Victoria University
of Bangladesh

Final Assessment

Md Bakhtiar Chowdhury

ID: 2121210061

Department: CSE

Semester: Spring -2023

Batch: 21th

Course Title: Algorithms

Course Code: CSI 227

Submitted To:

Umme Khadiza Tithi

Lecturer, Department of Computer Science & Engineering

Victoria University of Bangladesh

Submission Date: 07 June, 2023

Answer to the question no 1(a)

 # Why to learn Algorithm? Applications of Algorithm?

Answer:

Algorithm :An algorithm is a set of guidelines or rules used to solve a problem or carry out a task. It is a methodical approach that may be used to solve a mathematical riddle, run a computer program

it is no longer an unknown term for many of us, and today, algorithms exist 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 as the key 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.

The algorithm makes us to break anything into small steps, for easy and effective understanding of the complex things. For example, on a fine morning when you are ready to leave for the office, and you can't recollect where the car keys are. How would you find them? One smart approach might be the application of an algorithm, which is a sequential procedure to locate the answer (keys) quickly. At first, you will look at the places where you usually drop the keys. Next, you will recollect the last occasion when you have used them. Later, you will check the first room where you went, when you have entered the home. Eventually, by the flow of steps, you will find the car keys. Thus, the knowledge of algorithm stands to be a valued asset for enlightening our daily lives or even the lives of others. With algorithms, there is no limit for us to imagine, and making them real.

Learning algorithms is essential for several reasons:

- Problem-solving
- Efficiency
- Optimization
- Programming
- Data structures

Applications of Algorithm

- **Search Engines:** PageRank algorithm (used by Google) for ranking web pages based on relevance and authority.
- **Recommendation Systems:** Collaborative Filtering algorithm, Content-Based Filtering algorithm, and Hybrid Filtering algorithm for generating personalized recommendations.
- **Fraud Detection:** Machine learning algorithms like Decision Trees, Random Forests, and Neural Networks used for fraud detection and classification.
- **Natural Language Processing (NLP):** Naive Bayes algorithm for sentiment analysis, Hidden Markov Models for speech recognition, and Long Short-Term Memory (LSTM) for language generation.
- **Image and Video Processing:** Convolutional Neural Networks (CNN) for image recognition, object detection algorithms like Faster R-CNN and YOLO (You Only Look Once), and video analytics algorithms for tracking and analysis.
- **Logistics and Route Optimization:** Traveling Salesman Problem algorithms (e.g., genetic algorithms, ant colony optimization) for optimizing routes and vehicle allocation.

- **Financial Trading:** Moving Average Convergence Divergence (MACD), Bollinger Bands, and Arbitrage algorithms used for technical analysis and algorithmic trading.
- **Healthcare and Medical Diagnosis:** Support Vector Machines (SVM), Decision Trees, and Deep Learning algorithms used for medical diagnosis and disease prediction.
- **Social Media News Feed:** Facebook's EdgeRank algorithm for determining the relevance of content in users' news feeds.
- **DNA Sequencing:** Needleman-Wunsch algorithm for pairwise sequence alignment, and Smith-Waterman algorithm for local sequence alignment.

These algorithms are just a subset of the many algorithms used in these applications. Each application may involve a combination of different algorithms and techniques, depending on the specific requirements and goals of the system.

Answer to the question no 1(b)

 # Describe Dataflow of Algorithm? Difference between Algorithm and Psedocode?

Answer:

The data flow of an algorithm refers to the movement and transformation of data within the algorithm's computational steps. It illustrates how data is input, processed, and output throughout the algorithm's execution. Here's a general description of the data flow in an algorithm:

Input Data: The algorithm begins by receiving input data, which can be in various forms such as variables, arrays, lists, or data structures. This data serves as the initial input for the algorithm's computations.

Data Processing: The input data is processed through a series of computational steps or operations defined by the algorithm. These operations can include mathematical calculations, logical comparisons, data manipulations, or any other required tasks to achieve the algorithm's purpose.

Intermediate Data: As the algorithm progresses, intermediate data is generated. This data represents the partial results or transformations that occur during the execution of the algorithm. Intermediate data serves as inputs for subsequent computational steps.

Control Flow: The algorithm may incorporate control flow mechanisms such as conditionals (if statements), loops (for/while statements), or branching to determine the execution path based on certain conditions or iterations. The control flow influences the data flow, determining which computations are performed and how the data is processed.

Data Transformation: The algorithm may involve data transformation steps where the input data is modified or updated based on the computations performed. This can include updating variables, modifying data structures, or applying operations to transform the data into a desired format.

Output Data: Finally, the algorithm produces output data, which represents the result or solution generated by the algorithm. The output can be in the form of variables, arrays, lists, or any other data structure relevant to the problem being solved. The output data is typically used for further analysis, decision-making, or as input for subsequent algorithms or processes.

It's important to note that the data flow may vary depending on the specific algorithm and problem domain. Some algorithms may have more complex data flow patterns, involving multiple iterations, recursive calls, or parallel processing. The data flow within

an algorithm is designed to ensure that the input data is transformed and processed appropriately to produce the desired output and achieve the algorithm's objective.

Difference Between Algorithm and Pseudocode

Although there are various similarities between algorithm and pseudocode, there are a few differences between the two which are explained in the table provided below:

Algorithm	Pseudocode
It is a step-by-step description of the solution.	It is an easy way of writing algorithms for users to understand.
It is always a real algorithm and not fake codes.	These are fake codes.
They are a sequence of solutions to a problem.	They are representations of algorithms.
It is a systematically written code.	These are simpler ways of writing codes.
They are an unambiguous way of writing codes.	They are a method of describing codes written in an algorithm.
They can be considered pseudocode.	They can not be considered algorithms
There are no rules to writing algorithms.	Certain rules to writing pseudocode are there.

Answer to the question no 1(c)

 # Types of Algorithm? Define Greedy Algorithm ?

Answer:

There are many types of algorithms, and they can be categorized based on their purpose or the problem they solve. Some common types of algorithms include:

1. **Sorting Algorithms:** These algorithms are used to sort a list of items in a specific order, such as ascending or descending order. Examples include bubble sort, quicksort, and merge sort.

2. **Searching Algorithms:** These algorithms are used to find a specific item in a list of items. Examples include linear search, binary search, and interpolation search.

3. **Graph Algorithms:** These algorithms are used to solve problems related to graphs, such as finding the shortest path between two nodes. Examples include Dijkstra's algorithm, breadth-first search, and depth-first search.

4. **Computational Geometry Algorithms:** These algorithms are used to solve problems related to geometry, such as finding the area of a polygon. Examples include the convex hull algorithm and the Euclidean distance algorithm.

5. **Dynamic Programming Algorithms:** These algorithms are used to solve problems by breaking them down into smaller subproblems and solving them iteratively. Examples include the Fibonacci sequence algorithm and the Knapsack problem algorithm.

6. **Divide and Conquer Algorithms:** These algorithms solve problems by breaking them down into smaller subproblems, solving them independently, and then combining the solutions. Examples include the binary search algorithm and the merge sort algorithm.

7. **Greedy Algorithms:** These algorithms solve problems by making locally optimal choices at each step, with the hope of finding a global optimal solution. Examples include the Kruskal's algorithm and the Huffman coding algorithm.

8. **Backtracking Algorithms:** These algorithms solve problems by incrementally building a solution, and when it fails, backtracking to the previous step and trying a different approach. Examples include the N-Queens problem algorithm and the Sudoku solver algorithm.

9. **Randomized Algorithms:** These algorithms use randomness to solve problems, often by sampling a solution space to find a good approximation of the optimal solution. Examples include the Monte Carlo algorithm and the Quick select algorithm.

These are just a few examples of the many types of algorithms that exist.

Greedy Algorithms

A greedy algorithm is a type of algorithmic approach that follows the heuristic of making locally optimal choices at each step in the hope of finding a global optimum solution. It involves making the best possible decision based on the current information available without considering the future consequences or the overall effect on the final solution.

The term "greedy" refers to the algorithm's tendency to make choices that seem to be the most advantageous at the present moment. The algorithm selects the option that appears to be the best without considering the entire problem space. It makes decisions in a forward manner, typically without backtracking or reconsidering previous choices.

However, it's important to note that greedy algorithms do not guarantee the optimal solution for every problem. While they may provide efficient and satisfactory results in certain cases, they can also lead to suboptimal or incorrect solutions in other scenarios.

The design of a greedy algorithm involves identifying the problem's structure and properties to determine the locally optimal choices at each step. The algorithm

repeatedly makes these choices until a termination condition is met or a solution is found. Greedy algorithms are often employed in optimization problems, scheduling problems, and some graph-related problems.

Overall, the key idea behind greedy algorithms is to make the best choice available at each step, hoping that these locally optimal choices will lead to a reasonably good solution, even if it may not be the globally optimal solution.

Answer to the question no 2(a)

 # What is Searching Algorithm and Sorting Algorithm?

Answer:

Searching Algorithm: A searching algorithm is a method for finding a specific item or value in a collection of items, such as an array or a list. There are many types of searching algorithms, but some common ones include linear search, binary search, and interpolation search.

Linear search, also known as sequential search, involves scanning each item in the collection until the target value is found. This method is simple but can be time-consuming for large collections.

Binary search involves dividing the collection in half repeatedly until the target value is found. This method is more efficient than linear search, particularly for large collections, but requires that the collection is sorted beforehand.

Interpolation search is a variation of binary search that works better for collections with uniformly distributed values. It involves estimating the position of the target value based on its value relative to the minimum and maximum values in the collection.

Sorting Algorithm: A sorting algorithm is a method for arranging a collection of items in a specific order, such as ascending or descending order. There are many types of sorting algorithms, but some common ones include bubble sort, insertion sort, and quicksort.

Bubble sort involves repeatedly swapping adjacent elements in the collection until the entire collection is sorted. This method is simple but not very efficient, particularly for large collections.

Insertion sort involves building the sorted list one item at a time by inserting each new item into its correct position. This method is efficient for small collections but not very efficient for larger collections.

Quicksort involves partitioning the collection into smaller sub-collections and sorting each sub-collection separately. This method is efficient for large collections and is often used as a standard sorting algorithm in many programming languages.

Other common sorting algorithms include selection sort, merge sort, and heap sort. The choice of sorting algorithm depends on the size of the collection, the distribution of the values in the collection, and the desired time and space complexity.

Answer to the question no 2(b)

 # What is the function of an Algorithm?

Answer:

The function of an algorithm is to provide a step-by-step procedure or a set of instructions for solving a problem or performing a specific task. Algorithms serve as a blueprint or a guide that outlines the necessary operations and actions required to achieve a desired outcome. Here are some key functions of algorithms:

Problem Solving: Algorithms are primarily used to solve problems. They define a systematic approach to address a particular problem by breaking it down into smaller, manageable steps. Algorithms provide a clear and structured methodology to tackle complex problems efficiently.

Task Automation: Algorithms are employed to automate repetitive or computational tasks. By defining a sequence of instructions, algorithms enable the automation of various processes, reducing manual effort and increasing efficiency.

Decision Making: Algorithms can assist in making decisions based on predefined criteria or rules. They provide a logical framework for evaluating conditions and determining appropriate actions or choices.

Data Processing and Analysis: Algorithms play a crucial role in processing and analyzing data. They define the operations and transformations required to manipulate and extract meaningful insights from large datasets. Algorithms are used in various data-related tasks, such as sorting, searching, filtering, and statistical analysis.

Optimization: Algorithms are employed to optimize processes, systems, or solutions. They help identify the most efficient or effective course of action by considering different

factors and constraints. Optimization algorithms aim to find the best possible solution within a given set of parameters.

Pattern Recognition: Algorithms are utilized in pattern recognition tasks, such as image or speech recognition. They define the algorithms' logic and computations required to identify patterns or similarities within data and make accurate predictions or classifications.

Resource Management: Algorithms are used to manage resources effectively. In tasks like scheduling, allocation, or routing, algorithms help optimize the utilization of resources such as time, space, or bandwidth.

Overall, the function of an algorithm is to provide a structured and well-defined approach to problem-solving, automation, decision-making, data processing, optimization, pattern recognition, and resource management. Algorithms are fundamental tools in computer science and play a vital role in various applications across different domains.

Answer to the question no 2(c)

 # Define mathematical Algorithm and Graph Algorithm? Define divide and conquer Algorithm?

Answer:

Mathematical Algorithm:

A mathematical algorithm is a series of steps or instructions used to solve a mathematical problem. Mathematical algorithms can be used to solve a wide range of problems, such as finding the roots of a polynomial, solving a system of linear equations, or computing the value of a complex function.

Mathematical algorithms can be expressed in a variety of forms, including pseudocode, flowcharts, or programming languages such as MATLAB or Python. The efficiency and accuracy of a mathematical algorithm depend on its design, implementation, and input data.

Graph Algorithm:

A graph algorithm is a method for analyzing or manipulating a graph, which is a mathematical representation of a set of objects or nodes connected by edges or links. Graph algorithms can be used to solve a wide range of problems, such as finding the shortest path between two nodes, detecting cycles in a graph, or coloring the nodes of a graph.

Graph algorithms can be classified into two main categories: traversal algorithms and optimization algorithms. Traversal algorithms involve visiting all the nodes in a graph, while optimization algorithms involve finding the best or optimal solution to a specific problem.

Some common graph algorithms include depth-first search, breadth-first search, Dijkstra's algorithm, Bellman-Ford algorithm, and Kruskal's algorithm. The choice of graph algorithm depends on the specific problem being solved, the size and structure of the graph, and the desired time and space complexity.

A divide and conquer algorithm is a problem-solving technique that involves breaking down a complex problem into smaller, more manageable subproblems, solving them independently, and then combining their solutions to obtain the final result. The divide and conquer approach follows a recursive strategy and can be summarized in three steps:

Divide: The problem is divided into smaller, similar subproblems. This step typically involves breaking the problem into two or more subproblems of roughly equal size or dividing it into smaller instances of the same problem.

Conquer: Each subproblem is solved independently. This step involves applying the same algorithm recursively to solve the subproblems. If the subproblems are small enough, a base case is reached where the problem can be solved directly.

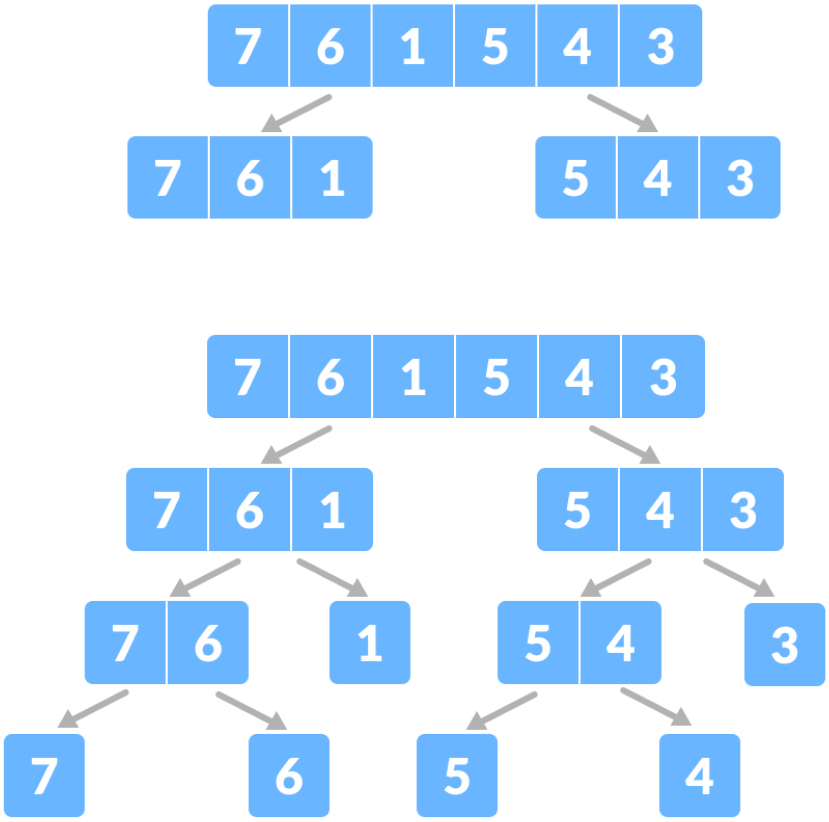
Combine: The solutions of the subproblems are combined to obtain the solution for the original problem. This step involves merging or aggregating the results of the subproblems into a single solution.

The divide and conquer strategy is based on the idea that if the subproblems can be solved independently and their solutions can be combined efficiently, then the solution to the original problem can also be obtained efficiently. This approach is often used to solve problems that can be broken down into smaller, similar subproblems, such as sorting, searching, and optimization problems.

Common examples of algorithms that employ the divide and conquer technique include:

Merge Sort: It recursively divides an array into smaller subarrays, sorts them, and then merges the sorted subarrays to obtain the final sorted array.

Quick Sort: It selects a pivot element, partitions the array around the pivot, and recursively sorts the subarrays on each side of the pivot.



Binary Search: It divides a sorted array into halves and eliminates one half based on the comparison of the target value with the middle element. This process is repeated until the target value is found or the search space is reduced to zero.

The divide and conquer approach offers an efficient way to solve many complex problems by breaking them down into smaller, more manageable parts and solving them recursively. It helps reduce the time and space complexity of the algorithm, making it a powerful tool in algorithm design and problem-solving.

>>>>>>END<<<<<<