Data Structure and Algorithm

Laboratory Activity No. 2

Algorithm Analysis and Flowchart

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Nerio, Hannah Grace A. | Engr. Maria Rizette H. Sayo |

July, 26, 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

≥ 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

* 1. Explain algorithm and flowchart

An **ALGORITHM** is like a recipe—it’s a step-by-step guide that tells you how to solve a problem or do a task. You write it down using short instructions in a logical order. For example, if you’re making a peanut butter sandwich, your algorithm might say: get bread, spread peanut butter, put the slices together. In computing, algorithms help tell the computer what to do, like sorting numbers or searching for information.

A **FLOWCHART**, on the other hand, is a drawing that shows these steps visually. It uses boxes and arrows to show how the process moves from one step to another. Different shapes mean different things—like ovals for the start or end, diamonds for decisions (like yes or no), and rectangles for actions. Flowcharts make it easier to understand and check the steps before you write any code.

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

**ALGORITHM**

**Start**

**Input value of x**

**If x < 0 then**

1. **Set f(x) = -x**

**Else**

1. **Set f(x) = x**

**Display f(x)**

**End**

**FLOWCHART**

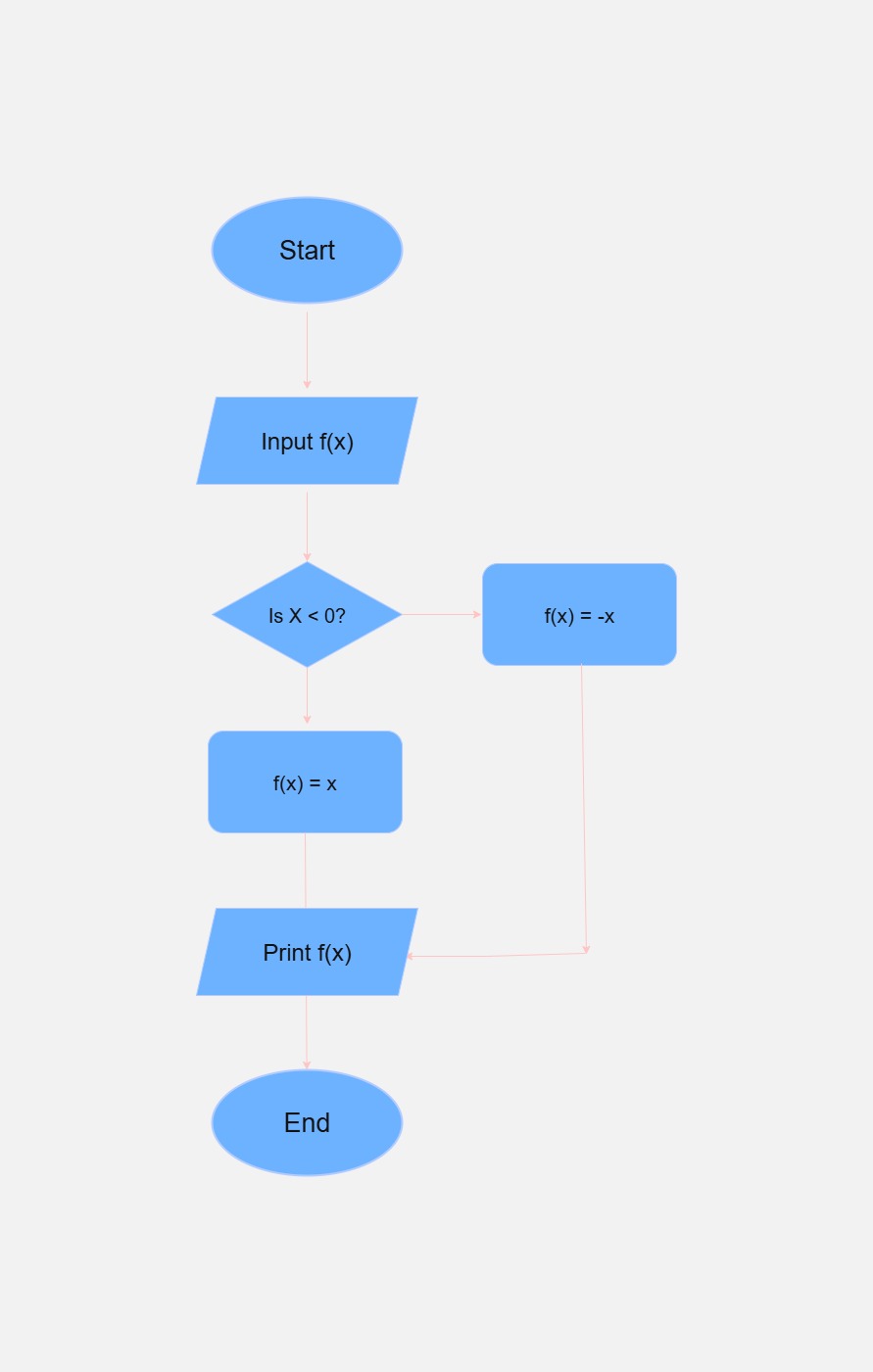
****

Figure 1. Flowchart

This flowchart shows the process of a program that calculates the value of the function f(x) depending on the input. It starts by asking the user to enter a value for x. Then, it checks if x is less than 0. If this condition is true, the program sets f(x) to -x, turning it into a positive value. If the condition is false (meaning x is 0 or more), it sets f(x) to x as is. This is a simple way to show how to get the absolute value of a number using a conditional check. The flowchart clearly presents the logic step by step, from input to decision, and then to the correct output based on the condition.

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

A computer screen shot of a program code

AI-generated content may be incorrect.

Figure 2. Screenshot of the program

This recursive Python program is designed to find the minimum and maximum values in a list without using any loops like for or while. Instead, it uses recursion, which means the function calls itself with a smaller part of the list each time. If the list has only one number left, it returns that number as both the minimum and maximum. Then, as the function goes back through each step, it compares the current number with the previous results to update the minimum and maximum values. This program is a good example of how recursion can be used to solve problems that would normally need loops.

# Conclusion

This laboratory activity demonstrated key principles of algorithm analysis and visual logic representation through flowcharts. The process clarified how structured procedures can enhance computational efficiency and accuracy. By implementing a conditional function and a recursive algorithm for determining minimum and maximum values, the activity highlighted practical applications of both decision-making constructs and recursion. The outcomes reinforce the importance of precise logic and systematic planning in programming tasks.

**References**

[1] Co Arthur O.. “University of Caloocan City Computer Engineering Department Honor Code,” UCC-CpE Departmental Policies, 2020.

[2] diagrams.net, "App.diagrams.net – Online Diagram Software," https://app.diagrams.net [Accessed: Aug. 2, 2025].

[3] Google Colab, "Untitled notebook," https://colab.research.google.com/drive/16WioaYP91QkmSV73zI\_4odOyUP-Usnli [Accessed: Aug. 2, 2025].