UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 2

Algorithm Analysis and Flowchart

Submitted by:
Adoracion, Jerick Dave D.

Instructor:
Engr. Maria Rizette H. Sayo

August 01, 2025

I. Objectives

Introduction

Data structure is a systematic way of organizing and accessing data, and an algorithm is a step-by-step procedure for performing some task in a finite amount of time. These concepts are central to computing, but to be able to classify some data structures and algorithms as "good," we must have precise ways of analyzing them.

This laboratory activity aims to implement the principles and techniques in:

- Writing a well-structured procedure in programming
- Writing algorithm that best suits to solve computing problems to improve the efficiency of computers
- Convert algorithms into flowcharting symbols

II. Methods

A. Explain algorithm and flowchart

B. Write algorithm to find the result of equation: $f(x) = \begin{cases} -x, & x<0 \\ x, & x \geq 0 \end{cases}$ and draw its flowchart

C. Write a short recursive Python function that finds the minimum and maximum values in a sequence without using any loops

A. Explain algorithm and flowchart

An **algorithm** is a step-by-step procedure or formula for solving a specific problem or performing a task. It is written in a way that can be understood by humans and eventually translated into a computer program.

Characteristics of an Algorithm

- **Finiteness**: It must terminate after a finite number of steps.
- **Definiteness**: Each step must be clearly and unambiguously defined.
- **Input**: It should have zero or more inputs.
- **Output**: It must produce at least one output.
- **Effectiveness**: Every step must be basic enough to be carried out.

Flowchart is a graphical representation of an algorithm. It uses various symbols to denote different types of instructions and arrows to show the flow of control from one step to the next.

Common Flowchart Symbols are:

- **Rectangle -** Process step (e.g., calculations or assignments)
- **Paralelogram** - Input/Output operations
- **Diamond** - Decision-making (e.g., Yes/No questions)
- **Oval** - Start/End

III. Results

B. Write algorithm to find the result of equation: f (x) = and draw its flowchart

Algorithm:

1. Start
2. Read Value of x
3. Check if $x < 0$
4. If YES, set $f = -x$
5. If NO, set $f= x$
6. Display value of f
7. End

Flowchart:

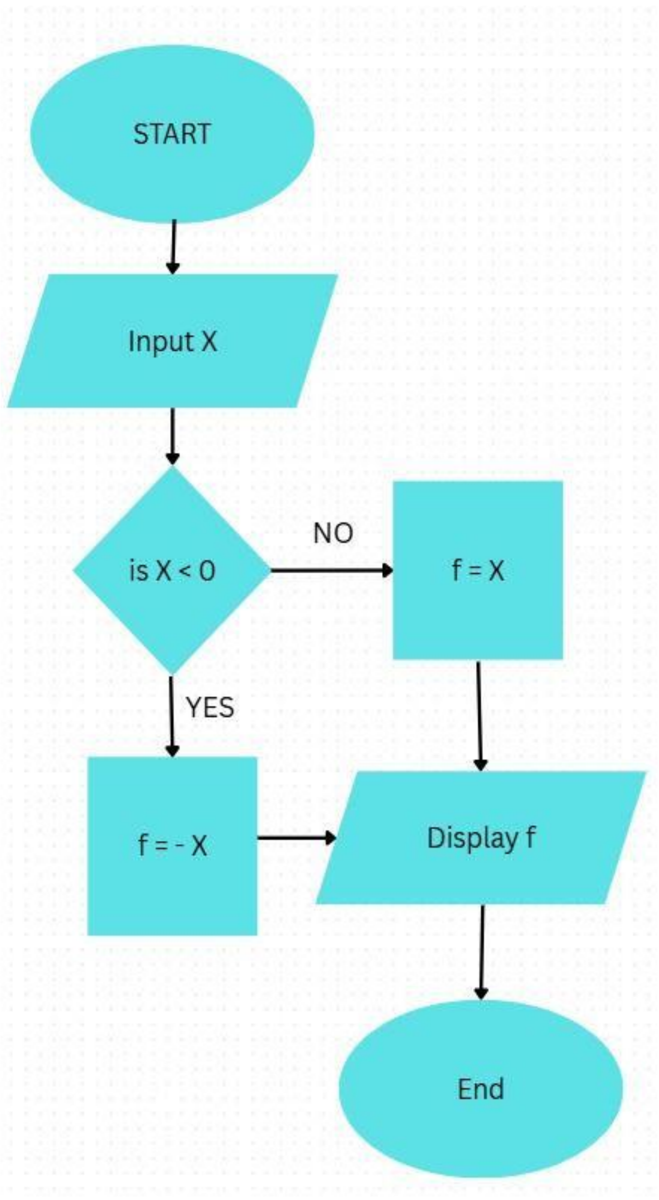
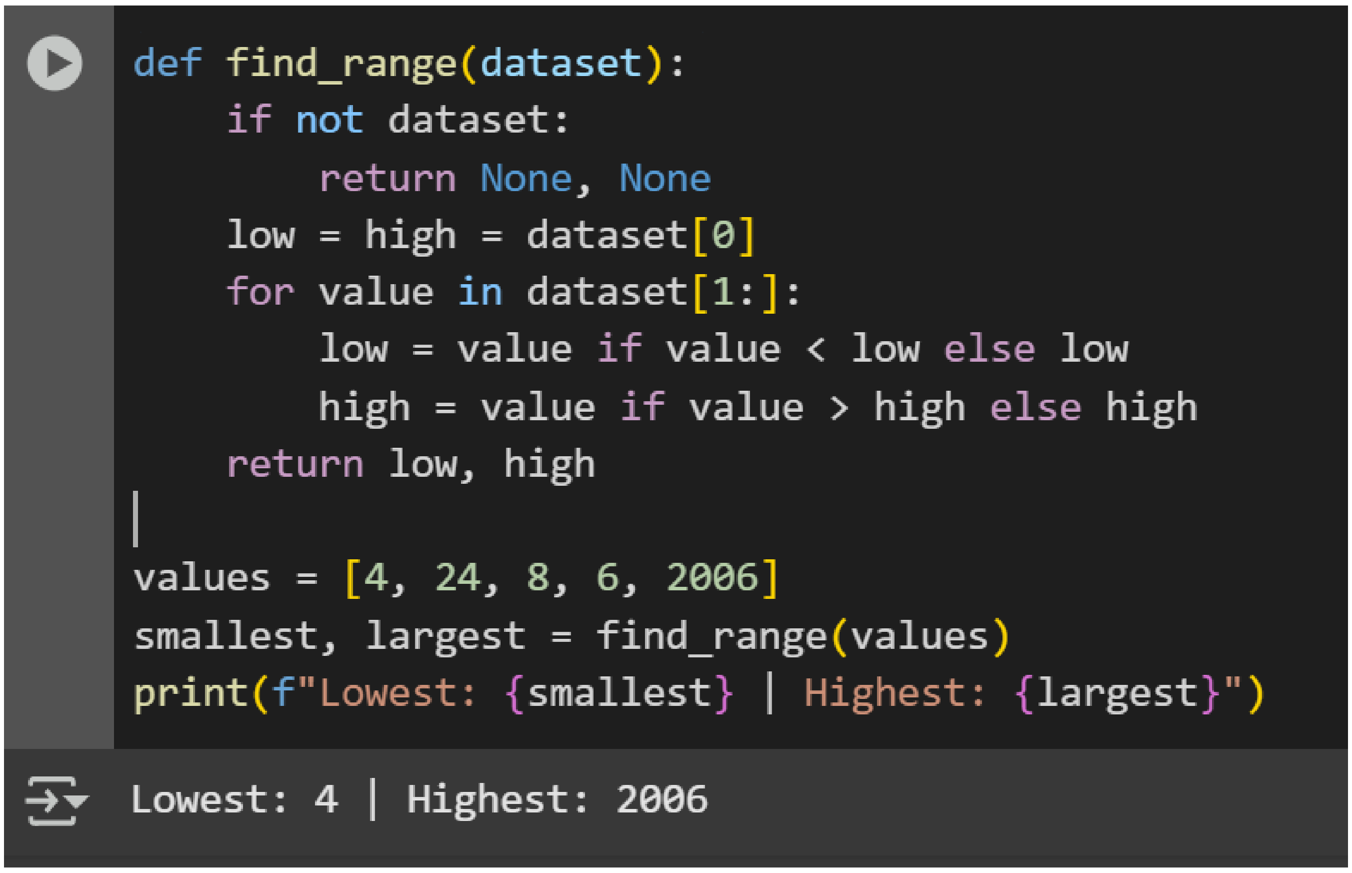


Figure 1: Screenshot of the flowchart of the program

This flowchart represents a simple program that calculates the absolute value of a number entered by the user. The process begins with a start point, followed by a step where the user is prompted to input a value for X. Once the input is received, the program evaluates whether the value of X is less than zero. If X is indeed less than zero, the program assigns f the value of -X, effectively converting the negative number to a positive one. If X is greater than or equal to zero, the program simply assigns f the same value as X. After determining the appropriate value of f, the program proceeds to display this result. Finally, the process ends. In essence, this flowchart outlines the logic for determining and displaying the absolute value of a number using a straightforward decision-making structure.



```python
def find_range(dataset):
    if not dataset:
        return None, None
    low = high = dataset[0]
    for value in dataset[1:]:
        low = value if value < low else low
        high = value if value > high else high
    return low, high

values = [4, 24, 8, 6, 2006]
smallest, largest = find_range(values)
print(f"Lowest: {smallest} | Highest: {largest}")
```

```
Lowest: 4 | Highest: 2006
```

Figure 2: Source code and Output

This Python script defines a find_range() function that efficiently identifies the minimum and maximum values in a dataset by first checking for empty input (returning None, None if true), then initializing tracking variables with the first element before iterating through the remaining values to update the lowest and highest found. When tested with [4, 24, 8, 6, 2006], it correctly outputs *"Lowest: 4 | Highest: 2006"* through a single-pass algorithm (O(n) time complexity) that compares each value against current extremes using ternary operators, offering optimal performance for data analysis tasks while maintaining clean, readable code with proper error handling and clear output formatting

IV. Conclusion

At first, I was struggling to write flowcharts during our PLD course as I am still adjusting to the new environment around me. But through this laboratory activity, I gained a deeper understanding of how algorithms and flowcharts assist in solving problems in a step-by-step manner. Writing procedures and converting them into flowcharts helped clarify the logic and made it easier to follow each step. I found the section on recursive functions particularly interesting, as it demonstrated how certain problems can be solved without using loops. Overall, this lab enhanced my problem-solving skills in programming and emphasized the importance of proper planning before writing actual code

References

“What is a programming algorithm? Data defined - indicative,” Indicative, Sep. 15, 2021. https://www.indicative.com/resource/programming-algorithm/

S. S. Bhamare, S. S. Patil and M. D. Patil, "Understanding Algorithms and Flowcharts: A Comparative Study," *International Journal of Computer Applications*, vol. 176, no. 23, pp. 1–5, June 2020. [Online]. Available: https://www.ijcaonline.org/archives/volume176/number23/31283-2020920164

A. Silberschatz, P. B. Galvin, and G. Gagne, Operating System Concepts, 9th ed. Hoboken, NJ: Wiley, 2018.