Explore flowchart and pseudocode concepts in algorithms and programming

Iin Tarsini*, Department of Computer and Network Engineering, Politeknik Siber Cerdika Internasional, Cirebon, West Java, Indonesia
Riska Anggraeni, Department of Digital Business, Politeknik Siber Cerdika Internasional, Cirebon, West Java, Indonesia

*Email for Correspondence: iin_tarsini@poltek.ac.id riskaan662@gmail.com

Keywords: flowchart, pseudocode, algorithm, programming

ABSTRACT

Programming algorithms are a class of mathematical algorithms that are widely used in computer science and engineering. They are used to solve problems involving mathematical calculations, but in their implementation, algorithms do not always handle complex calculations, especially when genetic algorithms are often used. This article uses the method of literature study or literature review to evaluate several studies related to the topic discussed, especially in introducing the basic concepts of programming algorithms, especially for beginners, focusing on the basic understanding required. The research includes a review of algorithmic concepts, algorithm notation, characteristics, properties, and the basic structure of algorithm learning, along with the steps of implementing programming algorithms. The conclusion of this article is that learning basic programming algorithms is an essential necessity for anyone who wants to get started in the world of programming.

INTRODUCTION

An algorithm is an efficient method described as a finite set of steps (Davari et al., 2021; Xia et al., 2022; Zhang et al., 2021). An algorithm is also a set of instructions for solving a particular problem, and those instructions can be interpreted step by step from beginning to end (Akgun & Greenhow, 2022; Blazewicz et al., 2013; Karimi-Mamaghan et al., 2022). The problems that the algorithm solves can vary, provided that each problem has initial conditions that must be met before running the algorithm (Elgeldawi et al., 2021; Schubert & Rousseeuw, 2021). Algorithms also involve repeating the process (iterating) and making decisions until the problem is solved.

In this field, algorithms are studied abstractly regardless of the computer system or programming language used. Different algorithms can be applied to problems that have the same criteria. The complexity of an algorithm measures the extent of computation required by the algorithm to solve its problem (Abualigah & Diabat, 2021; Ikotun et al., 2023; Tilly et al., 2022). Informally, algorithms that can solve problems quickly have a low level of complexity, while algorithms that take longer have a higher level of complexity.

The term “algorithm” comes from the word “algorithm,” which was first introduced by Al-Khwarizmi, a scientist of Persian origin. Initially, algorithms were used to solve arithmetic problems, but over time, algorithms were widely used to solve various mathematical problems. An algorithm is a clearly defined set of operations, where each operation requires a finite amount of memory and time to complete (Lee et al., 2021; Madsen et al., 2022; Sebastian et al., 2020). Algorithms also have a significant relationship with the field of mathematics. An algorithm is a special approach used to solve a problem. Based on this definition, an algorithm is a systematic approach that involves a sequence of steps used to solve problems and make decisions.

Algorithms in programming are closely related to mathematical calculations, but in their implementation, algorithms do not always handle complex calculations, especially when genetic algorithms are often used to solve problems involving mathematics. There are several considerations to consider when using algorithms. First, the algorithm must be correct, meaning that the output of the algorithm must match the input instructions entered. If the algorithm is entered with errors, the output will also be imprecise. Second, the level of accuracy of the algorithm, especially the degree to which the algorithm approaches the actual result. Third, the efficiency of the algorithm, which can be assessed from two aspects, namely storage capacity and time.

This is an open access article under the CC BY-SA license.
Although an algorithm gives very accurate results, its use is ignored because it takes a very long time to produce output. Therefore, people prefer to use algorithms with faster output. When considering the amount of memory, if an algorithm uses too much memory, then the quality of the algorithm is not good. Each individual can use algorithms in a different order, the most important thing is that the result remains the same. In information technology, a program is an application used to execute tasks and solve certain problems, such as applications for document processing, graphic design, calculations, and so on. However, if defined in detail, a program is a set of computer instructions written in a programming language.

Program and Programming Language are two inseparable concepts. Program refers to the set of instructions (Buchanan et al., 2008; Kim & Smith, 2002), while Programming Language refers to the standard instructions used in the creation of programs (Gabrielli & Martini, 2023; Vieira et al., 2020). A program is a series of coded instructions that can be interpreted by a computer to solve a problem or achieve a desired result. A Programming Language is a formal language that consists of a collection of instructions for a computer that produce output. Programming Languages are used in personal computer programming to implement algorithms. The use of programming languages allows programmers to specifically choose what data will be processed by the computer. Although there are many programming languages, there is only one programming language that can be understood and executed by all personal computers, which is the original binary code consisting of "0" and "1".

In solving a problem, algorithms involve stages that are arranged systematically and logically. Algorithms have five main characteristics, namely finiteness, definiteness, input, output, and effectiveness. Programming languages can be grouped based on their skill level, namely low-level, intermediate-level, and high-level programming languages.

Skills in computer programming are becoming increasingly essential in the modern era. Along with the rapid advancement of information technology, a basic understanding of programming algorithms is the main key to understanding and creating innovative software (Chen, 2017; Shafto et al., 2010; Webb et al., 2017). In an era where computing has become an indispensable part of almost all aspects of daily life, the ability to design, develop and understand software has become crucial. Learning programming is a very important first step to understanding the basic principles of computer science. By studying programming, one can understand the concepts of algorithms, programming logic, and basic data structures. This article aims to serve as a basic introduction to programming algorithms, especially for beginners, focusing on the basic understanding required.

To explain algorithms in a structured and understandable way, especially for a programmer who is responsible for implementing or executing a program, flowcharts become very important. Apart from the use of flowcharts, in writing algorithms to solve a problem, another alternative that can be used is pseudo-code. "Pseudocode" comes from a combination of the words "pseudo" and "code", which means pseudocode resembles or mimics real program code using a specific programming language.

Based on the background above, this study aims to analyze and explore the concepts of flowcharts and pseudocode in the context of algorithms and programming. The research can hopefully contribute to a deeper insight into the issue discussed and become a reference for future research bringing up a similar topic.

METHOD

In this study, the author uses the method of literature study or literature review to evaluate several studies related to the topic discussed, especially in introducing the basic concepts of programming algorithms. A number of reference sources that are the main focus come from books, journals, magazines, and various other sources. The research includes a review of algorithmic concepts, algorithm notation, characteristics, properties, and the basic structure of algorithm learning, along with the steps of implementing programming algorithms. The following is the flow or steps of the data collection and processing method carried out:

![Figure 1. Research Flow](image-url)
RESULTS AND DISCUSSION

Flowchart

A flowchart can be described as a series of steps for solving a problem represented by special symbols. This flowchart illustrates the logic of the program flow. Flowcharts are not only used as a means of communication, but also as a guide. Before the components can be better understood, the rules of org chart design need to be communicated. The design principles for organizational diagrams are as follows:

a. The flowchart arrangement is depicted in orientation from top to bottom and from left to right.
b. Each activity or process in the graphic of the organization should be described unequivocally and without ambiguity.
c. Each flowchart starts with a starting state or starting point and ends with one or more terminal, end, or page states.
d. Take advantage of Page State connectors and Page Exit connectors with uniform labels to indicate disconnections between algorithms, such as those caused by page changes or transitions.

To illustrate the steps or solutions of a problem in a simple, understandable, structured, and ambiguity-free way, the use of some symbols that are considered standard becomes the purpose of the flowchart. Here are the symbols and their functions that are often used to describe algorithms in flowchart format:

<table>
<thead>
<tr>
<th>No.</th>
<th>Symbol</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1" alt="Terminal" /></td>
<td>&quot;Terminal&quot;</td>
<td>The beginning or end of a program (Procedure).</td>
</tr>
<tr>
<td>2</td>
<td><img src="image2" alt="Output/Input" /></td>
<td>&quot;Output/Input&quot;</td>
<td>The input or output process is independent of the type of device.</td>
</tr>
<tr>
<td>3</td>
<td><img src="image3" alt="Process" /></td>
<td>&quot;Process&quot;</td>
<td>Computer operational processes.</td>
</tr>
<tr>
<td>4</td>
<td><img src="image4" alt="Decision" /></td>
<td>&quot;Decision&quot;</td>
<td>To indicate that a particular condition leads to two Likely, yes/no.</td>
</tr>
<tr>
<td>5</td>
<td><img src="image5" alt="Connector" /></td>
<td>&quot;Connector&quot;</td>
<td>Connect a connecting process to another process on the same page.</td>
</tr>
<tr>
<td>6</td>
<td><img src="image6" alt="Offline Connector" /></td>
<td>&quot;Offline Connector&quot;</td>
<td>Connection Connecting from one process to another on another page.</td>
</tr>
<tr>
<td>7</td>
<td><img src="image7" alt="Predefined Process" /></td>
<td>&quot;Predefined Process&quot;</td>
<td>Represents the terms of storage for processing to provide an initial price.</td>
</tr>
<tr>
<td>8</td>
<td><img src="image8" alt="Punched Card" /></td>
<td>&quot;Punched Card&quot;</td>
<td>The input comes from the card or the output is written to the card.</td>
</tr>
<tr>
<td>9</td>
<td><img src="image9" alt="Punch Tape" /></td>
<td>&quot;Punch Tape&quot;</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td><img src="image10" alt="Document" /></td>
<td>&quot;Document&quot;</td>
<td>Print the output in document format (via printer).</td>
</tr>
<tr>
<td>11</td>
<td><img src="image11" alt="Flow" /></td>
<td>&quot;Flow&quot;</td>
<td>Declare the course of the flow of a process.</td>
</tr>
</tbody>
</table>

Table 1. Symbol and Function Algorithm
The following is presented an example of a flowchart by determining a number is a leap year or not:

Input: year in numeric form (integer)
Output: displays the sentence "Leap Year" if the number is entered represents leap years, and the sentence "not Year leap) otherwise

Figure 2. Flowchart Example

**Pseudocode**

Pseudocode is a method of writing algorithms that mimic the style of high-level programming languages. In general, pseudocode notation uses a language that is easier to understand and concise when compared to programming languages. The content of the pseudocode is a description of computer programming algorithms that utilize the basic elements of several programming languages, but the language is only intended to be readable and understood by humans. Therefore, pseudocode has no ability to be understood by computers.

Pseudocode notation must be translated into the syntax of a particular programming language in order to be understandable by computers. There are no official rules binding pseudocode notation, but it is recommended to use commonly used keywords, such as if, then, else, while, do, for, repeat, and so on. Pseudocode writing guidelines include several parts, namely having a header or head section that includes titles, comments, and algorithm declarations. Next, there is the algorithm body and the end of the algorithm that shows the algorithm cover. Pseudocode can also have comments that start with the character "{" and end with the character "}".

The following Pseudocode example is presented as follows:
When compared to natural language algorithms, pseudocode has a tendency to become easier to understand. On the other hand, natural language algorithms are more practical to use because they are easier to manage relationships between blocks and easier to understand.

Algoritma

An algorithm refers to the way and sequence of steps or stages that are systematically organized to complete an activity, ensuring that the process conforms to a predefined algorithm or sequence of steps. Basically, the process algorithm consists of three stages. This process receives input, and further produces an output or output. Some of the criteria or characteristics that must be met by an algorithm in order to be considered a good algorithm include:

1) Finiteness: The algorithm must be able to complete its task within a limited time, with a reasonable number of steps or within an acceptable time limit. Efficiency plays an important role in determining the quality of the algorithm.

2) Definiteness: Algorithms must be deterministic, producing consistent output when given the same input. This certainty is important for predicting the results of algorithms in programming and computing.

3) Input: The algorithm must clearly define the type and format of input it can accept. This involves determining the type of data that can be processed (such as integers, text, or images) as well as specifying specific limits or conditions for input.

4) Output: The algorithm should provide expected and well-defined results. It covers the type of output data, the output format, and the way the output is formed from a given input. The resulting result must be correct and relevant to the problem being solved.

5) Effectiveness: The algorithm should be designed to achieve the set goals and deliver the solution with efficiency or precision. In addition, it is necessary to consider the efficient use of resources such as memory and time (Saputro & Pradana, 2022).

Algorithms in general have 3 basic structures, namely Looping, Branching and ordered or sequential:

1) Sequence: The pattern applied to execute a program with sequential statements is referred to as a sequential pattern. This pattern involves the sequential execution of procedures or instructions, starting at the beginning and ending at the end. This sequential pattern contains no elements of repetition or spike. Some features of sequential patterns include each statement executed one at a time sequentially, the execution of statements is done in order, the algorithm ends with the final statement, and several different results can be generated by changing the order of statements.

Here is an example of a reference algorithm:

```
Input: year
If (year mode 4 = 0) Then
  If (year mode 100 = 0) Then
    If (year mode 400 = 0) Then
      Output: "Leap Year"
    Other
      Output: "Not a leap year"
  Other
    Output: "Leap Year"
  Other
    Output: "Not a leap year"
```

Figure 3. Example of Pseudocode
2) Sorting: In the context of algorithms, selection refers to new events that will occur if a certain condition is met. Conditions here refer to conditions that must be met in order for an instruction to be executed. In other words, the instruction will only be executed if a condition is true, and will not be executed if the condition is false. For example, we can consider conditions to determine whether a number is an odd or even number.

Here are the steps of the algorithm to check whether a number is even or odd enter a number, then enter the number (-1), divide the number by two, and the number is considered even if the result of division is zero.

Here is an example of a sorting algorithm:

```
Begin
i = 1
As long as i <= 10 do
  Show i
  i = i + 1
Finish
```

**Figure 4. Example of Runtutan Algorithm**

3) Loop: A recurrence pattern is a pattern that repeatedly repeats a single line or a block of program lines. This repetition process will continue to be carried out according to predetermined needs. Calculates the area of a rectangle, accepts length and width inputs, calculates area by multiplying length and width, and displays area results.

Here is an example of a recurrence algorithm:

```
Input: number

Begin
  If number mod 2 = 0 Then
    Show "Even number"
  Other
    Show "Odd numbers"
Finish
```

**Figure 4. Example of a sorting algorithm**

```
Input: length, width
Begin
  area = length * width
  Show area
Finish
```

**Figure 5. Example of Recurrence Algorithm**
CONCLUSION

Algorithms are step-by-step instructions that guide beginners in designing programs and solving problems. A solid understanding of basic algorithms can make it easier for one to learn programming. The algorithm itself is a series of steps organized in a structured and logical manner to achieve the solution of a problem. The five key aspects of algorithms involve finiteness, definiteness, input, output, and effectiveness. Programming languages are classified into three levels, namely low, intermediate, and high. Algorithm notation, which includes the basic framework for building programs, has become essential for program builders. There are three general methods for writing algorithms: descriptive, pseudocode, and flowcharts. This understanding has an important role in the development of more complex programming projects. The conclusion of this article is that learning basic programming algorithms is an essential necessity for anyone who wants to get started in the world of programming.

REFERENCES


