Victoria University
of Bangladesh

Assessment Topic:

Mid 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

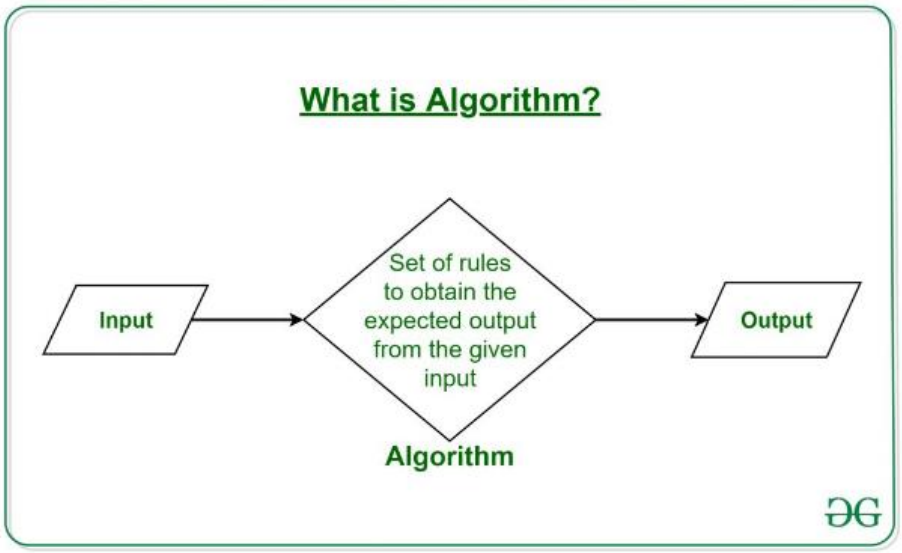
Submission Date: 20th April 2023

Answer to question no 1(a)

 # What is an Algorithm? Categories and Characteristics of Algorithm?

Answer: Algorithm: An algorithm is a process or a set of rules required to perform calculations or some other problem-solving operations, especially by a computer. The formal definition of an algorithm is that it contains a finite set of instructions that are being carried out in a specific order to perform a specific task. It is not the complete program or code; it is just a solution (logic) to a problem, which can be represented either as an informal description using a Flowchart or Pseudocode.

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



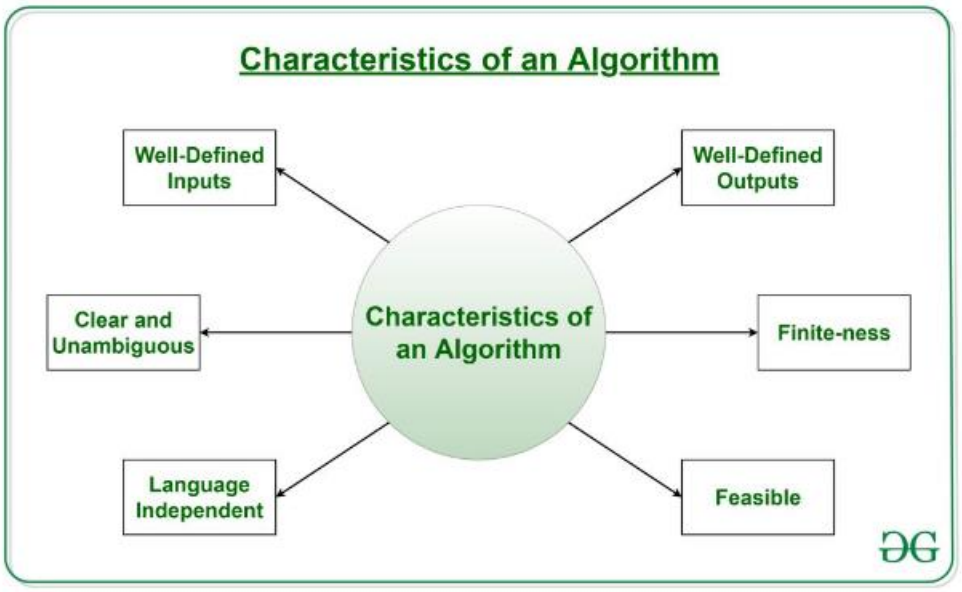
Categories of Algorithm:

i. Brute Force Algorithm
ii. Recursive Algorithm
iii. Randomized Algorithm
iv. Sorting Algorithm
v. Searching Algorithm
vi. Hashing Algorithm

Characteristics of Algorithm: The following are the characteristics of an algorithm:

Input: An algorithm has some input values. We can pass 0 or some input value to an algorithm.

Output: We will get 1 or more outputs at the end of an algorithm.
Unambiguity: An algorithm should be unambiguous which means that the instructions in an algorithm should be clear and simple.
Finiteness: An algorithm should have finiteness. Here, finiteness means that the algorithm should contain a limited number of instructions, i.e., the instructions should be countable.
Effectiveness: An algorithm should be effective as each instruction in an algorithm affects the overall process.
Language-independent: An algorithm must be language-independent so that the instructions in an algorithm can be implemented in any of the languages with the same output.



As one would not follow any written instructions to cook the recipe, but only the standard one. Similarly, not all written instructions for programming are an algorithm. In order for some instructions to be an algorithm, it must have the following characteristics:

Clear and Unambiguous: The algorithm should be clear and unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.
Well-Defined Inputs: If an algorithm says to take inputs, it should be well-defined inputs. It may or may not take input.
Well-Defined Outputs: The algorithm must clearly define what output will be yielded and it should be well-defined as well. It should produce at least 1 output.
Finite-ness: The algorithm must be finite, i.e., it should terminate after a finite time.

Feasible: The algorithm must be simple, generic, and practical, such that it can be executed with the available resources. It must not contain some future technology or anything.
Language Independent: The Algorithm designed must be language-independent, i.e., it must be just plain instructions that can be implemented in any language, and yet the output will be the same, as expected.
Input: An algorithm has zero or more inputs. Each that contains a fundamental operator must accept zero or more inputs.
Output: An algorithm produces at least one output. Every instruction that contains a fundamental operator must accept zero or more inputs.
Definiteness: All instructions in an algorithm must be unambiguous, precise, and easy to interpret. By referring to any of the instructions in an algorithm one can clearly understand what is to be done. Every fundamental operator in instruction must be defined without any ambiguity.
Finiteness: An algorithm must terminate after a finite number of steps in all test cases. Every instruction which contains a fundamental operator must be terminated within a finite amount of time. Infinite loops or recursive functions without base conditions do not possess finiteness.
Effectiveness: An algorithm must be developed by using very basic, simple, and feasible operations so that one can trace it out by using just paper and pencil.

Answer to question no 1(b)

 # Disadvantages of Algorithm and Advantages of Algorithm?

Answer: Disadvantages:

- **i.** Writing an algorithm takes a long time so it is time-consuming.
- **ii.** Understanding complex logic through algorithms can be very difficult.
- **iii.** Branching and Looping statements are difficult to show in Algorithms(imp).

Advantages:

- i. It is easy to understand.
- ii. An algorithm is a step-wise representation of a solution to a given problem.
- iii. In an Algorithm the problem is broken down into smaller pieces or steps hence, it is easier for the programmer to convert it into an actual program.

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
- Dijkastra'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:

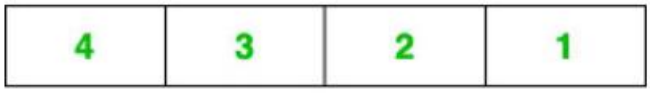
The sorting algorithm is used to sort data in maybe ascending or descending order. It's also used for arranging data in an efficient and useful manner.

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.

Unsorted Array

Array sorted in ascending order



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)

 # Define Huffman Coding.

Answer: Huffman Coding: Huffman coding is a lossless data compression algorithm. The idea is to assign variable-length codes to input characters, lengths of the assigned codes are based on the frequencies of corresponding characters.
The variable-length codes assigned to input characters are Prefix Codes, which means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bitstream.
Let us understand prefix codes with a counter-example. Let there be four characters a, b, c, and d, and their corresponding variable length codes be 00, 01, 0, and 1. This coding leads to ambiguity because the code assigned to c is the prefix of codes assigned to a and b. If the compressed bit stream is 0001, the de-compressed output may be “cccd” or “ccb” or “acd” or “ab”.

There are mainly two major parts in Huffman Coding

- Build a Huffman Tree from input characters.
- Traverse the Huffman Tree and assign codes to characters.

Algorithm:
The method which is used to construct optimal prefix code is called Huffman coding.

This algorithm builds a tree in a bottom-up manner. We can denote this tree by T

Let, |c| be number of leaves

|c| -1 are a number of operations required to merge the nodes. Q is the priority queue that can be used while constructing a binary heap.

```
Algorithm Huffman (c)
{
   n= |c|

   Q = c
   for i<-1 to n-1

   do
   {

       temp <- get node ()

      left (temp] Get_min (Q) right [temp] Get Min (Q)

      a = left [templ b = right [temp]

      F [temp]<- f[a] + [b]

      insert (Q, temp)

    }

return Get_min (0)
}
```

Answer to question no 2(c)

 # Define Mathematical Algorithm and Graph 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.