Lesson 5: Overflow and Rounding

Overview

Students extend their understanding of the binary number system by exploring errors that result from overflow and rounding. They use the binary odometer widget and develop their own systems for representing fractional amounts using the Flippy Do.

Purpose

This lesson introduces students to the practical aspects of using a binary system to represent numbers in a computing device. Students discover the limitations of creating numbers that are “too big” or “too small” to count. They learn that, while a number system is infinite, the physical representation of numbers requires place values -- which are finite, and limit the ability to represent numbers.

Standards

Full Course Alignment

CSP Conceptual Framework

▶ **DAT-1** - The way that the computer represents data is different from the way that the data are interpreted and displayed for the user. Programs are used to translate data into a representation that is more easily understood by people.

CSTA K-12 Computer Science Standards (2017)

▶ **DA** - Data & Analysis

Agenda

Warm Up (5 minutes)

Activity (35 minutes)

Odometer Activity (10 minutes)

More Flippy Do Practice (10 minutes)

Candy Shop Challenge (15 minutes)

Wrap Up (5 minutes)

Assessment

Objectives

Students will be able to:

- Understand that overflow and roundoff errors result from real-world limitations in representing place value.

Preparation

- Work through the Candy Shop exercise yourself so you can see where your students may get stuck or have questions.

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the teachers

- **CSP Unit 1 - Digital Information** - Slides

Teaching Guide

Warm Up (5 minutes)

Discuss: *Imagine you work at a local store. In the register all you have are nine $10 bills, nine $1 bills, and nine dimes.*

- *What's the largest amount of change that you can give someone?*
- *What's the least?*
- *What would you do if someone needed 7 cents in change?*

Discussion Goal: Today we're going to explore what happens when you don't have the right "places" to store information. This can happen when you try to store very large numbers, very small ones, and everything in-between! The goal of the prompt is to understand the very real problem of making sure that enough place values are available to represent numbers.

- The largest amount of change you can give someone is $99.90.
- The least change you can give is $0.10.
- You can't give someone $0.07. You also can't give someone $1.25 in change (because you have no nickels!)

At both extremes of the number range, large and small -- and in-between numbers -- you are unable to build some numbers because you don't have the place values to do so.

Activity (35 minutes)

Group: Place students in groups of two. Groups will need one computer per group.

Odometer Activity (10 minutes)

Remarks

> We will start exploring large place values to see what happens when a big number gets too big. Go to the Binary Odometer Widget on Level 2 of the lesson on Code Studio.This is a widget that simulates a car odometer - a device that tracks how far the car has driven (in miles or kilometers). Explore the odometer to understand how it works.

Teaching Tip

Students will tackle the problem of "running out of place values" when counting to bigger and bigger numbers. They will play with a digital odometer widget to explore what happens when you add one to the largest number you can represent in a number system.

Do This: Have students navigate to Level 1 and playwith the odometer to figure out how it works.

Circulate: Give students two to three minutes to explore the odometer.

1 **Widget: Binary Odometer**

Remarks

> Now I'm going to give you a challenge to tackle with the Binary Odometer. Write your response in your journals.

Do This: Instruct students to set the binary odometer to the highest number possible. Then let it run!

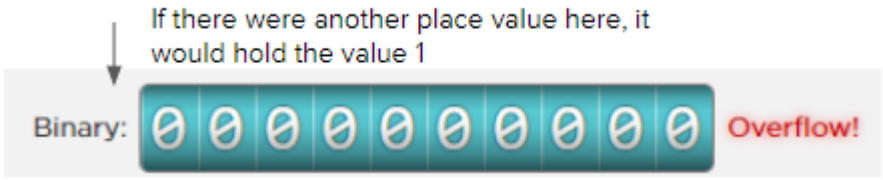
Discuss:

- *What happens to the odometer reading?*
- *Does the odometer still show the distance driven by the car?*

Discussion Goal: When the odometer turns over -- exceeds the number of place values it can physically display -- the odometer reading correctly shows all but one number in its place value positions.

For example:

After the odometer turns over:



As you can see, all the numbers should flip to 0 and the one of the left should flip to 1. But we are out of place values! So the numbers flip to 0, but nothing flips to 1. The number we are trying to represent does not fit in the number of bits we have available to accurately display it.

Like a bathtub that has reached its maximum capacity for holding water and is now overflowing, the odometer has reached its maximum capacity and has reached **overflow**.

More Flippy Do Practice (10 minutes)

Remarks

> Now let's look at the limits of our Flippy Do when it comes to overflow.

Discuss:

- *What value would cause your Flippy Do to overflow?*
- *What adaptation could you make to the Flippy Do to represent that value?*
- *Using your newly adapted Flippy Do, how many total numbers can be represented?*

Discussion Goal: Use this conversation to show that we can always add bits to represent more numbers but when working with a set number of bits, at some point we will run into overflow if we try to represent large enough values.

- Look for students to physically model overflow using the Flippy Do, going from when all of the 1s are flipped up to the next number causing all the 1s to reset to 0s.
- With the Flippy Do, trying to represent the number 256 will cause an overflow. One common adaptation students will come up with is adding an additional column/bit to the Flippy Do to be able to represent that number. For a 9-bit Flippy Do, the first number that will cause overflow is 512.

Teaching Tip

This would be a good time in the lesson to address any areas you noticed your students struggling with in the previous lesson when using their Flippy Dos. You may choose to review certain questions from the activity guide based on the responses students gave or develop your own practice questions.

Candy Shop Challenge (15 minutes)

Remarks

> Now that you have started thinking about place value and overflow, we are going to work on a different problem. What happens when there aren't enough place values to represent a number?

Have students work in pairs on setting the four prices for their candy shop.

Discuss:

- *How did you go about deciding which binary number to use for each price?*
- *In your system, who pays the difference between the true price and the value stored in your computer system: the shop owner or the customer?*

Discussion Goal: Use this discussion to highlight that the Flippy Do doesn't have the precision needed to represent these fractional values and in that case, decisions must be made on how to represent those values. In this case, those decisions impact who ends up benefitting when it comes to the price of each candy. You could ask you students to compare different systems.

Teaching Tip

Many students may use traditional rounding to decide on the prices for each of the candies. Some may come up with a system where the shop pays the difference and others may come up with a system where the customer pays the difference. See the table below for how those values would be represented in binary.

Candy	Price	Pure Rounding	Shop Pays	Customer Pays
Gummy Bears	$1.76/lb	**0010**	**0001**	**0010**
Chocolate	$4.16/lb	**0100**	**0100**	**0101**
Licorice	$7.52/lb	**1000**	**0111**	**1000**
Mints	$0.48/lb	**0000**	**0000**	**0001**

Discuss: *What problems could come from having both of these two types of scales in single one of your stores?*

Discussion Goal: Use this discussion to highlight the importance of precision. The customers and shop owners are depending on a given amount of candy always having the same price. Computers rely on precision. Think of a calculator. We depend on 1+1 always equaling 2. Rounding is necessary because of the limits of bits, but can lead to errors.

Wrap Up (5 minutes)

Review the information on the first wrapup slide showing how binary can be used to precisely represent some fractional values from the decimal number system. Students do not need to understand how to convert fractional amounts from decimal to binary.

Teaching Tip

For your own personal knowledge, you can read more about "decimal points" and "binary points" on **Wikipedia**.

Remarks

> The most important takeaway from this lesson is to understand that bits can represent a limited amount of information.

Discuss:

- *What does the binary odometer show about representing large numbers?*
- *If we had a big enough odometer or Flippy Do, could we represent every possible number?*

Discussion Goal: As students have seen, you can make “any number” in your head, but the tool you use to represent a number has limitations. It has a fixed number of place values it can show. The odometer keeps running after you move beyond its upper limit, but the largest place values cannot be displayed due to overflow error.

Journal: Students add to their journals the definitions for: Overflow Error and Round-off Error.

Assessment

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: Modern car odometers record up to a million miles driven. What happens to the odometer reading when a car drives beyond its maximum reading?

Question: When using bits to represent fractions of a number, can you create all possible fractions? Why or why not?

2-3 **Check For Understanding**

2 3