Design and Analysis of Algorithms

Lecture-1

Dharmendra Kumar (Associate Professor)
Department of Computer Science and Engineering
United College of Engineering and Research,
Prayagraj

Syllabus

Unit-I:

Introduction: Algorithms, Analyzing Algorithms, Complexity of Algorithms, Growth of Functions, Performance Measurements, Sorting and Order Statistics - Shell Sort, Quick Sort, Merge Sort, Heap Sort, Comparison of Sorting Algorithms, Sorting in Linear Time.

Unit-II:

Advanced Data Structures: Red-Black Trees, B–Trees, Binomial Heaps, Fibonacci Heaps, Tries, Skip List.

Syllabus

Unit-III:

Divide and Conquer with Examples Such as Sorting, Matrix Multiplication, Convex Hull and Searching. Greedy Methods with Examples Such as Optimal Reliability Allocation, Knapsack, Minimum Spanning Trees – Prim's and Kruskal's Algorithms, Single Source Shortest Paths - Dijkstra's and Bellman Ford Algorithms.

Syllabus

Unit-IV:

Dynamic Programming with Examples Such as Knapsack. All Pair Shortest Paths – Warshal's and Floyd's Algorithms, Resource Allocation Problem. Backtracking, Branch and Bound with Examples Such as Travelling Salesman Problem, Graph Coloring, n-Queen Problem, Hamiltonian Cycles and Sum of Subsets.

Unit-V:

Selected Topics: Algebraic Computation, Fast Fourier Transform, String Matching, Theory of NP Completeness, Approximation Algorithms and Randomized Algorithms

Text books

1. Thomas H. Coreman, Charles E. Leiserson and Ronald L. Rivest, “Introduction to Algorithms”, Printice Hall of India.

2. E. Horowitz & S Sahni, "Fundamentals of Computer Algorithms",

Course Outcome

CO1	Design new algorithms, prove them correct, and analyze their asymptotic and absolute runtime and memory demands.
CO2	Find an algorithm to solve the problem (create) and prove that the algorithm solves the problem correctly (validate).
CO3	Understand the mathematical criterion for deciding whether an algorithm is efficient, and know many practically important problems that do not admit any efficient algorithms.
CO4	Apply classical sorting, searching, optimization and graph algorithms.
CO5	Understand basic techniques for designing algorithms, including the techniques of recursion, divide-and-conquer, and greedy.

Definition of Algorithm

❖An **algorithm** is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output.

❖An **algorithm** is thus a sequence of computational steps that transform the input into the output.

❖An **algorithm** is a set of steps to accomplish or complete a task that is described precisely enough that a computer can run it**.**

Characteristics of an algorithm

- **Input:** Algorithm must contain 0 or more input values.
- **Output:** Algorithm must contain 1 or more output values.
- **Finiteness:** Algorithm must complete after finite number of steps.
- **Definiteness:** Each instruction of the algorithm must be defined precisely or clearly.
- **Effectiveness:** Every instruction must be basic i.e. simple instruction.

Design Strategy of Algorithm

- Divide and Conquer
- Dynamic Programming
- Branch and Bound
- Greedy Approach
- Backtracking
- Randomized Algorithm

Analysis of algorithms

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

Time complexity: The amount of time required by an algorithm is called time complexity.

Space complexity: The amount of space/memory required by an algorithm is called space complexity.

Analysis of algorithms

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 **n**.
Best case − The minimum number of steps taken on any instance of size **n**.
Average case − An average number of steps taken on any instance of size **n**.

Running time of an algorithm

- **Running time of an algorithm is** the time taken by the algorithm to execute it successfully.
- The ***running time*** of an algorithm on a particular input is the number of primitive operations or "steps" executed.
- It is convenient to define the notion of step so that it is as machine independent as possible.
- It is denoted by **T(n)**, where n is the input size.

Pseudo code conventions

We use the following conventions in our pseudo code.

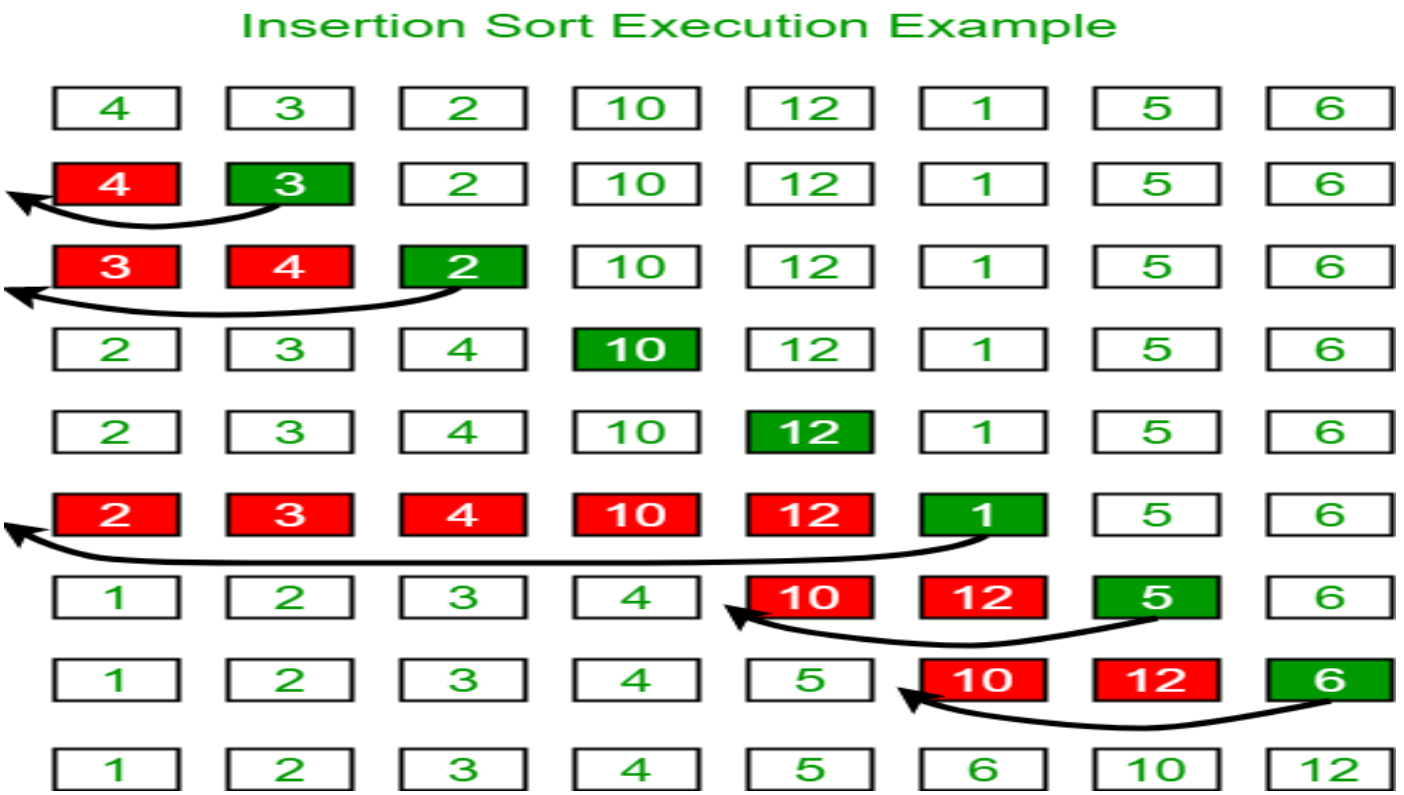
- Indentation is used to indicate a block.
- The looping constructs **while**, **for**, and **repeat-until** and the **if-else** conditional construct have interpretations similar to those in C.
- The symbol "//" indicates that the remainder of the line is a comment.
- A multiple assignment of the form i←j ← e assigns to both variables i and j the value of expression e; it should be treated as equivalent to the assignment j ← e followed by the assignment i ← j .

Pseudo code conventions

- We access array elements by specifying the array name followed by the index in square brackets. For example, A[i] indicates the i^{th} element of the array A.
- The notation ".." is used to indicate a range of values within an array. Thus, A[1..j] indicates the sub-array of A consisting of the j elements A[1], A[2],……,A[j].
- The Boolean operators "and" and "or" are used.
- A **return** statement immediately transfers control back to the point of call in the calling procedure.

Insertion Sort

Ex. Sort the following elements by insertion sort:- 4, 3, 2, 10, 12, 1, 5, 6.



Insertion Sort Algorithm

Insertion_Sort(A)

1. n← length[A]
2. for j←2 to n
    1. a←A[j]
    2. //Insert A[j] into the sorted sequence A[1 .. j-1].
    3. i← j-1
    4. While i > 0 and a < A[i]
        1. A[i+1]←A[i]
        2. i← i-1
    5. A[i+1] ← a

Insertion Sort Algorithm Analysis

Insertion_Sort(A)

Instruction	cost	times
n← length[A]	c_1	1
for j←2 to n	c_2	n
a←A[j]	c_3	n-1
//Insert A[j] into the sorted sequence A[1 .. j-1].	0	n-1
i← j-1	c_4	n-1
While i > 0 and a < A[i]	c_5	$\sum_{j=2}^{n} t_j$
A[i+1]←A[i]	c_6	$\sum_{j=2}^{n}(tj\text{-}1)$
i← i-1	c_7	$\sum_{j=2}^{n}(tj\text{-}1)$
A[i+1] ← a	c_8	n-1

Time complexity of Insertion sort

$$T(n) = c_1.1 + c_2.n + c_3.(n-1) + c_4.(n-1) + c_5.\sum_{j=2}^{n} t_j + c_6.\sum_{j=2}^{n}(tj-1) + c_7.\sum_{j=2}^{n}(tj-1) + c_8.(n-1) \ldots\ldots\ldots\ldots(1)$$

Here, there will be three case.

(1) Best case

(2) Worst case

(3) Average case

Time complexity of Insertion sort

Best case: This case will be occurred when data is already sorted.

In this case, value of t_j will be 1. Put the value of t_j in equation (1) and find T(n). Therefore,

$$T(n) = c_1.1 + c_2.n + c_3.(n-1) + c_4.(n-1) + c_5.\sum_{j=2}^{n} 1$$
$$+ c_6.\sum_{j=2}^{n}(1-1) + c_7.\sum_{j=2}^{n}(1-1) + c_8.(n-1)$$
$$= (c_2+c_3+c_4+c_5+c_8).n + (c_1 - c_3 - c_4 - c_5 - c_8)$$
$$= an + b$$

Clearly T(n) is in linear form, therefore,

$$\mathbf{T(n) = \theta(n)}$$

Time complexity of Insertion sort

Worst case: This case will be occurred when data is in reverse sorted order.

In this case, value of t_j will be j. Put the value of t_j in equation (1) and find T(n). Therefore,

$$T(n) = c_1.1 + c_2.n + c_3.(n-1) + c_4.(n-1) + c_5.\sum_{j=2}^{n} j$$
$$+ c_6.\sum_{j=2}^{n}(j-1) + c_7.\sum_{j=2}^{n}(j-1) + c_8.(n-1)$$
$$= c_1.1 + c_2.n + c_3.(n-1) + c_4.(n-1) + c_5.\ (n+2).(n-1)/2$$
$$+ c_6.n(n-1)/2 + c_7.\ n(n-1)/2 + c_8.(n-1)$$
$$= (c_5+c_6+c_7).n^2/2 + (c_2+c_3+c_4+(c_5 - c_6 - c_7)/2+c_8).n +$$
$$(c_1 - c_3 - c_4 - c_5 - c_8)$$
$$= an^2 + bn + c$$

Clearly T(n) is in quadratic form, therefore, $\mathbf{T(n) = \theta(n^2)}$

Time complexity of Insertion sort

Average case: This case will be occurred when data is in any order except best and worst case.

In this case, value of t_j will be j/2. Put the value of t_j in equation (1) and find T(n). Therefore,

$$T(n) = c_1.1 + c_2.n + c_3.(n-1) + c_4.(n-1) + c_5.\sum_{j=2}^{n} (j/2)$$
$$+ c_6.\sum_{j=2}^{n}(j/2-1) + c_7.\sum_{j=2}^{n}(j/2-1) + c_8.(n-1)$$
$$= c_1.1 + c_2.n + c_3.(n-1) + c_4.(n-1) + c_5.\ (n+2).(n-1)/4$$
$$+ c_6.(n-2)(n-1)/4 + c_7.\ (n-2)(n-1)/4 + c_8.(n-1)$$
$$= (c_5+c_6+c_7).n^2/4 + (c_2+c_3+c_4+(c_5 - 3c_6 - 3c_7)/4+c_8).n +$$
$$(c_1 - c_3 - c_4 - c_5/2 + c_6/2 + c_7/2 - c_8)$$
$$= an^2 + bn + c$$

Clearly T(n) is in quadratic form, therefore, $\mathbf{T(n) = \theta(n^2)}$