



UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT



Data Structure and Algorithm

Laboratory Activity No. 2

Algorithm Analysis and Flowchart

Submitted by:
Catahan, Joshua A.

Instructor:
Engr. Maria Rizette H. Sayo

8, 4, 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

III. Results

A. The difference between algorithm and flowchart lies within their purpose and functions. Algorithm is a set of procedure done in order to arrive at the expected output. While flowchart is a graphical diagram that uses shapes and lines in order to demonstrate the flow of data. This process of creating a flowchart is called “flowcharting” [1].

Below are figures that will help us visualize how algorithm and flowchart works [2].

Example 3: Determine Whether A Student Passed the Exam or Not:

Algorithm:

- Step 1: Input grades of 4 courses M1, M2, M3 and M4,
- Step 2: Calculate the average grade with formula "Grade=(M1+M2+M3+M4)/4"
- Step 3: If the average grade is less than 60, print "FAIL", else print "PASS".

Figure 1: Algorithm example [2]

As we can see in figure 1, algorithm is just a text-based step by step process of the program. It is a simple procedure done in order to understand how the program should print the output based on the given input.

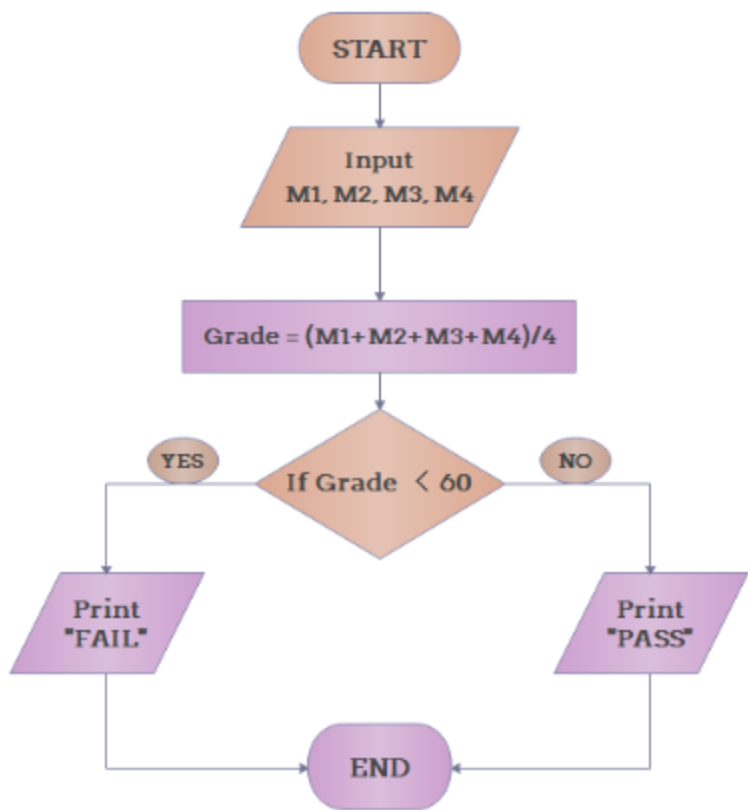


Figure 2: Flowchart example [2]

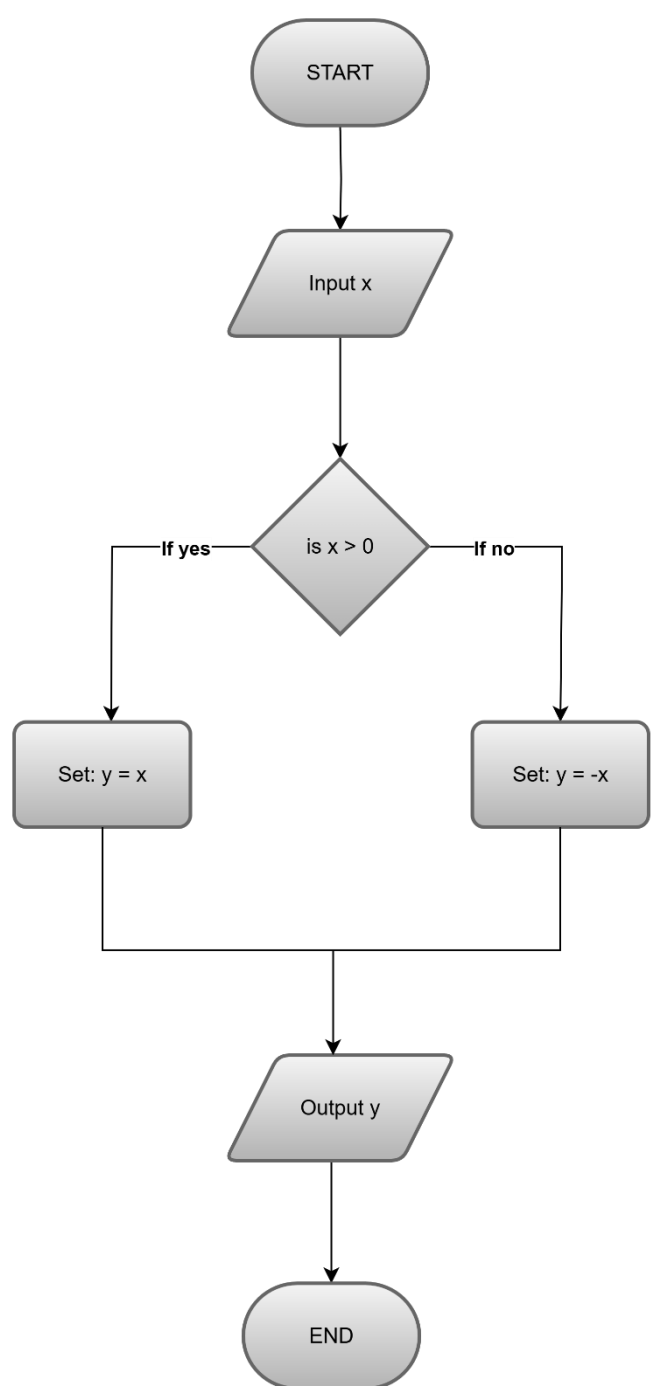
While in figure 2, we can see that shapes and lines are the main components in the flowchart. These shapes/symbols have their own meanings assigned to them for proper use. Just like an algorithm, a flowchart has the same goal, albeit it is more specific and complex so that the programmer can follow through it while coding.

B.

ALGORITHM:

- Step 1: Start
- Step 2: Input a value of x
- Step 3: Check: Is $x \geq 0$?
 - If yes, then set $y = x$
 - If no, then set $y = -x$
- Step 4: Output the value of y
- Step 5: End

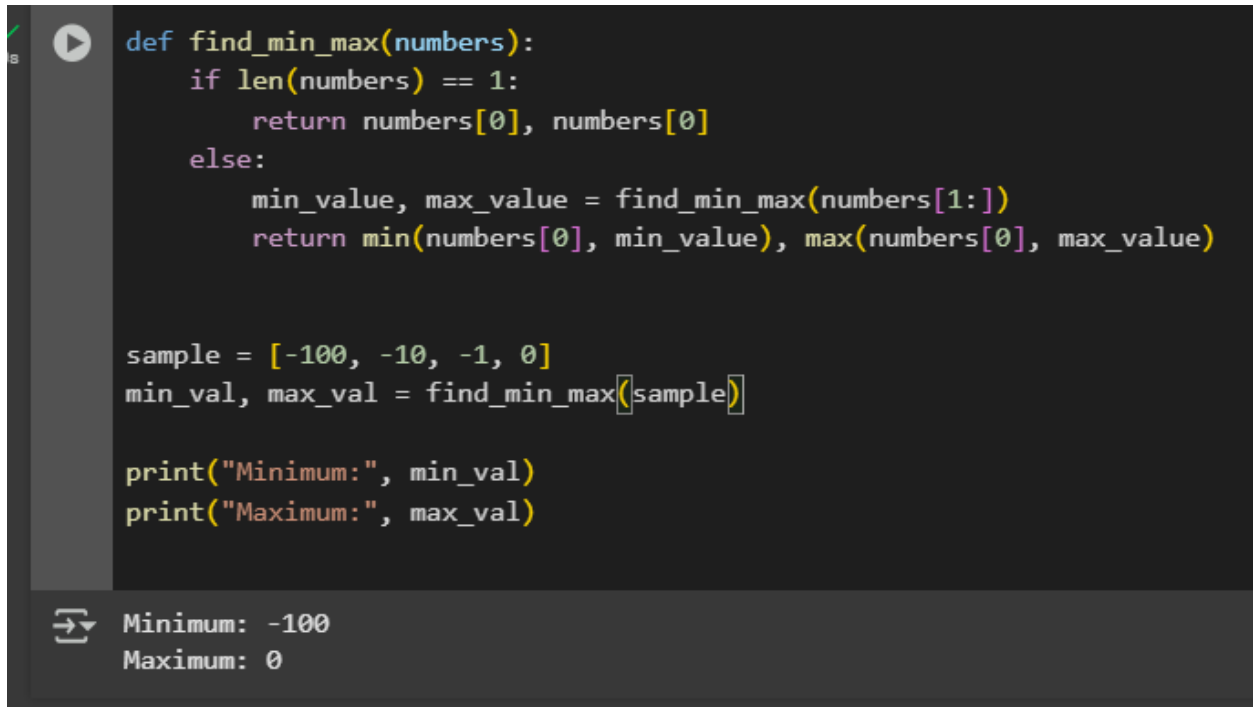
FLOWCHART:



COMMENTARIES (B):

- We can observe that whatever is written in the algorithm is also included in the flowchart. The only difference between them is the shapes/symbols and lines that connect the procedure in the flowchart. In this flowchart, the user will first enter the value of x. Then, it will check if $x > 0$; if yes, it will set $y = x$, and if no, it will set $y = -x$. What it does is simply print the absolute value of the number. If the input number is -1, it will be set as $y = -x$, which will lead to a positive number if the value of x is negative.

C.

A screenshot of a Python IDE with a dark background. The code defines a recursive function `find_min_max` that takes a list `numbers` and returns its minimum and maximum values. The function uses a base case for a single-element list and a recursive case for lists with more than one element. Below the function, a sample list `[-100, -10, -1, 0]` is used to test the function, and the results are printed. The output at the bottom shows the minimum as -100 and the maximum as 0.

```
def find_min_max(numbers):  
    if len(numbers) == 1:  
        return numbers[0], numbers[0]  
    else:  
        min_value, max_value = find_min_max(numbers[1:])  
        return min(numbers[0], min_value), max(numbers[0], max_value)  
  
sample = [-100, -10, -1, 0]  
min_val, max_val = find_min_max(sample)  
  
print("Minimum:", min_val)  
print("Maximum:", max_val)
```

Minimum: -100
Maximum: 0

COMMENTARIES (C):

As we can observe above, this code uses recursion to find the minimum and maximum values in a list without using loops. It starts by checking if the list has only one item. If yes, it returns that number as both the min and max. If not, it keeps calling itself using the rest of the list.

Once it reaches the last number, it compares each previous number one by one using `min()` and `max()` until it gets the final result. In the example `[-100, -10, -1, 0]`, it ends up printing -100 as the minimum and 0 as the maximum. What this code does is simply break the problem into smaller parts, solve it from the bottom, and combine the results going back up.

IV. Conclusion

In this activity, I explored the difference between algorithm and flowchart, where the algorithm shows a step-by-step process using text, while the flowchart uses shapes and lines to represent the flow of data. I also saw how both can work together to explain a simple program, such as finding the absolute value of a number. Lastly, I applied what I learned by creating a recursive Python function that finds the minimum and maximum values in a list without using any loops. This shows how algorithmic thinking and flowcharting help me understand and solve problems in programming.

References

- [1] GeeksforGeeks, “Difference between Algorithm and Flowchart,” *GeeksforGeeks*, [Online]. Available: <https://www.geeksforgeeks.org/dsa/difference-between-algorithm-and-flowchart/>. [Accessed: Jul. 30, 2025].
- [2] Wondershare Edraw, “What is an Algorithm and Flowchart?”, *EdrawMax*, [Online]. Available: <https://edraw.wondershare.com/explain-algorithm-flowchart.html>. [Accessed: Jul. 30, 2025].