Contents

Interdisciplinary Mathematics and Computer Science

A Pedagogical Approach to Teaching Both Fields

Derek Neilson

September 12, 2024

Chapter 1

Preface

1.1 Preface

This paper is a collection of thoughts and ideas that have been developed over the past few years. The concepts presented here are the result of my experiences as a computer science student. Although I have always been interested in mathematics, it wasn't until I approached the subject from a computer science perspective that I fully appreciated its beauty. This paper is an effort to share some of the insights I have gained through this interdisciplinary lens.

1.1.1 Acknowledgements

This work was completed independently, but I am deeply grateful to Weber State University for providing the resources and opportunities that have enabled me to pursue my interests. I would also like to thank my family and friends for their support and encouragement. Finally, I extend my gratitude to the authors of the textbooks and papers I have referenced throughout this work. Their contributions have been invaluable in helping me develop my own ideas.

1.1.2 Abstract

This paper explores the relationship between mathematics and computer science, focusing on the ways in which that students can benefit from an interdisciplinary approach to these fields. The paper begins by comparing elementary mathematical concepts to their computer science counterparts and beginner programming concepts to their mathematical equivalents. It then discusses the benefits of teaching mathematics and computer science together, including the development of problem-solving skills, the promotion of creativity, and the enhancement of students' understanding of both subjects. As the paper progresses, it delves into more advanced topics, such as the connections between discrete mathematics and computer science, the role of algorithms in both fields, and the ways in which computer science can be used to solve complex mathematical

problems. The paper concludes by emphasizing the importance of interdisciplinary education and encouraging educators to incorporate elements of both mathematics and computer science into their curricula. The goal of this paper is to inspire students and educators alike to explore the connections between these two disciplines and to appreciate the beauty and power of mathematics and computer science when studied together.

1.1.3 Figures

List of Figures

1.1.4 Tables

List of Tables

1.1.5 Abbreviations

The following abbreviations are used throughout this paper:

Computer Science Abbreviations

CS & Computer Science
DS & Data Structures
OOP & Object-Oriented Programming
AI & Artificial Intelligence
DBMS & Database Management System
ML & Machine Learning
OS & Operating System
API & Application Programming Interface

Mathematics Abbreviations

DM & Discrete Mathematics
P & C & Probability and Combinatorics
S & L & Sets and Logic
T & F & Truth and Falsity
FFT & Fast Fourier Transform
LCM & Least Common Multiple
GCD & Greatest Common Divisor
PDF & Probability Density Function

General Abbreviations

STEM & Science, Technology, Engineering, and Mathematics
UI & User Interface
UX & User Experience
BFS & Breadth-First Search
DFS & Depth-First Search
RSA & Rivest-Shamir-Adleman (encryption algorithm)

Chapter 2

Introduction

2.1 Introduction

Computer Science (**CS**) and mathematics are two closely related disciplines that share concepts and techniques. Both fields focus on the study of abstract structures and the development of algorithms for manipulating these structures. Despite their close relationship, the two disciplines have traditionally been taught separately, with minimal emphasis on the connections between them. This paper explores the deep interrelationship between **CS** and mathematics, highlighting the ways in which students can benefit from a more integrated, interdisciplinary approach to these fields.

The paper begins by comparing elementary mathematical concepts to their **CS** counterparts and beginner programming concepts to their mathematical equivalents. This comparison serves to illustrate the similarities and differences between the two disciplines and to lay the groundwork for a more in-depth exploration of their connections. Python code snippets are included throughout the paper to provide concrete examples of the concepts being discussed. These code snippets are intended to be accessible to readers with little to no programming experience. Each code snippet is accompanied by a brief explanation of the code and its relevance to the topic at hand. The Math examples are thoroughly explained and are intended to be accessible to readers with little to no mathematical experience. As the paper progresses, it delves into more advanced topics. Some topics covered may be challenging for readers without a strong background in mathematics or **CS**, but the paper is designed to be accessible to a wide audience.

Chapter 3

Mathematical and Programming Fundamentals

3.1 Introduction

In this section, we will explore the fundamental concepts of mathematics and programming that serve as the building blocks for more advanced topics in both fields. We will cover topics such as place value, variables, arithmetic operators, and the order of operations. These concepts are essential for understanding the deep connections between mathematics and computer science and will provide a solid foundation for the rest of the paper.

3.2 Place Value

3.2.1 Place Value and Number Systems

A number system is a way of representing numbers. The most common number system is the decimal system, which is also known as the base-10 system. The base-10 system uses ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. The value of a digit in a number depends on its position in the number. For example, in the number 123, the digit 3 is in the ones place, the digit 2 is in the tens place, and the digit 1 is in the hundreds place. The value of the number is calculated by multiplying each digit by its place value and adding the results together. In this case, the value of the number 123 is $1 \times 100 + 2 \times 10 + 3 \times 1 = 100 + 20 + 3 = 123$.

$$\begin{aligned}123 &= 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0\\ &= 1 \times 100 + 2 \times 10 + 3 \times 1\\ &= 100 + 20 + 3 = 123\end{aligned}$$

3.3 Variables

3.3.1 What is a Variable?

A variable is a named storage location in a computer's memory that holds a value. The value of a variable can change during the execution of a program. Variables are used to store data that is used by the program, such as numbers, strings, and objects. In Python, variables are created by assigning a value to a name using the assignment operator, which is the equals sign (=). For example, the following code creates a variable named x and assigns it the value 10:

```
# Create a variable named x and assign it the
   value 10
x = 10
```

In this code, the variable `x` is created and assigned the value `10`. You will notice that there is some text following a hash symbol (#) this is a comment in Python. Comments are used to explain the code and are ignored by the Python interpreter. Comments are useful for documenting code and making it easier to understand.

3.3.2 Mathematical Variables

In mathematics, variables are used to represent unknown values or values that can change. Variables are typically denoted by letters, such as x, y, and z. For example, in the equation $y = 2x + 3$, x and y are variables. The value of y depends on the value of x. If x is 1, then y is 5. If x is 2, then y is 7. In this way, variables in mathematics are similar to variables in programming. They both represent values that can change.

3.3.3 Commonalities and Differences

Variables in mathematics and programming share some commonalities. They both represent values that can change, and they are both used to store data. However, there are some key differences between variables in mathematics and programming. In mathematics, variables are used to represent unknown values or values that can change, while in programming, variables are used to store data that is used by the program. Additionally, variables in programming have a specific type, such as integer, float, or string, while variables in mathematics are more abstract and can represent any type of value.

3.4 Arithmetic-operators

3.4.1 Addition (+) and Subtraction (-)

Addition and subtraction are fundamental arithmetic operations used in both mathematics and programming.

Addition

In mathematics, addition is represented by the plus sign (+), and it combines two numbers to produce a sum. For example, $2 + 3 = 5$. Similarly, in programming, addition is also represented by the plus sign. In Python, this would be written as `2 + 3 = 5`. Here's a simple code snippet that demonstrates addition in Python:

```python
# Addition
x = 2
y = 3
print(x + y)  # Output: 5
```

In this code, the variables `x` and `y` are assigned the values `2` and `3`, respectively. The sum of `x` and `y` is then computed and printed to the console. The `print` function outputs the result of the expression `x + y`, which is `5`.

Subtraction

Subtraction, on the other hand, is represented by the minus sign (-) and is used to find the difference between two numbers. In mathematics, for example, $5 - 3 = 2$. In Python, subtraction is represented in the same way, as shown below:

```python
# Subtraction
x = 5
y = 3
print(x - y)  # Output: 2
```

In this code, the variables `x` and `y` are assigned the values `5` and `3`, respectively. The difference between `x` and `y` is calculated and printed as `2`.

In both cases, whether it's addition or subtraction, the operations in Python mirror those in traditional mathematics, making it easy for programmers to apply their knowledge of arithmetic directly in code.

3.4.2 Multiplication (*) and Division (/)

Multiplication

Multiplication is another fundamental arithmetic operation that is used in both mathematics and programming. In mathematics, multiplication is represented by the asterisk (*) symbol, and it is used to find the product of two numbers. For example, $2 \times 3 = 6$. Similarly, in programming, multiplication is also represented by the asterisk symbol. Here's an example of multiplication in Python:

```python
# Multiplication
x = 2
y = 3
print(x * y)  # Output: 6
```

In this code, the variables `x` and `y` are assigned the values `2` and `3`, respectively. The product of `x` and `y` is then computed and printed to the console. The `print` function outputs the result of the expression `x * y`, which is `6`.

Division

Division is represented by the forward slash (/) symbol and is used to find the quotient of two numbers. In mathematics, for example, $6/3 = 2$. In Python, division is more nuanced due to the different types of division that can be performed. Here's an example of classic division in Python:

```python
# Division
x = 5
y = 2
print(x / y)  # Output: 2.5
```

In this code, it is important to note that the division operation `x / y` results in a floating-point number, such as `2.5`, rather than an integer. A floating-point number includes a decimal point, and Python automatically performs floating-point division when the result is not an integer.

In mathematics, dividing two integers may result in a fraction or a decimal, and Python mirrors this behavior by default. For example, dividing `5 / 2` in Python will yield `2.5`.

This behavior is different in some other programming languages, such as C or Java, where dividing two integers can result in an integer due to integer division. For instance, in C or Java, the operation `5 / 2` would result in `2`, effectively discarding the fractional part. This difference is important to keep in mind, as it can lead to unexpected results if you're accustomed to integer division in these languages.

3.4.3 The Modulo Operator (%) and Floor Division (//)

Modulo Operator (%)

The modulo operator (%) is another arithmetic operator that is used in both mathematics and programming. In mathematics, the modulo operator is used to find the remainder of the division of two numbers. For example, $5\%2 = 1$, because when 5 is divided by 2, the remainder is 1. In Python, the modulo operator is also represented by the percent sign (%). Here's an example of the modulo operator in Python:

```python
# Modulo
x = 5
y = 2
print(x % y)  # Output: 1
```

In math this is written as $5 \mod 2 = 1$. The modulo operator is particularly useful in programming for tasks such as determining whether a number is even or odd, or for iterating over a sequence of numbers.

Floor Division (//)

Floor division (//) is another division operator in Python that is used to find the quotient of two numbers, rounded down to the nearest integer. For example, $5//2 = 2$, because the result of dividing 5 by 2 is 2.5, which is then rounded down to 2. Here's an example of floor division in Python:

```python
    # Floor Division
    x = 5
    y = 2
    print(x // y)  # Output: 2
```

Floor division is useful when you want to divide two numbers and obtain an integer result, rather than a floating-point number. In Python, the operator `//` performs floor division. The mathematical equivalent of floor division is dividing two numbers and rounding down to the nearest integer, represented as:

$$\lfloor x \div y \rfloor$$

For example, the result of `5 // 2` is `2`, which corresponds to rounding down the result of `5 / 2` (which is `2.5`) to the nearest integer.

3.4.4 Ceiling Division

Ceiling division is the opposite of floor division. It rounds up to the nearest integer instead of rounding down. In Python, there is no built-in operator for ceiling division, but you can achieve the same result using the math module. Here's an example of ceiling division in Python:

```python
# Ceiling Division
import math
x = 5
y = 2
print(math.ceil(x / y))  # Output: 3
```

In this code, the math module is imported to access the `ceil` function, which rounds a number up to the nearest integer. Notice that there is a dot operator (`.`) used to access the `ceil` function from the math module. The result of `math.ceil(x / y)` is `3`, which corresponds to rounding up the result of `5 / 2` (which is `2.5`) to the nearest integer. We will discuss modules and functions in more detail in later sections for now just know that the math module is a built-in module in Python that provides mathematical functions and constants. The `.` operator is used to access functions and constants within a module.

Mathematically, ceiling division is represented as:

$$\lceil x \div y \rceil$$

3.5 Order of Operations

3.5.1 What is the Order of Operations?

The order of operations, also known as precedence rules, is a set of rules that determines the order in which mathematical expressions should be evaluated. The order of operations ensures that expressions are evaluated in a consistent and unambiguous manner. The order of operations is as follows:

1. Parentheses
2. Exponents
3. Multiplication and Division
4. Addition and Subtraction
5. Left to Right

3.5.2 CS and Math Notation

Parentheses

In both computer science and mathematics, parentheses are used to group expressions and indicate the order in which operations should be performed. Expressions inside parentheses are evaluated first. For example, in the expression $2 \times (3 + 4)$, the addition inside the parentheses is evaluated first, resulting in $2 \times 7 = 14$.

Exponents

Exponents are used to raise a number to a power. In mathematics, exponents are denoted by the caret symbol (ˆ) or by superscript notation. For example, 2^3 represents 2 raised to the power of 3, which is $2 \times 2 \times 2 = 8$. In Python, the double asterisk (`**`) operator is used for exponentiation. For example, `2 ** 3` represents 2 raised to the power of 3, which is `8`.

Multiplication and Division

Multiplication and division have the same precedence level and are evaluated from left to right. For example, in the expression $2 + 3 \times 4$, the multiplication is performed first, resulting in $2 + 12 = 14$. In Python, the multiplication operator (`*`) and the division operator (`/`) have the same precedence level and are evaluated from left to right.

Addition and Subtraction

Addition and subtraction also have the same precedence level and are evaluated from left to right. For example, in the expression $2 + 3 - 4$, the addition is performed first, resulting in $5 - 4 = 1$. In Python, the addition operator (`+`) and the subtraction operator (`-`) have the same precedence level and are evaluated from left to right.

Left to Right

If two operators have the same precedence level, they are evaluated from left to right. For example, in the expression $2 + 3 - 4$, the addition is performed first because addition and subtraction have the same precedence level, and the expression is evaluated from left to right.

3.5.3 Conclusion of Mathematical and Programming Fundamentals

In this section, we have explored the fundamental concepts of mathematics and programming, including place value, variables, arithmetic operators, and the order of operations. We now have a solid foundation in these concepts, which will serve as the building blocks for more advanced topics further in the paper. The examples and explanations provided in this section are intended to familiarize readers with the format and syntax of the paper, as well as to introduce key concepts that will be referenced throughout the paper. The next section will delve into more advanced topics, building on the knowledge gained here to explore the deep connections between mathematics and computer science.