1. Explain the characteristics of algorithm.

Answer:

The word Algorithm means “a process or set of rules to be followed in calculations or other problem-solving operations”.

The characteristics of algorithm are as follows:

- Clear and Unambiguous: 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.
- Well-Defined Outputs: The algorithm must clearly define what output will be yielded and it should be well-defined as well.
- Finite-ness: The algorithm must be finite, i.e. it should not end up in an infinite loops or similar.
- Feasible: The algorithm must be simple, generic and practical, such that it can be executed upon will 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 same, as expected.

2. What are the implementation issues of algorithm? How it is different from implementation issues of data structure.

Answer:

Algorithms are commonly used to solve certain types of computational problems. We can often describe such a problem by specifying a relationship between input and output.

The sorting problem, for example, can be described like this:

- Input: a sequence $a_1, a_2, ..., a_n$ of n numbers.
- Output: a permutation (reordering) $a'_1, a'_2, ..., a'_n$ of the input, which satisfies the property $a'_1 \leq a'_2 \leq ... \leq a'_n$.

As an example: given the input sequence 12, 24, 13, 12, 8 a sorting algorithm must produce the output sequence 8, 12, 12, 13, 24. One such specific example of input for a problem is called an instance of the problem.

We say that an algorithm is correct if the algorithmic procedure stops for each problem instance and returns the required output. The algorithm solves the problem.

Data structure is a data organization, management, and storage format that enables efficient access and modification. More precisely, a data structure is a collection of data values, the relationships among them, and the functions or operations that can be applied to the data.

To develop a program of an algorithm , we should select an appropriate data structure for that algorithm . Therefore algorithm and its associated data structures form a program.

Algorithm +Data structure =Program

3. How do you analyze algorithm? Explain briefly.

Answer: In theoretical analysis of algorithms, it is common to estimate their complexity in the asymptotic sense, i.e., to estimate the complexity function for arbitrarily large input. The term "analysis of algorithms" was coined by Donald Knuth.

Algorithm analysis is an important part of computational complexity theory, which provides theoretical estimation for the required resources of an algorithm to solve a specific computational problem. Most algorithms are designed to work with inputs of

arbitrary length. Analysis of algorithms is the determination of the amount of time and space resources required to execute it.

Usually, the efficiency or running time of an algorithm is stated as a function relating the input length to the number of steps, known as time complexity, or volume of memory, known as space complexity.

Analysis of algorithm is the process of analyzing the problem-solving capability of the algorithm in terms of the time and size required (the size of memory for storage while implementation). However, the main concern of analysis of algorithms is the required time or performance. Generally, we perform the following types of analysis −

- Worst-case − The maximum number of steps taken on any instance of size a.
- Best-case − The minimum number of steps taken on any instance of size a.
- Average case − An average number of steps taken on any instance of size a.
- Amortized − A sequence of operations applied to the input of size a averaged over time.

To solve a problem, we need to consider time as well as space complexity as the program may run on a system where memory is limited but adequate space is available or may be vice-versa. In this context, if we compare bubble sort and merge sort. Bubble sort does not require additional memory, but merge sort requires additional space. Though time complexity of bubble sort is higher compared to merge sort, we may need to apply bubble sort if the program needs to run in an environment, where memory is very limited.

4. What is order of growth? Explain with suitable example.

Answer:

An order of growth is a set of functions whose asymptotic growth behavior is considered equivalent.

For example, 2n, 100n and n + 1 belong to the same order of growth, which is written O(n) in Big-Oh notation and often called linear because every function in the set grows linearly with n.