



Victoria University
of Bangladesh

Assessment Topic:

Final Assessment

Course Title: Algorithm

Course Code: CSI-227

Submitted To:

Umme Khadiza Tithi

Lecturer, Department of Computer Science & Engineering

Victoria University of Bangladesh

Submitted By:

Ruhul Amin

ID: 2120180051

Department: CSE

Semester: Spring-2023

Batch: 18th

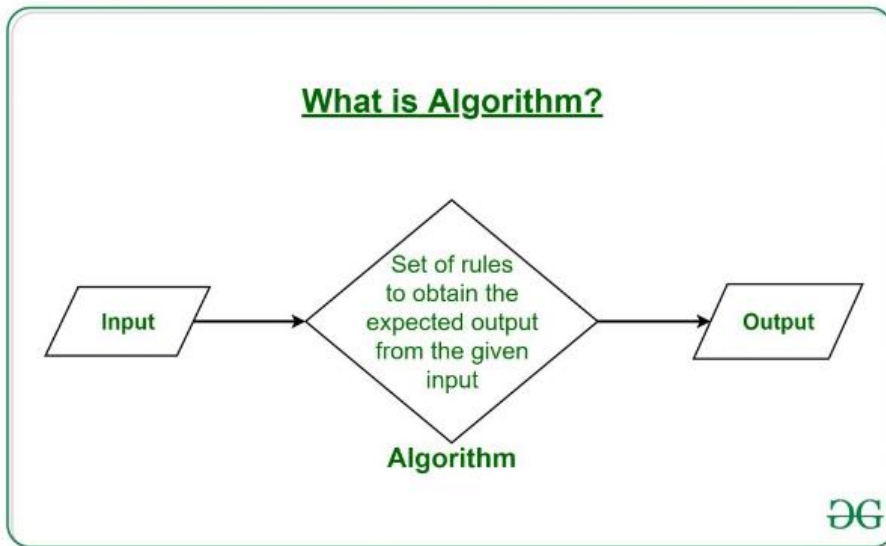
Submission Date: 05th June 2023

Answer to question no 1(a)

Why Learn Algorithms? Applications of Algorithm?

Answer: As applications are getting complex and data-rich, there are three common problems that applications face nowadays.

Algorithms can be simple and complex depending on what you want to achieve.



Data Search – Consider an inventory of 1 million (10⁶) items of a store. If the application is to search for an item, it has to search an item in 1 million (10⁶) items every time slowing down the search. As data grows, the search will become slower.

Processor speed – Processor speed although very high, falls limited if the data grows to billion records.

Multiple requests – As thousands of users can search data simultaneously on a web server, even the fast server fails while searching the data.

To solve the above-mentioned problems, data structures come to the rescue. Data can be organized in a data structure in such a way that all items may not be required to be searched, and the required data can be searched almost instantly.

Application of Algorithm: An algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output. Algorithms are generally created independent of underlying languages, i.e., an algorithm can be implemented in more than one programming language.

From the data structure point of view, the following are some important categories of algorithms –

- **Search** – Algorithm to search an item in a data structure.
- **Sort** – Algorithm to sort items in a certain order.
- **Insert** – Algorithm to insert an item in a data structure.
- **Update** – Algorithm to update an existing item in a data structure.
- **Delete** – Algorithm to delete an existing item from a data structure.

The following computer problems can be solved using Algorithm –

- Fibonacci number series
- Knapsack problem
- Tower of Hanoi
- All pair shortest path by Floyd-Warshall
- Shortest path by Dijkstra
- Project scheduling

Answer to question no 1(b)

Describe Dataflow of the Algorithm? Difference between an Algorithm and Pseudocode?

Answer: Dataflow of the Algorithm:

A flowchart is a blueprint that pictorially represents the algorithm and its steps. The steps of a flowchart do not have a specific size and shape rather it is designed in different shapes and sizes (see the image given below).



As shown in the above image, the boxes in different shapes and interconnected with arrows, are logically making a flow chart. A flowchart represents the general steps in a process.

Benefits of Flowchart

Let us now discuss the benefits of a flowchart.

Simplify the Logic

As it provides the pictorial representation of the steps; therefore, it simplifies the logic and subsequent steps.

Makes Communication Better

Because of having easily understandable pictorial logic and steps, it is a better and simple way of representation.

Effective Analysis

Once the flowchart is prepared, it becomes very simple to analyze the problem in an effective way.

Useful in Coding

The flowchart also helps in the coding process efficiently, as it gives directions on what to do, when to do it, and where to do it. It makes the work easier.

Proper Testing













Further, a flowchart also helps in finding the error (if any) in the program

Applicable Documentation

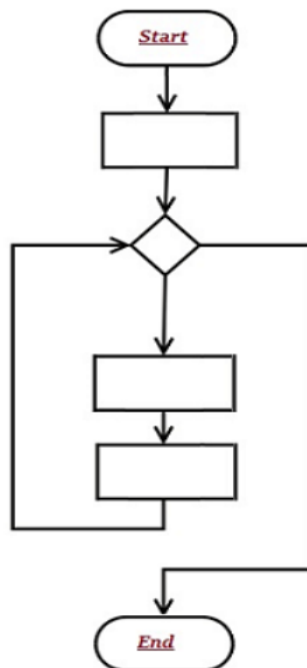
Last, but not the least, a flowchart also helps in preparing the proper document (once the codes are written).

Flow-Chart Symbols

The following table illustrates the symbols along with their names (used in a flow-chart)

Name	Symbol	Name	Symbol
	Flow Line		Magnetic Disk
	Terminal		Communication Link
	Processing		Offline Storage
	Decision		Annotation
	Connector		Flow line
	Document		Off-Page Connector

Sample of Flow Chart



Difference between Algorithm and Pseudocode

The following table highlights the key differences between algorithm and pseudocode –

Algorithm	Pseudocode
It is defined as a sequence of well-defined steps. These steps provide a solution/ a way to solve a problem at hand.	It can be understood as one of the methods that help in the representation of an algorithm.
It is a systematic, and logical approach, where the procedure is defined step-wise	It is a simpler version of coding in a programming language.
Algorithms can be represented using natural language, flowcharts, and so on.	It is written in plain English, and uses short phrases to write the functionalities that a specific line of code would do.
This solution would be translated to machine code, which is then executed by the system to give the relevant output.	There is no specific syntax that is actually present in other programming languages. This means it can't be executed on a computer.
Many simple operations are combined to help form a more complicated operation, which is performed with ease by the computer	There are many formats that could be used to write pseudo-codes.
It gives the solution to a specific problem.	Most of these formats take the structure from languages such as C, LIST, FORTRAN, and so on.
It can be understood as the pseudocode for a program.	Pseudocode is not actually a programming language.
Plain text is used.	Control structures such as 'while', 'if-then else', 'repeat-until', and so on can be used.
It is easy to debug.	It is relatively difficult to debug.
Its construction is tough.	Its construction is easy.
There are no rules to follow while constructing it.	It has certain rules to follow while constructing it.

Answer to question no 1(c)

Types of Algorithms? Define Greedy Algorithm.

Answer: Types of Algorithms:

There are several types of algorithms available. Some important algorithms are:

1. **Brute Force Algorithm:** It is the simplest approach for a problem. A brute force algorithm is the first approach that comes to finding when we see a problem.
2. **Recursive Algorithm:** A recursive algorithm is based on recursion. In this case, a problem is broken into several sub-parts and called the same function again and again.
3. **Backtracking Algorithm:** The backtracking algorithm basically builds the solution by searching among all possible solutions. Using this algorithm, we keep on building the solution following criteria. Whenever a solution fails, we trace back to the failure point and build on the next solution and continue this process till we find the solution or all possible solutions are looked after.
4. **Searching Algorithm:** Searching algorithms are the ones that are used for searching elements or groups of elements from a particular data structure. They can be of different types based on their approach or the data structure in which the element should be found.
5. **Sorting Algorithm:** Sorting is arranging a group of data in a particular manner according to the requirement. The algorithms which help in performing this function are called sorting algorithms. Generally sorting algorithms are used to sort groups of data in an increasing or decreasing manner.
6. **Hashing Algorithm:** Hashing algorithms work similarly to the searching algorithm. But they contain an index with a key ID. In hashing, a key is assigned to specific data.
7. **Divide and Conquer Algorithm:** This algorithm breaks a problem into sub-problems, solves a single sub-problem, and merges the solutions together to get the final solution. It consists of the following three steps:
 - Divide
 - Solve
 - Combine

8. **Greedy Algorithm:** In this type of algorithm the solution is built part by part. The solution for the next part is built based on the immediate benefit of the next part. The one solution giving the most benefit will be chosen as the solution for the next part.

9. **Dynamic Programming Algorithm:** This algorithm uses the concept of using the already found solution to avoid repetitive calculation of the same part of the problem. It divides the problem into smaller overlapping subproblems and solves them.

10. **Randomized Algorithm:** In the randomized algorithm we use a random number so it gives immediate benefit. The random number helps in deciding the expected outcome.

Greedy Algorithm: Standard Greedy Algorithms:

- Activity Selection Problem
- Job Sequencing Problem
- Huffman Coding
- Huffman Decoding
- Water Connection Problem
- Minimum Swaps for Bracket Balancing
- Egyptian Fraction
- Policemen catch thieves
- Fitting Shelves Problem
- Assign Mice to Holes
- Greedy Problems on Array:

Greedy Problems with Operating System:

- First Fit Algorithm in Memory Management
- Best Fit Algorithm in Memory Management
- Worst Fit Algorithm in Memory Management
- Shortest Job First Scheduling
- Job Scheduling with two jobs allowed at a time
- Program for Optimal Page Replacement Algorithm

Greedy Problems on Graph:

- Kruskal's Minimum Spanning Tree

- Prim's Minimum Spanning Tree
- Boruvka's Minimum Spanning Tree
- Dijkstra's Shortest Path Algorithm
- Dial's Algorithm
- Minimum cost to connect all cities
- Max Flow Problem Introduction
- Number of single-cycle components in an undirected graph

Answer to question no 2(a)

What is Searching Algorithm and Sorting Algorithm?

Answer: Sorting Algorithm:

The sorting algorithm is used to sort data in maybe ascending or descending order. It's also used for arranging data in an efficient and useful manner.

Some common problems that can be solved through the sorting Algorithm are Bubble sort, insertion sort, merge sort, selection sort, and quick sort are examples of the Sorting algorithm.

For Example, the below list of characters is sorted in increasing order of their ASCII values. That is, the character with a lesser ASCII value will be placed first than the character with a higher ASCII value.

2	1	4	3
---	---	---	---

Unsorted Array

1	2	3	4
---	---	---	---

Array sorted in ascending order

4	3	2	1
---	---	---	---

Array sorted in descending order

Searching Algorithm:

The searching algorithm is the algorithm that is used for searching the specific key in particular sorted or unsorted data. Some common problems that can be solved through the Searching Algorithm are Binary search or linear search is one example of a Searching algorithm.

Based on the type of search operation, these algorithms are generally classified into two categories:

Sequential Search: In this, the list or array is traversed sequentially and every element is checked.

Interval Search: These algorithms are specifically designed for searching in sorted data structures. These types of searching algorithms are much more efficient than Linear Search as they repeatedly target the center of the search structure and divide the search space in half.

Answer to question no 2(b)**# What is the function of an Algorithm?**

Answer: Function of an Algorithm: The function of an algorithm is to provide a step-by-step procedure or a set of instructions for solving a problem or accomplishing a specific task. Algorithms are commonly used in computer science and programming to solve complex problems and automate processes.

The primary purpose of an algorithm is to outline a clear and unambiguous set of instructions that can be followed by a computer or a human to achieve a desired outcome. It defines the logic and sequence of operations required to solve a problem, process data, perform calculations, make decisions, or control the flow of information.

Algorithms can be designed for various purposes, such as sorting data, searching for information, encrypting or decrypting data, optimizing resource allocation, simulating real-world scenarios, or solving mathematical equations. They can be implemented in different programming languages and can range from simple and straightforward to highly complex and sophisticated.

In summary, the function of an algorithm is to provide a systematic and well-defined approach for solving problems and performing tasks, enabling efficient and accurate execution by computers or humans.

1. **Problem-Solving:** Algorithms are designed to solve specific problems or address certain tasks. They break down complex problems into smaller, manageable steps, making it easier to understand and solve the problem at hand. Algorithms provide a systematic approach to problem-solving by defining the necessary operations, data structures, and logical flow required to arrive at a solution.
2. **Repetitive Tasks:** Algorithms are used to automate repetitive tasks that would be time-consuming or error-prone if performed manually. By defining a sequence of steps, an algorithm can be used to perform calculations, manipulate data, or execute a series of actions repeatedly without human intervention.
3. **Efficiency and Optimization:** Algorithms play a crucial role in optimizing processes and improving efficiency. They help identify the most efficient way to perform a task, minimizing time, resources, or other constraints. Through careful algorithm design, it is possible to reduce computational complexity, improve performance, and achieve optimal results.
4. **Decision Making:** Algorithms can incorporate decision-making processes based on logical conditions. They can evaluate input data or conditions and make decisions accordingly, leading to different paths or outcomes within the algorithm's execution. Conditional statements, loops, and branching structures within an algorithm allow for decision-making and adaptive behavior.
5. **Information Retrieval and Search:** Algorithms are used to search for specific information or patterns within a dataset or collection. For example, search algorithms like binary search or hashing algorithms enable efficient retrieval of data by quickly narrowing down the search space and finding the desired information.
6. **Data Sorting and Organization:** Algorithms provide methods for sorting and organizing data. Sorting algorithms arrange data in a specific order, such as numerical or alphabetical, while algorithms for organizing data can structure it into appropriate data structures like lists, arrays, trees, or graphs.
7. **Resource Allocation and Scheduling:** Algorithms are used to allocate resources efficiently, such as assigning tasks to processors in a multi-core system or scheduling activities to optimize time usage. Resource allocation algorithms take into account various factors like availability, priority, constraints, and fairness to make optimal decisions.

8. **Mathematical Computations:** Algorithms are extensively used in mathematical computations, including arithmetic operations, statistical calculations, numerical analysis, solving equations, and modeling complex mathematical relationships. These algorithms provide precise instructions for performing mathematical operations and enable accurate results.

Overall, algorithms are fundamental tools in computer science and play a vital role in problem-solving, automation, optimization, decision-making, data processing, and various other areas where structured and logical instructions are required to achieve desired outcomes efficiently.

Answer to question no 2(c)

Define Mathematical Algorithm and Graph Algorithm. Define Divide and Conquer Algorithm.

Answer: Mathematical Algorithm:

An algorithm in math is a procedure, a description of a set of steps that can be used to solve a mathematical computation.

For example, a step-by-step procedure used in long divisions is a common example of a mathematical algorithm.

Example of Math Algorithm: The process of solving a mathematical problem such as, "What is 82 divided by 3?" could be achieved by doing the following algorithm:

How many times does 3 go into 8?

The answer is 2.

How many are left over now? 2

Put the 2 (tens) in front of the 3.

How many times does 3 go into 22?

The answer is 7, with a remainder of one.

And of course, the answer is 27 with a remainder of 1.

Graph Algorithm:

A graph is an abstract notation used to represent the connection between pairs of objects. A graph consists of –

Vertices – Interconnected objects in a graph are called vertices. Vertices are also known as nodes.

Edges – Edges are the links that connect the vertices.

There are two types of graphs –

Directed graph – In a directed graph, edges have direction, i.e., edges go from one vertex to another.

Undirected graph – In an undirected graph, edges have no direction.

Divide and Conquer Algorithm: To understand the divide and conquer design strategy of algorithms, let us use a simple real-world example. Consider an instance where we need to brush a type C curly hair and remove all the knots from it. To do that, the first step is to section the hair in smaller strands to make the combing easier than combing the hair altogether. The same technique is applied to algorithms.

The divide and conquer approach break down a problem into multiple sub-problems recursively until it cannot be divided further. These sub-problems are solved first and the solutions are merged together to form the final solution.

The common procedure for the divide and conquer design technique is as follows –

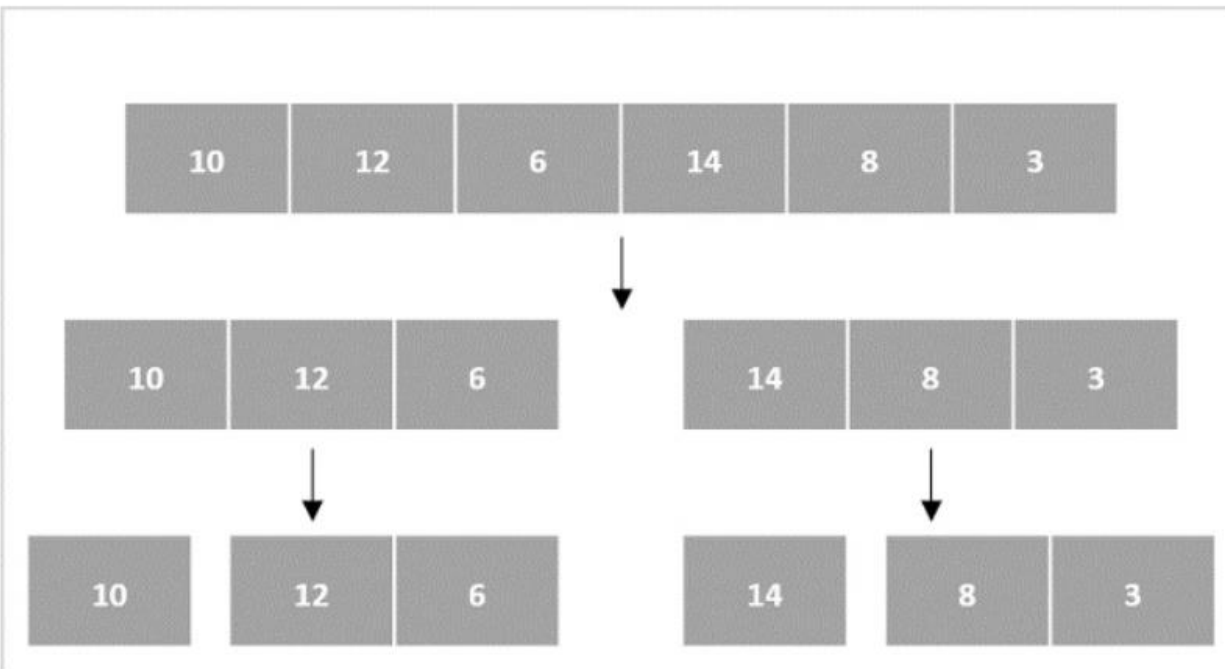
- **Divide** – We divide the original problem into multiple sub-problems until they cannot be divided further.
- **Conquer** – Then these subproblems are solved separately with the help of recursion
- **Combine** – Once solved, all the subproblems are merged/combined together to form the final solution of the original problem.

There are several ways to give input to the divide and conquer algorithm design pattern. Two major data structures used are – arrays and linked lists. Their usage is explained as

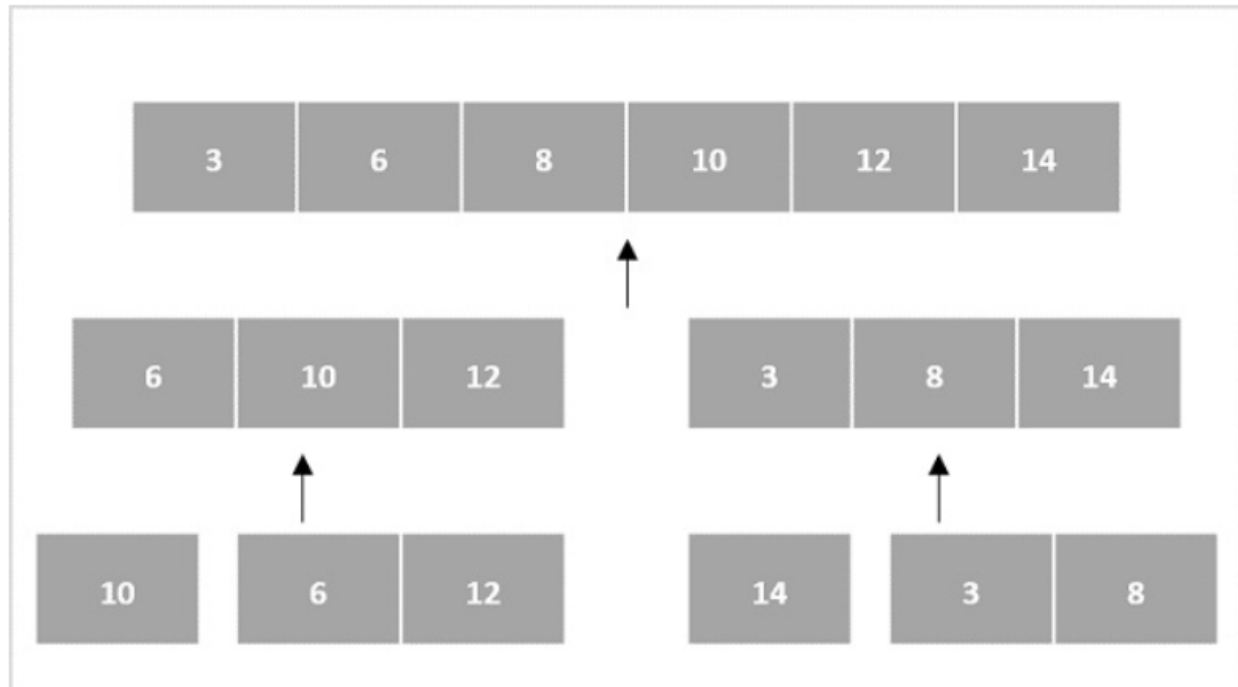
Arrays as Input

There are various ways in which various algorithms can take input such that they can be solved using the divide and conquer technique. Arrays are one of them. In algorithms that require input to be in the form of a list, like various sorting algorithms, array data structures are most commonly used.

In the input for a sorting algorithm below, the array input is divided into subproblems until they cannot be divided further.



Then, the subproblems are sorted (the conquer step) and are merged to form the solution of the original array back (the combine step).

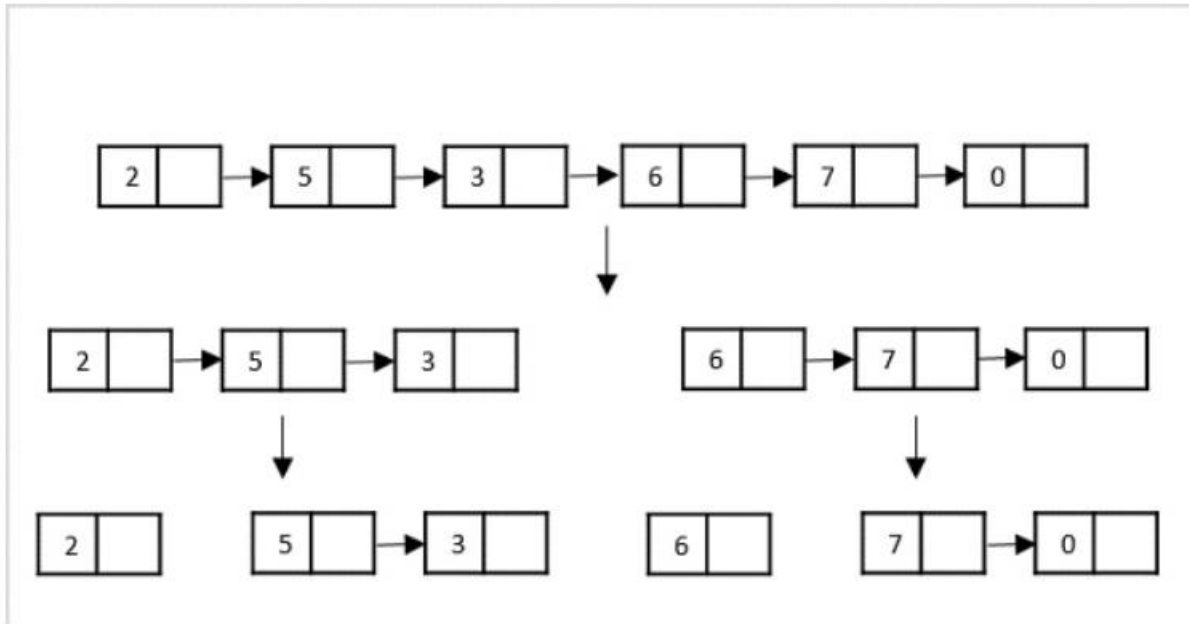


Since arrays are indexed and linear data structures, sorting algorithms most popularly use array data structures to receive input.

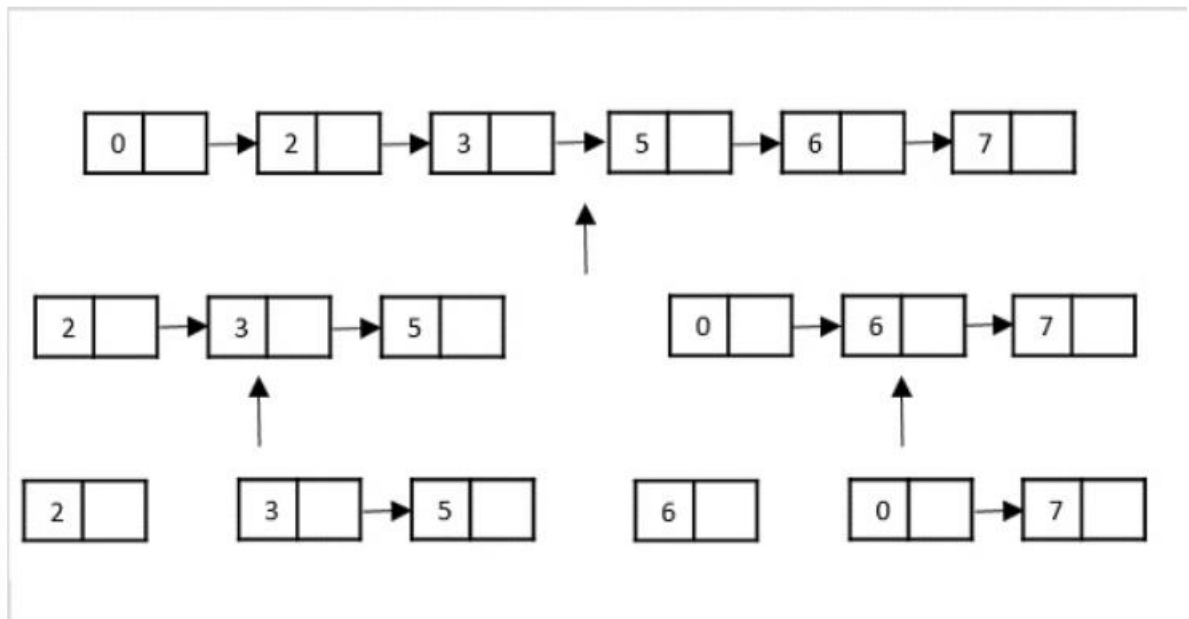
Linked Lists as Input

Another data structure that can be used to take input for divide and conquer algorithms is a linked list (for example, merge sort using linked lists). Like arrays, linked lists are also linear data structures that store data sequentially.

Consider the merge sort algorithm on a linked list; following the very popular tortoise and hare algorithm, the list is divided until it cannot be divided further.



Then, the nodes in the list are sorted (conquered). These nodes are then combined (or merged) recursively until the final solution is achieved.



Various searching algorithms can also be performed on the linked list data structures with a slightly different technique as linked lists are not indexed linear data structures. They must be handled using the pointers available in the nodes of the list.