

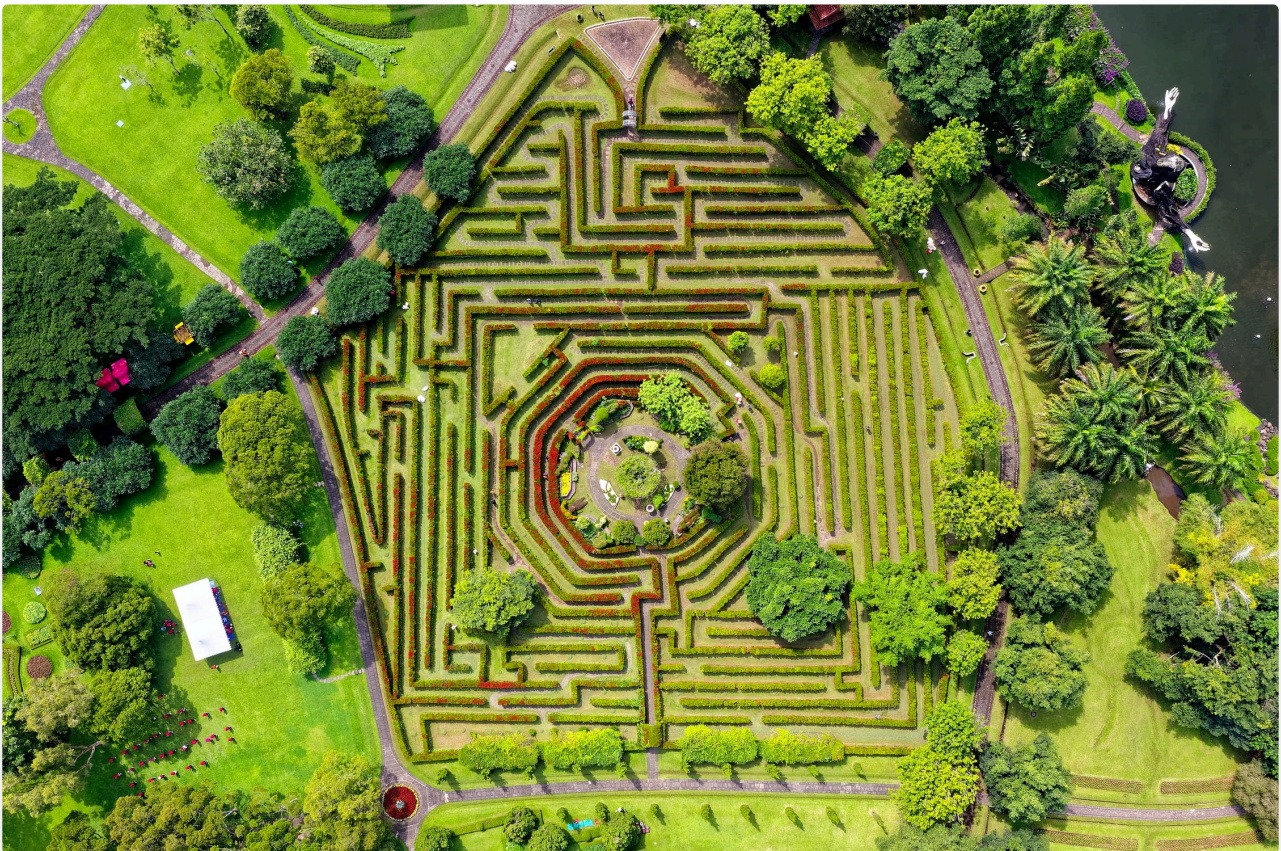
# Introduction to Algorithms

---

## Textbook

---

## Introduction to Algorithms



[Algorithms](#) are basically step-by-step instructions to accomplish a task. Algorithms are everywhere in our daily lives, and they help us accomplish what we want to do.

For example, this is an algorithm for making toast.

1. Open the bread bag.
2. Pull slices of bread out of the bag.
3. Place slices of bread into the toaster.
4. Turn the toaster on.
5. Pull slices out of the toaster.

Starting with step one, if you follow all the steps you will have made toast.

Question: How might you write down an algorithm for building a snowman?



Algorithms can be expressed in a variety of ways, such as natural language, diagrams, and pseudocode.

## Computer Algorithms

A [computer algorithm](#) is a set of instructions for a computer to follow to accomplish a certain task.

Just like most problems in the world, there are many ways to build an algorithm that would accomplish the same task. Sometimes, programs that look really similar are actually accomplishing different tasks. Usually, the algorithms that are considered the best are the answers that get the solution the fastest. Especially in computer science, we are constantly looking for ways to make programs run faster and smoother.

Algorithms can either be created from an idea, by combining existing algorithms, or modifying an existing algorithm. Sometimes it's easier to take an existing algorithm and tweak it to fit your specific situation. This can reduce development time, testing time, and cuts down on the number of bugs.

Every algorithm can be constructed using combinations of sequencing, selection, and iteration. Algorithms executed by programs are implemented using programming languages such as Python or JavaScript. Combinations of code statements carry out an algorithm.

## Sequencing



One important aspect of algorithms is [sequencing](#), or the order of the steps. Let's look at the toast algorithm again. What would happen if you did the steps in the wrong order? Could you pull slices of bread out of a bag you haven't opened yet? Or could you pull slices of bread out of a toaster if you haven't put bread in it in the first place?

The sequence of the steps in an algorithm matter.

Let's look at what a computer algorithm might look like to find the total cost of a grocery list.

1. Create a variable named total.
2. Add the price of each grocery item to the variable named total.
3. Create a variable named tax.

4. Multiply the variable named total by .03 and set it equal to the variable named tax.
5. Add the variable named tax to the variable named total.
6. Print the variable named total.

This is an example of a computer algorithm using pseudocode.

## Selection

[Selection](#) refers to the parts of an algorithm where choices are made. Often, in programming selection determines which parts of an algorithm are executed based on a condition being `true` or `false`. In the grocery example above, let's say they ask if you want to donate to a local charity. The algorithm needs to take that choice into consideration. The new algorithm might look like this.

1. Create a variable named total.
2. Add the price of each grocery item to the variable named total
3. Create a variable named tax.
4. Multiply the variable named total by .03 and set it equal to the variable named tax.
5. Add the variable named tax to the variable named total.
6. Create an input that asks if the user wants to donate to charity
7. If the answer is yes, add the donation to the total. If the answer is no, do not add a donation to the total.
8. Print the variable named total.

Step number 7 is an example of selection in the algorithm. [Selection](#) is important because it accounts for different events happening.

## Iteration

Let's explore how [iteration](#) would work in the example above. What would happen if you were buying 35 oranges? According to the algorithm above, you would need to add the price of an orange to your total 35 times. Wouldn't it be easier to multiply the price of oranges by 35 and add that to your total? Iteration refers to parts of your algorithm that repeat. Often there are ways to simplify your code when parts repeat. [Iteration](#) is an important part of algorithms that helps to produce the correct output in an efficient way.

## Same Result? Or Different Result?

[Sequencing](#), [selection](#), and [iteration](#) are important to make sure that your algorithms produce the correct output every time. Let's explore a few algorithms to see if they yield the same result or a different result.

### Toast Example 1:

1. Open the bread bag.
2. Pull slices of bread out of the bag.
3. Place slices of bread into the toaster
4. Turn toaster on
5. Pull slices out of the toaster.

**Toast Example 2:**

1. Turn toaster on
2. Open the bread bag.
3. Pull slices of bread out of the bag.
4. Place slices of bread into the toaster
5. Pull slices out of the toaster.

Does Toast Example 1 and 2 both produce the same result?

[Show answer/example](#)

Let's explore another example.

**Grocery List Example 1:**

1. Create a variable named total.
2. Add the price of each grocery item to the variable named total
3. Create a variable named tax.
4. Multiply the variable named total by .03 and set it equal to the variable named tax.
5. Add the variable named tax to the variable named total.
6. Print the variable named total.

**Grocery List Example 2:**

1. Create a variable named total.
2. Create a variable named tax.
3. Multiply the variable named total by .03 and set it equal to the variable named tax.
4. Add the price of each grocery item to the variable named total
5. Add the variable named tax to the variable named total.
6. Print the variable named total.

Do Grocery List Examples 1 and 2 produce the same result?

If you look carefully through the algorithms, these do not yield the same result. The second example calculates the tax on a total that does not yet have the price of groceries added to it, so the value for tax would be incorrect. In this example, the sequence matters.

An important point to note is that many different algorithms can be used to accomplish the same task. Often programmers need to choose between different options to find the best possible algorithm.

# Recursion

Sometimes, an algorithm can solve a problem by breaking it down into smaller, similar versions of the same problem. This technique is called [recursion](#).

Instead of repeating steps using a loop (iteration), a recursive algorithm solves a problem by calling itself with simpler inputs until it reaches a very basic case that it can solve directly.

Think of it like a set of Russian nesting dolls, where each doll contains a smaller version of itself. Once the smallest doll is found, you can then build your way back up.

A common example is calculating a factorial (like  $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$ ); a recursive algorithm would calculate  $5!$  by first asking for  $4!$ , and so on, until it reaches  $1!$ , which it knows is just 1. Recursion is a powerful and elegant way to solve certain types of problems, particularly those that have a naturally self-similar structure.

## Summary

[Algorithms](#) are step by step instructions to accomplish a task. These algorithms exist in our everyday lives as well as in computer programming. Algorithms depend on [iteration](#), [sequencing](#), and [selection](#) to work correctly. Often, many different algorithms can accomplish the same task. Also, sometimes algorithms that appear similar can yield different side effects or results.

## Questions (8)

1. Which of the following refers to how an algorithm repeats?

MULTIPLE CHOICE

Choose the correct answer:

- A. sequencing
- B. selection
- C. iteration
- D. variety

2. Which of the following refers to the order of the steps in an algorithm?

MULTIPLE CHOICE

Choose the correct answer:

- A. selection
- B. sequence
- C. iteration
- D. variety

**3. Which of the following refers to the parts of an algorithm where choices affect the output?**

MULTIPLE CHOICE

**Choose the correct answer:**

- A. sequencing
- B. selection
- C. iteration
- D. variety

**4. True or False: Algorithms are only found in computer programs.**

MULTIPLE CHOICE

**Choose the correct answer:**

- A. True
- B. False

**5. True or False: If you want to get the correct result every time, you must use the exact same algorithm every time.**

MULTIPLE CHOICE

**Choose the correct answer:**

- A. True
- B. False

**6. What are some ways that algorithms can be expressed? (Select all that apply)**

SELECT MULTIPLE

**Select all that apply:**

- A. natural language
- B. diagrams
- C. pseudocode
- D. algorithms can only be expressed with programming languages.

**7. True or False: Often programmers need to choose between different options to find the best possible algorithm.**

MULTIPLE CHOICE

**Choose the correct answer:**

- A. True
- B. False

**8. Often, in programming selection determines which parts of an algorithm are executed based on a condition being what?**

MULTIPLE CHOICE

**Choose the correct answer:**

- A. true or false
- B. up or down
- C. left or right
- D. in or out

## Answer Keys & Solutions

### Questions

1. Which of the following refers to how an algorithm repeats?

MULTIPLE CHOICE

Correct Answer:

- |               |             |
|---------------|-------------|
| A. sequencing | ✗ Incorrect |
| B. selection  | ✗ Incorrect |
| C. iteration  | ✓ Correct   |
| D. variety    | ✗ Incorrect |

**Explanation:**

Algorithms that repeat iterate multiple times.

2. Which of the following refers to the order of the steps in an algorithm?

MULTIPLE CHOICE

Correct Answer:

- |              |             |
|--------------|-------------|
| A. selection | ✗ Incorrect |
| B. sequence  | ✓ Correct   |
| C. iteration | ✗ Incorrect |
| D. variety   | ✗ Incorrect |

**Explanation:**

The order of steps in an algorithm is essential to effective algorithms.

3. Which of the following refers to the parts of an algorithm where choices affect the output?

MULTIPLE CHOICE

Correct Answer:



A. sequencing

✗ Incorrect

B. selection

✓ Correct

C. iteration

✗ Incorrect

D. variety

✗ Incorrect

**Explanation:**

The algorithm selects a choice that affects the outcome.

**4. True or False: Algorithms are only found in computer programs.**

MULTIPLE CHOICE

**Correct Answer:**

A. True

✗ Incorrect

B. False

✓ Correct

**Explanation:**

Algorithms appear all over in our daily lives.

**5. True or False: If you want to get the correct result every time, you must use the exact same algorithm every time.**

MULTIPLE CHOICE

**Correct Answer:**

A. True

✗ Incorrect

B. False

✓ Correct

**Explanation:**

There are often many algorithms to get the same result.

**6. What are some ways that algorithms can be expressed? (Select all that apply)**

SELECT MULTIPLE

**Correct Answers:**

A. natural language

✓ Correct

B. diagrams

✓ Correct

C. pseudocode

✓ Correct

D. algorithms can only be expressed with programming languages.

✗ Incorrect

#### Explanation:

There are 3 correct answers.

### 7. True or False: Often programmers need to choose between different options to find the best possible algorithm.

MULTIPLE CHOICE

#### Correct Answer:

A. True

✓ Correct

B. False

✗ Incorrect

#### Explanation:

There are often many ways to generate an algorithm that works.

### 8. Often, in programming selection determines which parts of an algorithm are executed based on a condition being what?

MULTIPLE CHOICE

#### Correct Answer:

A. true or false

✓ Correct

B. up or down

✗ Incorrect

C. left or right

✗ Incorrect

D. in or out

✗ Incorrect

#### Explanation:

True and false are used often in computer science.

