Data Structure and Algorithm

Laboratory Activity No. 2

Algorithm Analysis and Flowchart

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Hermosura, Leigh B. | Engr. Maria Rizette H. Sayo |

August, 14, 2025

# 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

# Methods

* 1. Explain algorithm and flowchart

-x, x<0

x, x ≥ 0

* 1. Write algorithm to find the result of equation: f (x) = and draw its flowchart
  2. Write a short recursive Python function that finds the minimum and maximum values in a sequence without using any loops

# Results

The diagrams, images, and explanation below present the outcome yielded by the laboratory activity.

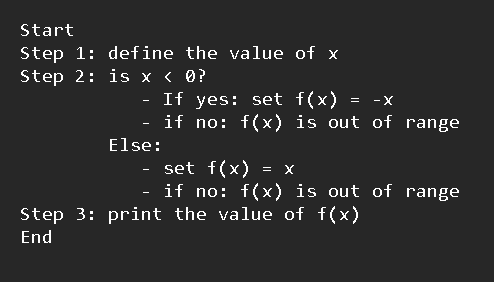
An *algorithm* is a sequence of mathematical operations designed to solve a given problem. In order to gain a deeper understanding of algorithms, they are often represented into visual diagrams called *flowcharts* [1]. These *flowcharts* provide a step-by-step visualization of the program's logic, showing how it progresses from start to finish.

Figure 1 Algorithm of Method B

The illustration above shows the *algorithm* of the piecewise function in Method B where we first define the value of *x*. The program then checks if *x* is greater than 0. If the condition is true, it is set to the range of the first piecewise function, *f(x) = -x*. If not, it is then set to the range of the second piecewise function *f(x) = x*. Lastly, the program prints the value of *f(x)* and finishes the program. The corresponding *flowchart* of the *algorithm* is provided below.

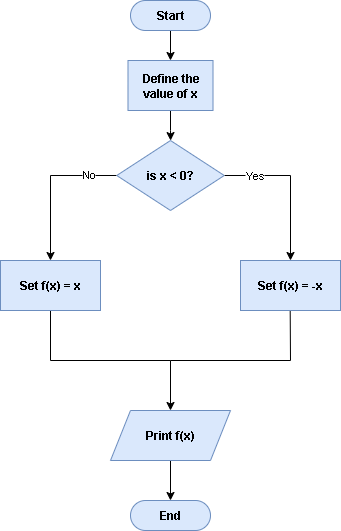


Figure 2 Flowchart of Method B

The final part of the laboratory experiment features a program that determines the minimum and maximum values in a given sequence of numbers without the use of looping. The following is the image of the program.



Figure 3 Image of Method C’s Python program

The code is composed of three parts: a function that identifies the minimum and maximum values in a list of numbers, a random number generator that creates a list of five numbers between 0 and 1000, and lastly, the output display. Inside *find\_minmax()* is a recursive function that finds the minimum and maximum value in a list without looping. It works by comparing the first element of the list with the minimum and maximum values obtained from the rest of the list. The function returns the minimum and maximum values, which are then assigned into the variables *highest* and *lowest* which is then printed. A sample output is provided below.

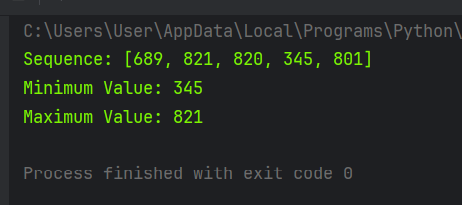


Figure 4 Sample output of the program

# Conclusion

This laboratory activity provided a hands-on exploration of algorithm design and implementation, emphasizing both logical flow and practical coding techniques. Through the use of flowcharts, we were able to visually interpret how algorithms function step by step, particularly in the context of a piecewise function. Additionally, the implementation of a recursive function to determine the minimum and maximum values in a number sequence demonstrated how problems can be solved without relying on traditional looping structures.

**References**

[1] A. B. Chaudhuri, \*Flowchart and Algorithm Basics: The Art of Programming\*, Dulles, VA: Mercury Learning and Information, July 2020, 188 pp. [Online]. Available: https://books.google.com.ph/books?id =JJYJEAAAQBAJ