2023

Victoria University of Bangladesh

58/11/A, Panthapath, Dhaka, Bangladesh

Algorithm

CSI-227

Final Assessment

Submitted By:

ARS Nuray Alam Parash

ID # 2120180041

18th Batch, BSc in CSE

E-mail: chatokparash@gmail.com

Mobile: 01922339393

Submission Date: 2 June 2023

Submitted To:

Umme Khadiza Tithi

Lecturer

Department of Computer Science & Engineering

Victoria University of Bangladesh (VUB)

Answer to the Question No- 1 (a)

Learning Algorithms is very important at this moment, it exists in every aspect of computer science. Algorithms are widely used in different outperforming fields such as social media, e-commerce, transportation, health care, education, etc. With the growing technology trends, algorithms these days are considered the critical building blocks for Machine Learning (ML) and Artificial Intelligence (AI).

An algorithm generally is a sequence of instructions, which eventually confirms the successful completion of a specific task. Being humans, we apply algorithms to perform some actions in every aspect of our daily lives.

Applications of the Algorithm:

- The internet is very much important for our daily life and we cannot even imagine our life without the internet, it is the outcome of clever and creative algorithms. Numerous sites on the internet can operate and falsify this huge number of data only with the help of these algorithms.
- Everyday electronic commerce activities are massively subject to our data, such as credit or debit card numbers, passwords, OTPs, etc. The center technologies used incorporate public-key cryptocurrency and digital signatures which depend on mathematical algorithms.
- An application that doesn't need algorithm content at the application level depends vigorously on the algorithm as the application relies upon hardware, GUI, networking, or object direction and all of these create a substantial use of algorithms.

Answer to the Question No- 1 (b)

The data flow in an algorithm refers to how data is input, processed, and output within the algorithm. It outlines the path that the data follows as it undergoes transformations and computations within the algorithm. Here's a general description of the data flow in an algorithm:

- Input: The algorithm begins with the input phase, where the necessary data is provided to the algorithm. This input can come from various sources, such as user input, files, databases, or other algorithms. The input data is typically in a specific format or structure required by the algorithm.

- Processing: Once the input data is available, the algorithm processes it according to the defined steps and logic. The processing phase involves applying various computations, operations, and transformations to the input data. This may include calculations, comparisons, sorting, filtering, or any other operations necessary to solve the problem at hand.

- Intermediate data: As the algorithm progresses through its steps, it may generate intermediate data as a result of the computations. Intermediate data represents the partially processed or transformed data at specific stages of the algorithm. This data is often used as input for subsequent steps or as a basis for decision-making within the algorithm.

- Control flow: The control flow of an algorithm determines the sequence in which different steps and operations are executed. It guides the order of processing and decision-making within the algorithm. Control flow structures, such as loops, conditionals, and branching statements, direct the flow of execution based on certain conditions or iterations.

- Output: Once the algorithm completes its processing, it produces an output. The output represents the final result or solution generated by the algorithm based on the input and computations. The output can take various forms, such as numerical values, data structures, visualizations, or reports, depending on the nature of the problem and the algorithm's purpose.

- Error handling: During the data flow, algorithms may encounter errors or exceptional conditions that need to be addressed. Error handling mechanisms are put in place to detect and handle errors appropriately. This may involve error messages, exception handling, or alternative paths in the algorithm to handle exceptional cases.

Algorithm and Pseudocode are two related terms in computer programming. The basic difference between an algorithm and a pseudocode is that an algorithm is a step-by-step procedure developed to solve a problem, while a pseudocode is a technique for developing an algorithm.

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 s 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-thenelse', '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 the Question No- 1 (c)

An algorithm consists of instructions or steps that are followed in a specific order to solve a problem. Algorithms are used in many fields, including computer science, mathematics, and logistics. An algorithm is made of well-defined steps, including mathematical operations, conditional statements, and loops executed in a specific order.

Here is a list of the most important type of algorithms to begin with:

1. Brute Force algorithm
2. Greedy algorithm
3. Recursive algorithm
4. Backtracking algorithm
5. Divide & Conquer algorithm
6. Dynamic programming algorithm
7. Randomized algorithm

Greedy Algorithm: In this algorithm, a decision is made that is good at that point without considering the future. This means that some local best is chosen and considered as the global optimal.

There are two properties in this algorithm:

- Greedily choosing the best option
- Optimal substructure property: If an optimal solution can be found by retrieving the optimal solution to its subproblems.

Greedy Algorithm does not always work but when it does, it works like a charm! This algorithm is easy to devise and most of the time the simplest one. But making locally best decisions does not always work as it sounds. So, it is replaced by a reliable solution called the Dynamic Programming approach.

Applications:

- Sorting: Selection Sort, Topological sort
- Prim's & Kruskal's algorithms
- Coin Change problem
- Fractional Knapsack Problem
- Job Scheduling algorithm

For better understanding let's go through the most common problem i.e. Job scheduling problem: Let us consider a situation where we are given the starting and end times of various events in an auditorium. Now your job is to maximize the number of events that can be organized in the auditorium where no two events overlap (starting time or ending time of one event does not fall in between the starting and endpoint of another event).

We consider six such events:

	A	B	C	D	E	F
Starting Time	1	2	5	4	3	6
Ending Time	6	3	9	6	5	10

Now a brute force solution would make us think that if we sort the events by their starting times & starting with the first event while excluding all events which overlap the previous will certainly give a solution but it won't maximize the number of events. Let us see, after sorting by starting time-

	A	B	E	D	C	F
Starting Time	1	2	3	4	5	6
Ending Time	6	3	5	6	9	10

So, the events that can be organized are- A, F. So, our brute force approach will have multiple such cases and fail if we don't select the optimal initial event. Now let's see what our greedy algorithm suggests. According to the greedy algorithm, we sort the events by their ending times, i.e. we select events that end first. Our new event table will become:

	B	E	A	D	C	F
Starting Time	2	3	1	4	5	6
Ending Time	3	5	6	6	9	10

So, we choose – B, E, C which is certainly a larger number of events than previous. Hence in such cases, the Greedy Algorithm gives the best solution to this type of problem.

Answer to the Question No- 2 (a)

Searching Algorithm: Any algorithm which solves the search problem, namely, to retrieve information stored within some data structure, or calculated in the search space of a problem domain, either with discrete or continuous values.

Searching algorithms are designed to check or retrieve an element from any data structure where that element is being stored. They search for a target (key) in the search space.

Types of Search Algorithms:

Linear Search: Linear Search is defined as a sequential search algorithm that starts at one end and goes through each element of a list until the desired element is found, otherwise, the search continues till the end of the data set.

Binary Search: Binary Search is a searching algorithm used in a sorted array by repeatedly dividing the search interval in half. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(Log n).

Sorting Algorithm: Sorting Algorithm is used to rearrange a given array or list of elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of elements in the respective data structure.

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.

Answer to the Question No- 2 (b)

The primary function of an algorithm is to solve a specific problem or perform a specific task in a systematic and efficient manner. Algorithms provide a step-by-step set of instructions or a procedure to achieve a desired outcome. Here are the key functions of an algorithm:

- **Problem-solving:** Algorithms are designed to solve problems by breaking them down into smaller, more manageable steps. They provide a structured approach to addressing complex problems, allowing programmers to devise solutions and achieve desired results.

- **Task automation:** Algorithms automate repetitive or tedious tasks by providing a predefined set of instructions. By following the algorithm, a computer or program can execute the necessary operations to perform the task accurately and efficiently, reducing the need for manual intervention.

- **Efficiency optimization:** Algorithms help optimize resource usage, such as time, memory, and processing power. By using efficient algorithms, developers can reduce execution time, minimize memory usage, and improve overall performance. Optimized algorithms can solve problems more quickly and handle larger datasets or complex operations more effectively.

- **Standardization:** Algorithms provide standardized procedures for solving common problems. They establish a common language and set of instructions that can be shared and understood by developers across different systems and programming languages. This standardization enables collaboration, code reuse, and the exchange of solutions in the programming community.

- **Decision-making:** Algorithms can incorporate decision-making processes based on specific conditions or criteria. By using logical constructs like conditionals (if-else statements), loops, and branching, algorithms can make choices and adapt their behavior based on input or intermediate results. This functionality allows algorithms to handle varying scenarios and handle different types of data.

- **Scalability:** Algorithms are designed to work efficiently with varying input sizes. They should be able to handle small datasets as well as large-scale problems without compromising performance. Scalable algorithms ensure that the solution remains efficient and effective even as the size or complexity of the problem increases.

- **Data manipulation and transformation:** Algorithms are used to process and manipulate data. They can transform data from one form to another, filter or sort data based on specific criteria,

perform calculations or aggregations, and extract meaningful insights from raw data. Algorithms provide the computational procedures to perform these data-related operations.

In summary, the function of an algorithm is to provide a systematic, efficient, and standardized approach to problem-solving and task execution. They automate processes, optimize efficiency, make decisions, handle data, and provide reliable and reusable solutions. Algorithms are a fundamental tool for programmers and computer scientists to solve a wide range of problems and achieve desired outcomes in various domains.

Answer to the Question No- 2 (c)

Mathematical Algorithms: An algorithm in math is a procedure describing 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.

Standard Algorithm for Addition- There are four simple steps for the standard algorithm for addition:

Step 1: Line up the numbers vertically by matching the place values.

Step 2: Add together the numbers that share the same place value, starting with the one's column.

Step 3: Write the sum below each column.

Step 4: If the sum of a column is greater than 9, carry over the tens digit to the next column.

Standard Algorithm for Subtraction- There are four simple steps for the standard algorithm for addition:

Step 1: Line up the numbers vertically by matching the place values.

Step 2: Subtract the numbers that share the same place value, starting with the one's column.

Step 3: Write the difference below each column.

Step 4: If the number on the top of a column is less than the number at the bottom, then regroup before subtracting.

Graph Algorithm: Graph algorithms are a set of instructions that traverse (visits nodes of a) graph. Some algorithms are used to find a specific node or the path between two given nodes.

Graphs are very useful data structures that can be to model various problems. These algorithms have direct applications on Social Networking sites, State Machine modeling, and many more.

Some of the most common graph algorithms are:

Breadth First Search (BFS)- Breadth First Search is one of the most simple graph algorithms. It traverses the graph by first checking the current node and then expanding it by adding its successors to the next level. The process is repeated for all nodes in the current level before moving to the next level. If the solution is found the search stops.

Depth First Search (DFS)- Depth First Search is one of the simplest graph algorithms. It traverses the graph by first checking the current node and then moving to one of its successors to repeat the process. If the current node has no successor to check, we move back to its predecessor and the process continues (by moving to another successor). If the solution is found the search stops.

Dijkstra Algorithm- Dijkstra's Algorithm is a graph algorithm presented by E.W. Dijkstra. It finds the single source shortest path in a graph with non-negative edges.

Floyd-Warshall Algorithm- Floyd-Warshall algorithm is great for finding the shortest distance between all vertices in a graph. It has a very concise algorithm and O(V^3) time complexity (where V is a number of vertices). It can be used with negative weights, although negative weight cycles must not be present in the graph.

Evaluation: Space Complexity: O(V^2), Worse Case Time Complexity: O(V^3).

Divide and Conquer: This technique can be divided into the following three parts:

1. **Divide:** This involves dividing the problem into smaller sub-problems.
2. **Conquer:** Solve sub-problems by calling recursively until solved.
3. **Combine:** Combine the sub-problems to get the final solution of the whole problem.

The following are some standard algorithms that follow Divide and Conquer algorithm.

- Quicksort is a sorting algorithm. The algorithm picks a pivot element and rearranges the array of elements so that all elements smaller than the picked pivot element move to the left side

of the pivot, and all greater elements move to the right side. Finally, the algorithm recursively sorts the subarrays on the left and right of the pivot element.

▪ Merge Sort is also a sorting algorithm. The algorithm divides the array into two halves, recursively sorts them, and finally merges the two sorted halves.

▪ Closest Pair of Points The problem is to find the closest pair of points in a set of points in the x-y plane. The problem can be solved in O(n^2) time by calculating the distances of every pair of points and comparing the distances to find the minimum. The Divide and Conquer algorithm solves the problem in O(N log N) time.

▪ Strassen's Algorithm is an efficient algorithm to multiply two matrices. A simple method to multiply two matrices needs 3 nested loops and is O(n^3). Strassen's algorithm multiplies two matrices in O(n^2.8974) time.

▪ Cooley–Tukey Fast Fourier Transform (FFT) algorithm is the most common algorithm for FFT. It is a divide and conquer algorithm which works in O(N log N) time.

>> END <<