Data Structure and Algorithm

Laboratory Activity No. 2

Algorithm Analysis and Flowchart

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Elpedes, Glen Jorge 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, 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

Present the visualized procedures done. Also present the results with corresponding data visualizations such as graphs, charts, tables, or image. Please provide insights, commentaries, or explanations regarding the data. If an explanation requires the support of literature such as academic journals, books, magazines, reports, or web articles please cite and reference them using the IEEE format.

Please take note of the styles on the style ribbon as these would serve as the style format of this laboratory report. The body style is Times New Roman size 12, line spacing: 1.5. Body text should be in Justified alignment, while captions should be center aligned. Images should be readable and include captions. Please refer to the sample below:

Algorithm

An algorithm is a set of step-by-step instructions used to solve a specific problem or complete a particular task. It tells you exactly what to do, in what order, to achieve the desired outcome.

Flowchart

A flowchart is a diagram that represents an algorithm or a process using different shapes connected by arrows. Each shape has a specific meaning and helps visualize the flow of steps clearly.

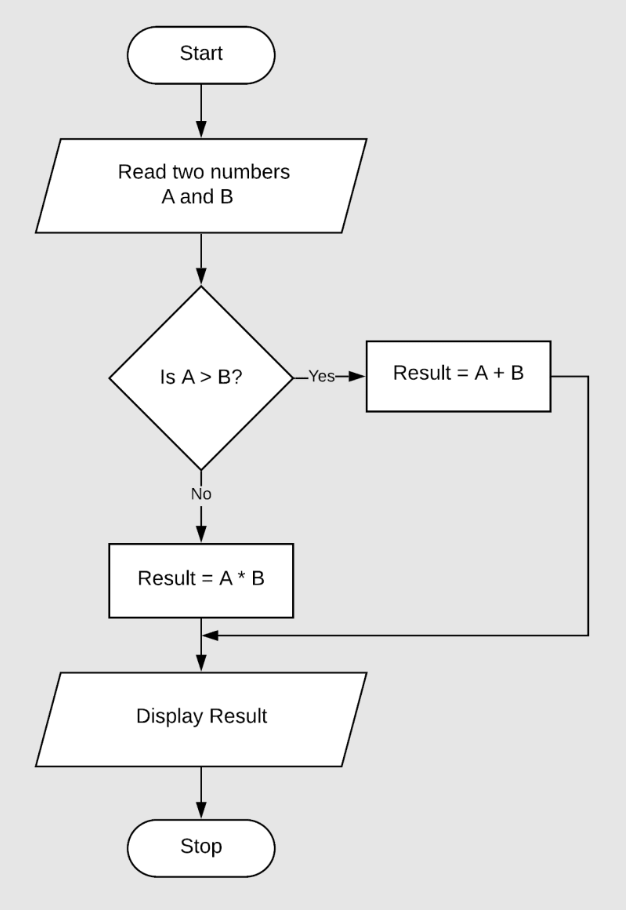


Figure 1. flowchart

This flow shows the results of two number if the number “A” is greater than number “B” it will add the two numbers. And if the number "A” is less than it will multiply the two numbers then stop.

**B.**

Algorithm

**Step 1:** Start

**Step 2:** Input of value of x

**Step 3:** If x < 0, then set f = -x

**Step 4:** Else, set f = x

**Step 5:** Output

**Step 6:** Stop

Flowchart

1.Start

2.Input the value of x

3.x < 0?

4.If yes, head to 6

5.If no, head to 7

6.f = -x

7.f = x

8.Output of f

9.Stop

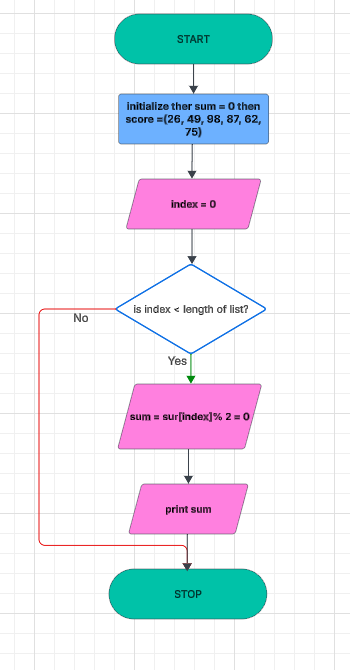


Figure 2. Flow Chart

This figure shows the flow chart of process shows the result value of “f” where if it is less than zero print f = -x, and if not print f = x and stop the program.

C.

def find\_min\_max\_helper(seq, index, current\_min, current\_max):

    if index == len(seq):

        return current\_min, current\_max

    current\_min = min(current\_min, seq[index])

    current\_max = max(current\_max, seq[index])

    return find\_min\_max\_helper(seq, index + 1, current\_min, current\_max)

def find\_min\_max(seq):

    if not seq:

        raise ValueError("Sequence is empty")

    return find\_min\_max\_helper(seq, 1, seq[0], seq[0])

numbers = [10, 30, 20, 9, 7, 35, 11]

minimum, maximum = find\_min\_max(numbers)

print(f"Min: {minimum}")

print(f"Max: {maximum}")

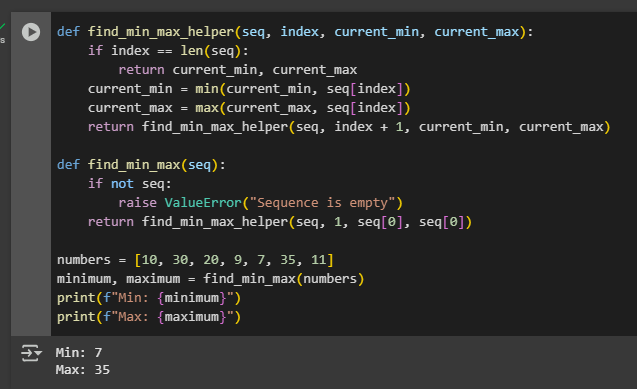


Figure 3. Source Code

This source code shows us this program looks through a list of numbers and figures out which one is the smallest and which is the biggest. Instead of using loops, it uses a helper function that checks each number one by one using recursion. It starts with the first number and keeps comparing as it moves through the list. In the end, it prints the lowest and highest numbers it found just like how you'd scan a list of grades to find the best and worst scores.

# Conclusion

This version of the code still finds the smallest and largest numbers in a list, just like the original but it does things a bit smarter. Instead of breaking the list into smaller pieces every time (which uses more memory), it keeps track of where it is using an index and updates the minimum and maximum as it goes. This makes the function more efficient and better for longer lists. In simple terms, it still gets the job done, just in a cleaner and more thoughtful way.

**References**

<https://www.bing.com/newtabredir?url=https%3A%2F%2Fwww.rechargecolorado.org%2Falgorithm-and-flowchart-for-addition-of-two-numbers-in-excel%2F>