including their syntax, problem solving strategy, ability to break down problems into chunks, and ability to communicate their thinking to their peers. For the second and third part, students will be evaluated in their pair work, for having productive conversations, trying out solutions, iterating on strategies for solving the problem, as well as reporting back to the class on their progress/failure. Finally, the files from the pairs will serve as a record for the work that they done.

Additional Information

This lesson was adapted from MIT's open courseware intro to programming class, shared under creative commons licensing (http://ocw.mit.edu/terms/). Their problem set is included in the files.