**Chapter -1 :** **The Role of Algorithms in Computing**

**Algorithm**:

An algorithm is a step by step method of solving a problem. It is commonly used for data processing, calculation and other related computer and mathematical operations.

An algorithm is also used to manipulate data in various ways, such as inserting a new data item, searching for a particular item or sorting an item.

**Applications of algorithm:**

* In different types of sorting process
* in genome sequencing for analyzing data
* manipulation of large volume of data
* public key cryptography and digital signatures
* finding the possible shortest rute
* in hash technique
* in searching technique

**Data Structure:**

Data structureis a way to store and organize data in order to facilitate access and modifications. No single data structure works well for all purposes, and so it is important to know the strengths and limitations of several of them.

**Hard Problems:**

Generally, we calculate the efficiency of an algorithm by checking how long time it will take to execute for each case. There are some problems, however, for which no efficient solution is known. This type of problem can be solved by NP complete problem.

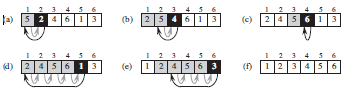
**Parallelism:**

Almost every machine or computer has more than one processing core now a days. So, for better efficiency of our algorithm, we should develop our algorithm keeping in mind that in a single second more than one tasks can be completed or executed. This type of algorithm is called multithreaded algorithm which takes the advantages of multiple cores.

**Chapter -2 : Getting Started**

**Insertion Sort:**

Insertion sort is a sorting algorithm which works fast and perfectly for a small numberof items. It's complexity is n^2



**Algorithm of insertion sort:**

1 for j = 2 to A.length

2 key = A[j]

3 // Insert A[j] into the sorted sequence A[1 .. j - 1]

4 i = j - 1

5 while i > 0 and A[i] > key

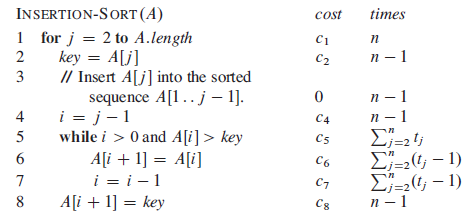
6 A[i + 1] = A[i]

7 i = i - 1

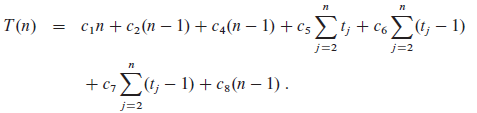
8 A[i + 1] = key

**Analysis of Insertion sort:**

Generally, the size of the input increases with the time taken by an algorithm which means it takes longer time to sort one thousand numbers than four numbers. Suppose, the cost of all the statement in the algorithm is c and the number of items are n. The cost of 1st statement will be c1 & the cost of 2nd statement will be c2 & so on. According to the Insertion sort algorithm, the 1st statement will the executed for n times, the 2nd statement will be executed for n -1 times and so on. the statement will be executed for the summation of of t j times from j=2 to n and the followings:



So the total running time will be :



**Chapter-3: Growth Of Functions**

The running time of an algorithm depends on how long it takes a computer to run the lines of code of the algorithm—and that depends on the speed of the computer, the programming language, and the compiler that translates the program from the programming language into code that runs directly on the computer, among other factors.

When it comes to analysing the complexity of any algorithm in terms of time and space, we can never provide an exact number to define the time required and the space required by the algorithm, instead we express it using some standard notations, also known as **Asymptotic Notations**.

**Θ-notation:**

The Θ-notation asymptotically bounds a function from above and below. When we have

only an asymptotic tight bound, we use Θ-notation.

**O-notation:**

The notation Ο(n) is the formal way to express the upper bound of an algorithm's running time. It measures the worst case time complexity or the longest amount of time an algorithm can possibly take to complete.

**Ω-notation:**

The notation Ω(n) is the formal way to express the lower bound of an algorithm's running time. It measures the best case time complexity or the best amount of time an algorithm can possibly take to complete.

**o-notation:**

We use o-notation to denote an upper bound that is not asymptotically tight. The asymptotic upper bound provided by O-notation may or may not be asymptotically tight.

**ω-notation:**

By analogy, ω-notation is to Ω-notation as o-notation is to O-notation. We use ω-notation

to denote a lower bound that is not asymptotically tight.