ABSTRACT DATA TYPE

DATA TYPE

- **A data type consists of:**
  - A collection of data elements (a type)
  - A set of operations on these data elements
- **Data types in languages:**
  - **predefined:**
    - any language defines a group of predefined data types
    - (In C) int, char, float, double, …
  - **user-defined:**
    - allow programmers to define their own (new) data types
    - (In C) struct, union, …

Data type elements

- **Predefined:**
  - type: int
  - elements: …, -2, -1, 0, 1, 2, …
  - operations: +, -, *, /, %, …
- **User-defined:**
  - type: complex
  - elements: 1+3i, -5+8i, …
  - operations: add, remove, distance, …

Abstraction

- concentrating on the essentials and ignoring the details.
    - Sometimes abstraction is described as "remembering the 'what' and ignoring the 'how'".

Two Types

.Procedural abstraction

- The separation of the *logical properties of an action* from the details of how the action is implemented.

Eg:

```
from math import log
print(log(10))
```

O/P 2.302585092994046

We do not mind how log() function is implemented in math library. ie., how log(2) actually gets computed. We just care about functioning of log here.

.Data abstraction

- The separation of the *logical properties of data* from the details of how the data are represented.
- A data abstraction is a mental model of what can be done to a collection of data. It deliberately excludes details of how to do it.

Example: elapsed time

- Elapsed time refers to a period or extent of time, as opposed to an instant in time that you might read in a single glance at a clock.
- Elapsed time is generally measured in a mixture of hours, minutes, and seconds.

Example: a book

- How to describe a book?
- If we are implementing a card catalog and library checkout, it is probably enough to list the metadata
- (e.g., title, authors, publisher, date).

ABSTRACT DATA TYPE

- An abstract data type is a mathematical model for data types.
- The functional definition of a data structure is known as ADT (Abstract Data Type) which is independent of implementation.
- In abstract data type:
  - separates data type definition from representation
  - separates function declaration (prototypes) from implementation
- The keyword "Abstract" is used as we can use these data types, we can perform diff e working that is totally hidd

Object
Level of abstraction increases
Becomes more specific

Let's understand the abstract data type with a real-world example.

If we consider the smartphone,

- 4 GB RAM
- Snapdragon 2.2ghz processor
- 5 inch LCD screen
- Dual camera
- Android 8.0

The above specifications of the smartphone are the data, and we can also perform the following operations on the smartphone:

- **call():** We can call through the smartphone.
- **text():** We can text a message.
- **photo():** We can click a photo.
- **video():** We can also make a video.

The smartphone is an entity whose data or specifications and operations are given above.

But these operations are hidden from the end user.

TYPES OF DATA STRUCTURES

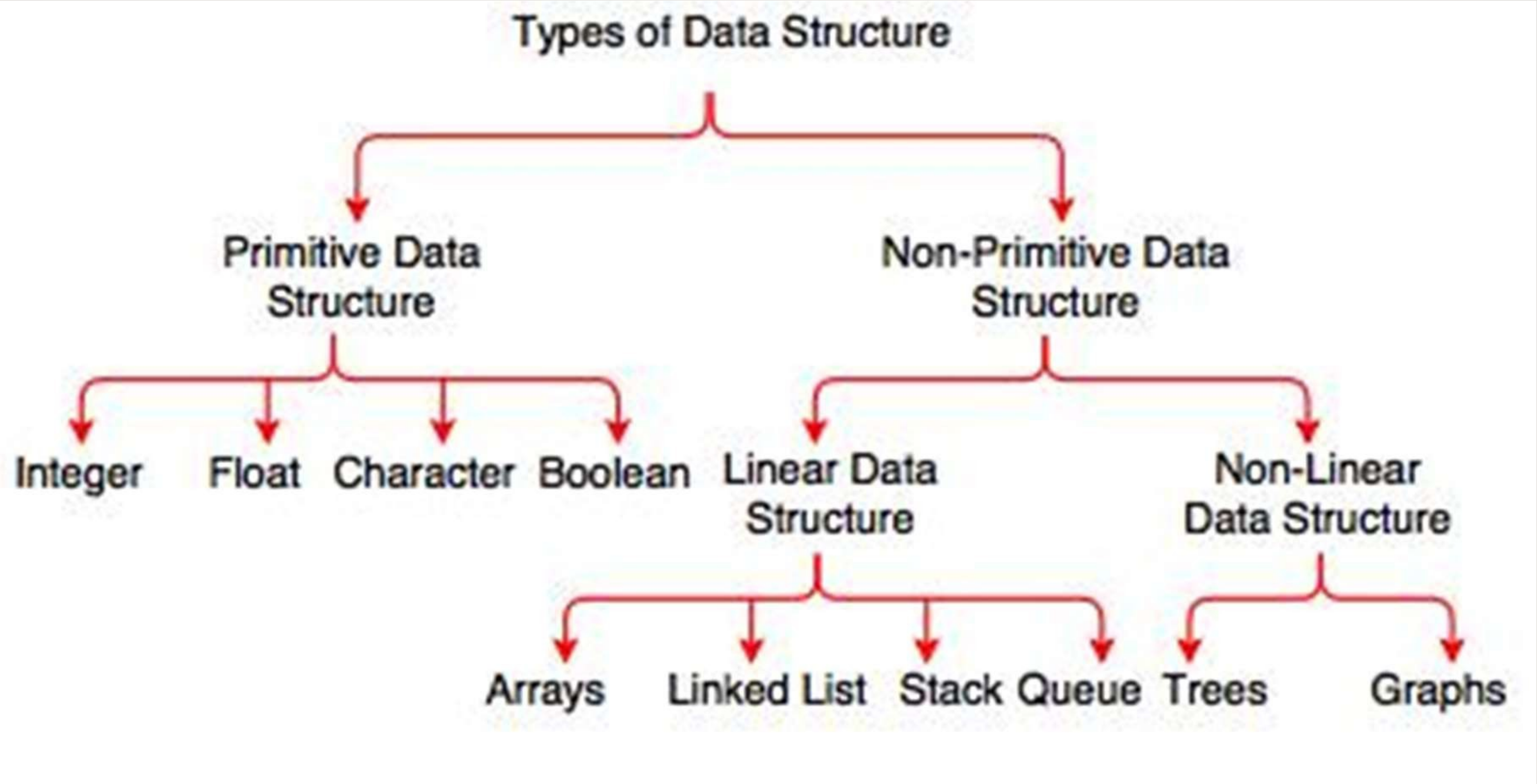


Fig. Types of Data Structure

TYPES OF DATA STRUCTURES

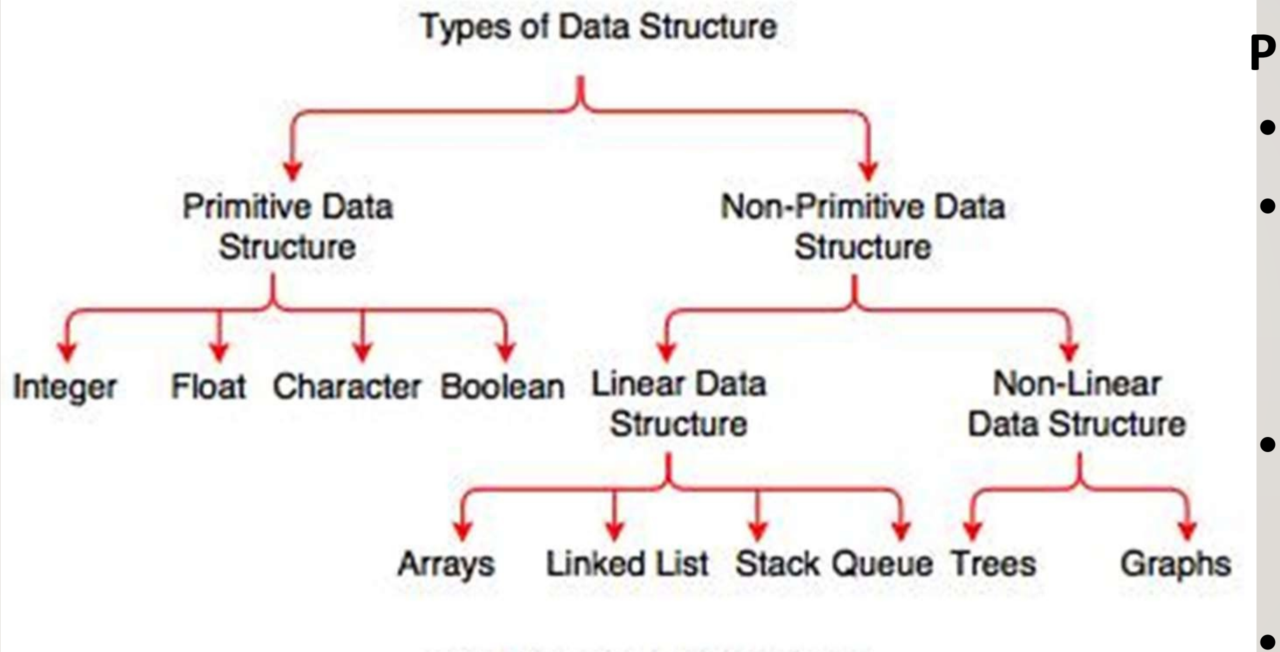


Fig. Types of Data Structure

Primitive Data Type

- pre-defined types
- used to represent **single value,** that cannot be broken down into a more simple data type.
- a basic data type available in most of the programming language.
- Ex. - Integer, Float, Boolean, Character

TYPES OF DATA STRUCTURES

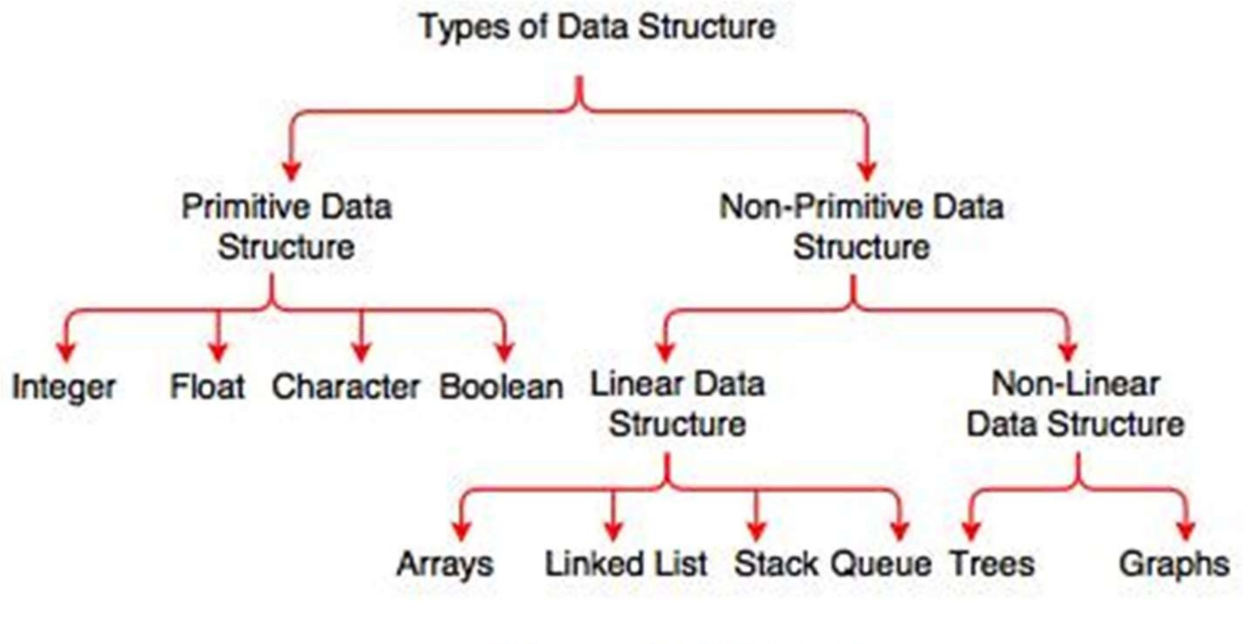


Fig. Types of Data Structure

Non-Primitive Data Type

- derived from primary data types are known as Non- Primitive data types.
- used to store group of values.

TYPES OF DATA STRUCTURES

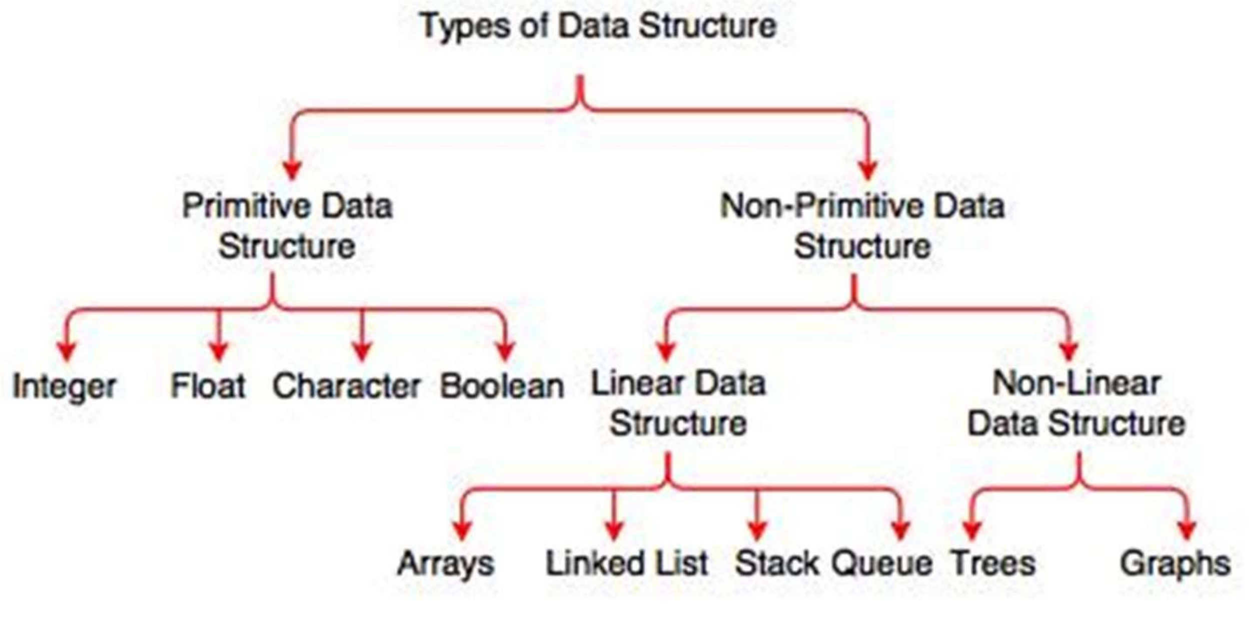


Fig. Types of Data Structure

Linear Data Structure

- data elements are arranged in a linear order where each and every elements are attached to its previous and next adjacent.
- Traverse the data elements sequentially.
- only one data element can directly be reached.
- memory is not utilized in an efficient way.
- includes array, linked list, stack and queues

TYPES OF DATA STRUCTURES

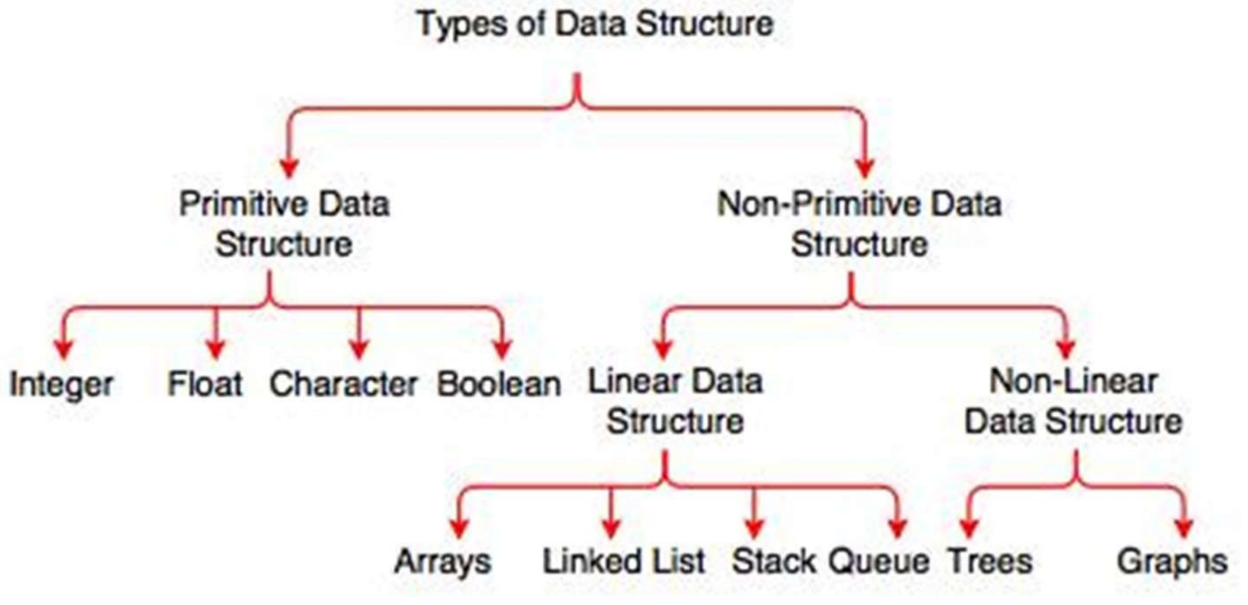


Fig. Types of Data Structure

Non-linear Data Structure

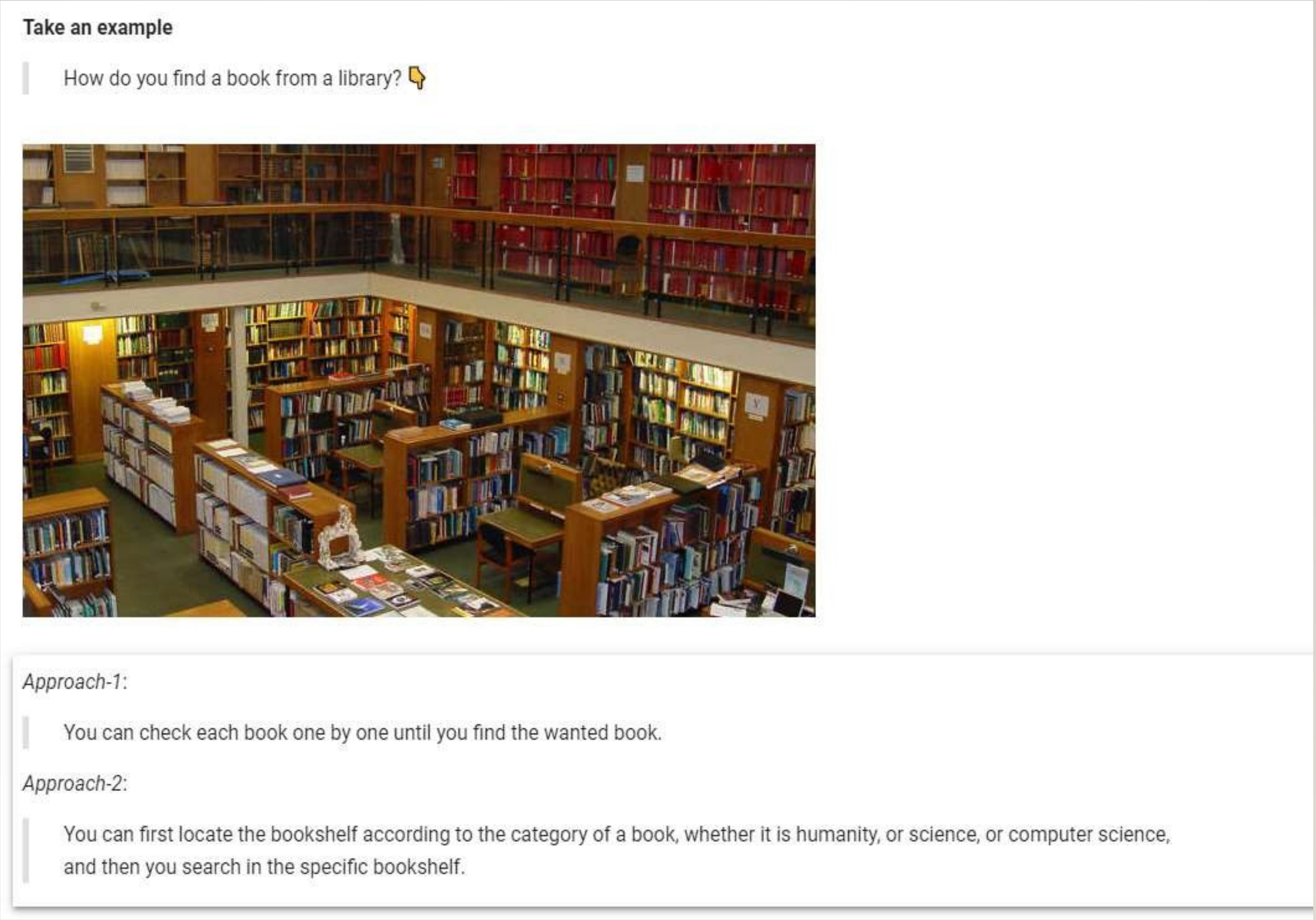
- data values are not arranged in order and a data item is connected to several other data items.
- It uses memory efficiently. Free contiguous memory is not required for allocating data items.
- It includes trees and graphs.

WHAT IS AN ALGORITHM?

- **An algorithm is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.**
- Algorithms can be specified in a natural language or **pseudo code**; In the earlier days of computing, **flowchart** method is used to express the algorithm's steps by a collection of connected geometric shapes .
- Algorithms are written at design time. Program is written at implementation time.

EVERY ALGORITHM MUST SATISFY THE FOLLOWING CRITERIA:

- **Unambiguous** − Algorithm should be clear and unambiguous. Each of its steps (or phases), and their inputs/outputs should be clear and must lead to only one meaning.
- **Input** − An algorithm should have 0 or more well-defined inputs.
- **Output** − An algorithm should have 1 or more well-defined outputs, and should match the desired output.
- **Finiteness** − Algorithms must terminate after a finite number of steps.
- **Feasibility** − Should be feasible with the available resources.
- **Independent** − An algorithm should have step-by-step directions, which should be independent of any programming code.



Take an example

> How do you find a book from a library? 👇

Approach-1:

> You can check each book one by one until you find the wanted book.

Approach-2:

> You can first locate the bookshelf according to the category of a book, whether it is humanity, or science, or computer science, and then you search in the specific bookshelf.