Final Assessment | Spring 2023

Md. Shafayet Hossain

CSE - 21st Batch | **Course Title:** Algorithm

Course Code: CSI - 227 | **ID:** 2121210071

Answer to the Question no- 1

(a)

Why should we learn Algorithm?

Algorithms are a set of instructions or steps that are followed to accomplish a specific task or solve a specific problem. They are used in many fields such as computer science, mathematics, engineering and other disciplines. Learning algorithms can be beneficial for many reasons. For one, they are a fundamental part of computer science and are used to create computer programs and systems. They can also be used to analyze and interpret data, making them useful in fields like finance, healthcare, and business. Additionally, understanding algorithms can help you develop better problem-solving skills and think more logically.

Applications of Algorithm-

Algorithms are a fundamental part of programming and have many applications. Here are a few examples:

1. **Sorting and searching data:** Algorithms are used to sort and search through large amounts of data quickly and efficiently. This is useful in many applications such as databases, search engines, and e-commerce websites.
2. **Artificial intelligence and machine learning:** Algorithms are used to teach computers to recognize patterns, make predictions, and learn from data.
3. **Cryptography:** Algorithms are used to encrypt and decrypt data to ensure secure communication over the internet. This is important for online transactions, secure messaging, and other sensitive communications.
4. **Optimization:** Algorithms are used to optimize processes and systems, such as scheduling, logistics, and resource allocation. This is important in industries such as transportation, manufacturing, and healthcare.

These are just a few examples of the many applications of algorithms in programming.

(b)

Dataflow of Algorithm-

The dataflow of an algorithm refers to the flow of data through the different steps and operations of the algorithm. It describes how input data is transformed through a series of operations to produce output data.

The dataflow of an algorithm typically starts with the input data, which is fed into the algorithm. The input data may be in various forms such as text, images, audio, or numerical data. The algorithm then processes the input data through a series of steps or operations to transform it into the desired output.

During the processing, the algorithm may use various data structures such as arrays, linked lists, stacks, or queues to store and manipulate the data. The data may also be subjected to various operations such as sorting, filtering, or searching.

Once the algorithm has completed its processing, it produces the output data, which is typically in a different form than the input data. For example, an image processing algorithm may take an input image and produce an output image with various filters applied to it.

The dataflow of an algorithm can be visualized using flowcharts, data flow diagrams, or other similar diagrams that show the flow of data through the different steps and operations of the algorithm. By understanding the dataflow of an algorithm, programmers can better understand how it works and optimize its performance.

Differences between Algorithm and Pseudocode-

Algorithm and pseudocode are related concepts often used in programming. However, they are different in their nature and purpose. Here are the main differences between algorithm and pseudocode:

1. Nature: An **algorithm** is a step-by-step procedure for solving a problem or performing a task, expressed in a formal language. It is a precise description of how a program should behave. Pseudocode, on the other hand, is an informal high-level description of how a program works, expressed in a mix of natural language and programming language-like constructs. **Pseudocode** is used to sketch out the basic structure of an algorithm or program in a way that is easy to understand.

2. Formality: An **algorithm** is a formal description of a solution to a problem. It follows a specific syntax and structure, and is written to be executed by a computer. **Pseudocode**, on the other hand, has no formal syntax or structure. It is a rough sketch of an algorithm that is primarily meant to be read and understood by humans.

3. Readability: Pseudocode is generally more readable than algorithms because it uses plain language and high-level programming concepts. It is often used as a communication tool between programmers, as well as between programmers and non-technical stakeholders. **Algorithms**, on the other hand, can be more difficult to read and understand because they are written in a formal language and may use technical jargon.

4. Expressiveness: Algorithms are more expressive than pseudocode because they allow for precise and detailed descriptions of a program's behavior. Algorithms can be used as a blueprint for writing code and can be translated directly into a programming language. **Pseudocode**, on the other hand, is less precise and detailed than an algorithm and is used more for brainstorming and planning than for actual coding.

(c)

Categories of Algorithm-

Here are 7 types of algorithms:

- Brute Force Algorithm
- Recursive Algorithm
- Dynamic Programming Algorithm
- Divide and Conquer Algorithm
- Greedy Algorithm
- Backtracking Algorithm
- Randomized Algorithm

Greedy Algorithm:

A greedy algorithm is an algorithmic strategy that makes the locally optimal choice at each step in the hope of finding a global optimum. In other words, it chooses the best possible option at each step without considering the overall impact of that choice.

The basic idea behind a greedy algorithm is to make the best choice available at each step, based on the current information available, without worrying about the future consequences of that choice. The algorithm iteratively makes choices that seem best at the time, and it never revisits a decision once it has been made.

Greedy algorithms are often used for optimization problems where the goal is to find the best solution among a finite set of possibilities. They are simple to implement and can be very efficient for certain types of problems. However, they do not always guarantee the optimal solution and can sometimes lead to suboptimal results.

Answer to the Question no- 2

(a)

Searching Algorithm:

A search algorithm is the step-by-step procedure used to locate specific data among a collection of data. It is considered a fundamental procedure in computing. In computer science, when searching for data, the difference between a fast application and a slower one often lies in the use of the proper search algorithm.

The two most classical examples of that is the binary search and the merge sort algorithm.

Sorting Algorithm:

A sorting algorithm is an algorithmic procedure that arranges a list of items in a particular order, such as numerical or alphabetical order. The purpose of a sorting algorithm is to organize the items in a way that makes them easier to search, access, or analyze.

Sorting algorithms are commonly used in computer science, data analysis, and other fields that deal with large amounts of data. They are essential for efficient searching, filtering, and processing of data.

There are many different sorting algorithms, each with its own advantages and disadvantages depending on the specific application. Some of the most commonly used sorting algorithms are:

1. Bubble sort

2. Insertion sort

3. Selection sort

4. Quick sort

5. Merge sort

(b)

Function of an Algorithm-

The function of an algorithm is to provide a step-by-step solution to a problem or a set of instructions for performing a task. Algorithms are used in computer programming and other fields to automate processes, solve problems, and make decisions.

An algorithm takes input data and processes it through a set of instructions to produce an output. The instructions are typically written in a programming language or a formal language that can be executed by a computer. The output of an algorithm can be any type of data, such as a number, a string, or a complex data structure.

Algorithms are used in a wide range of applications, such as:

1. Sorting and searching data
2. Artificial intelligence and machine learning
3. Cryptography
4. Optimization

In summary, the function of an algorithm is to provide a systematic and efficient way of solving problems or performing tasks. By automating processes and making decisions based on data, algorithms can help improve efficiency, accuracy, and productivity in many fields.

(c)

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.

Graph Algorithm-

An algorithm is a mathematical process to solve a problem using a well-defined or optimal number of steps. It is simply the basic technique used to get a specific job done.

A graph is an abstract notation used to represent the connection between all pairs of objects. Graphs are widely-used mathematical structures visualized by two basic components: nodes and edges.

Graph algorithms are used to solve the problems of representing graphs as networks like airline flights, how the Internet is connected, or social network connectivity on Facebook. They are also popular in NLP and machine learning to form networks.

A divide algorithm is a type of algorithm used to solve problems by breaking them down into smaller, more manageable sub-problems. The basic idea behind a divide algorithm is to divide a problem into smaller sub-problems, solve each sub-problem independently, and then combine the solutions to obtain the final solution to the original problem.

Divide and Conquer Algorithm-

In divide-and-conquer algorithms, the "conquer" step refers to the process of combining the solutions of the sub-problems that were solved in the "divide" step. After solving the sub-problems, the conquer step combines the solutions of these sub-problems to arrive at the final solution for the entire problem.

For example, in the merge sort algorithm, the conquer step involves merging the two sorted sub-lists obtained from the previous recursive calls into a single sorted list. Similarly, in the quicksort algorithm, the conquer step involves combining the sorted sub-arrays from the partition step into a single sorted array.

The conquer step is an essential part of the divide-and-conquer paradigm and is what allows these algorithms to solve complex problems efficiently. By breaking down a large problem into smaller sub-problems and then combining the solutions of these sub-problems, divide-and-conquer algorithms can often solve problems much faster than brute-force methods.