DESIGN AND ANALYSIS OF ALGORITHMS

UNIT –I

1. What is an Algorithm?

An algorithm is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time

2. What are Sequential Algorithms?

The central assumption of the RAM model is that instructions are executed one after another, one operation at a time. Accordingly, algorithms designed to be executed on such machines are called Sequential algorithms.

3. What is Algorithm Design Technique?

An algorithm design technique is a general approach to solving problems algorithmically that is applicable to a variety of problems from different areas of computing.

4. Define Pseudo code.

A Pseudo code is a mixture of a natural language and programming language like constructs. A pseudo code is usually more precise than a natural language, and its usage often yields more succinct algorithm descriptions.

5. Define Flowchart.

A method of expressing an algorithm by a collection of connected geometric shapes containing descriptions of the algorithm's steps.

6. Explain Algorithm's Correctness

To prove that the algorithm yields a required result for every legitimate input in a finite amount of time.

Example: Correctness of Euclid's algorithm for computing the greatest common divisor stems from correctness of the equality gcd (m, n) = gcd (n, m mod n).

7. What is Efficiency of algorithm?

Efficiency of an algorithm can be precisely defined and investigated with mathematical rigor. There are two kinds of algorithm efficiency

1) Time Efficiency – Indicates how fast the algorithm runs

2) Space Efficiency – Indicates how much extra memory the algorithm needs.

8. What is generality of an algorithm?

It is a desirable characteristic of an algorithm. Generality of the problem the algorithm solves is sometimes easier to design an algorithm for a problem posed in more general terms.

9. What is algorithm's Optimality?

Optimality is about the complexity of the problem that algorithm solves. What is the minimum amount of effort any algorithm will need to exert to solve the problem in question is called algorithm's Optimality.

10. What do you mean by ″Worst case-Efficiency" of an algorithm?

The ″Worst case-Efficiency" of an algorithm is its efficiency for the worst-case input of size n, which is an input (or inputs) of size n for which the algorithm runs the longest among all possible inputs of that size.

Ex: if you want to sort a list of numbers in ascending order when the numbers are given in descending order. In this running time will be the longest.

11. What do you mean by ″Best case-Efficiency" of an algorithm?

The ″Best case-Efficiency" of an algorithm is its efficiency for the Best-case input of size n, which is an input(or inputs) of size n for which the algorithm runs the fastest among all

possible inputs of that size.
Ex: if you want to sort a list of numbers in ascending order when the numbers are given in ascending order. In this running time will be the smallest.

12. Define the ″Average-case efficiency" of an algorithm?

The ″Average-case efficiency" of an algorithm is its efficiency for the input of size n, for which the algorithm runs between the best case and the worst case among all possible inputs of that size.

13. How to measure the algorithm's efficiency?

It is logical to investigate the algorithm's efficiency as a function of some parameter n indicating the algorithm's input size.
Example: It will be the size of the list for problems of sorting, searching, finding the list's smallest element, and most other problems dealing with lists.

14. What is called the basic operation of an algorithm?

The most important operation of the algorithm is the operation contributing the most to the total running time is called basic operation of an algorithm.

15. How to measure an algorithm's running time?

Let C_{op} be the time of execution of an algorithm's basic iteration on a particular computer and let C (n) be the number of times this operation needs to be executed for this algorithm. Then we can estimate the running time T(n) of a program implementing this algorithm on that computer by the formula
$T(n) \approx C_{op} C(n)$

16. Define order of growth.

The efficiency analysis framework concentrates on the order of growth of an algorithm's basic operation count as the principal indicator of the algorithm's efficiency. To compare and rank such orders of growth we use three notations
1) O (Big oh) notation
2) Ω (Big Omega) notation &
3) Θ (Big Theta) notation

17. Define Big oh notation May/June 2006, April/May 2008

A function t(n) is said to be in O(g(n)) denoted t(n) ε O (g(n)), if t(n) is bounded above by some constant multiple of g(n) for all large n, i.e., if there exist some positive constant c and some non negative integer n_0 such that
$T(n) < c\ g(n)$ for $n > n_0$

18. Prove that $100n+5 \in O(n^2)$?

Clearly $100n+5 \leq 100n+n$ (for all $n \geq 5$) $= 101n \leq 101n^2$
By choosing $n_0=5$ and c=101 we find that $100n+5 \in O(n^2)$.

19. Define Ω notation

A function t(n) is said to be in Ω (g(n)), denoted $t(n) \in \Omega(g(n))$, if t(n) is bounded below by some positive constant multiple of g(n) for all large n, i.e., if there exist some positive constant c and some non negative integer n_0 such that
$T(n) < c\ g(n)$ for $n > n_0$

20. Prove that $n^3 \in \Omega(n^2)$?

Clearly $n^3 \geq n^2$ for all $n \geq 0$. i.e., we can select c=1 and n0=0.

21. Define Θ - notation

A function t(n) is said to be in Θ(g(n)), denoted $t(n) \in \Theta(g(n))$, if t(n) is bounded both

above and below by some positive constant multiples of g(n) for all large n, i.e., if there exist some positive constant c_1 and c_2 and some non negative integer n_0 such that
$c_2 g(n) < t(n) < c_1 g(n)$ for $n > n_0$

22. Prove that(½)n(n-1) ∈ Θ(n^2)

$1/2n(n-1)=(1/2)n^2-1/2n \le 1/2\ n^2$ for all n≥0.(we have proved upper inequality) now $1/2n(n-1)=(1/2)n^2-1/2n \ge (1/2)n^2-1/2n*1/2n$(for all n≥2)$=1/4\ n^2$ hence we can select c2=1/4,c1=1/2 and n0=2.

23. What is the use of Asymptotic Notations?

The notations O, Ω and Θ and are used to indicate and compare the asymptotic orders of growth of functions expressing algorithm efficiencies.

PART-B

1. (a) Describe the steps in analyzing & coding an algorithm. (10)
(b) Explain some of the problem types used in the design of algorithm. (6)
2. (a) Discuss the fundamentals of analysis framework . (10)
(b) Explain the various asymptotic notations used in algorithm design. (6)
3. (a) Explain the general framework for analyzing the efficiency of algorithm. (8)
(b) Explain the various Asymptotic efficiencies of an algorithm. (8)
4. (a) Explain the basic efficiency classes. (10)
(b) Explain briefly the concept of algorithmic strategies. (6)
5. Describe briefly the notions of complexity of an algorithm. (16)
6. (a) What is Pseudo-code? Explain with an example. (8)
(b) Find the complexity C(n) of the algorithm for the worst case, best case and average case.(Evaluate average case complexity for n=3,Where n is the number of inputs) (8)
7. Set up & solve a recurrence relation for the number of key comparisons made by above pseudo code. (4)

UNIT II

1) Explain divide and conquer algorithms

Divide and conquer is probably the best known general algorithm design technique. It work according to the following general plan
i)A problem's instance is divided into several smaller instances of the same problem, ideally of about the same size.
ii) The smaller instances are solved
iii) If necessary, the solutions obtained for the smaller instances are combined to get a solution to the original problem

2) Define Merge Sort

Merge sort is a perfect example of a successful application of the divide and conquer technique. It sorts a given array A[0...n-l] by dividing it into two halves A[0...[n/2] - 1] and A[[n/2]....n-l], sorting each of them recursively, and then merging the two smaller sorted arrays into a single sorted one.

3) Define Binary Search

Binary Search is remarkably efficient algorithm for searching in a sorted array. It works by comparing a search key K with the array's middle element A[m]. If they match, the algorithm stops; Otherwise, the same operation is repeated recursively for the first half of the array if K< A[m] and for the second half if K > A[m]
A[0].....A[m-l] A[m] A[m+l]..... A[n-l]

4) What can we say about the average case efficiency of binary search?

A sophisticated analysis shows that the average number of key comparisons made by binary search is only slightly smaller than that in the worst case

$C_{avg}(n) \approx \log_2 n$

5) How divide and conquer technique can be applied to binary trees?

Since the binary tree definition itself divides a binary tree into two smaller structures of the same type, the left subtree and the right subtree, many problems about binary trees can be solved by applying the divide-conquer technique.

6) Explain Internal and External Nodes

To draw the tree's extension by replacing the empty subtrees by special nodes.The extra nodes shown by little squares are called external. The original nodes shown by littile circles are called internal.

7) Define the Internal Path Length

The Internal Path Length I of an extended binary tree is defined as the sum of the lengths of the paths - taken over all internal nodes- from the root to each internal node.

8) Define the External Path Length

The External Path Length E of an extended binary tree is defined as the sum of the lengths of the paths - taken over all external nodes- from the root to each external node.

11) Explain about greedy technique

The greedy technique suggests constructing a solution to an optimization problem through a sequence of steps, each expanding a partially constructed solution obtained so far, until a complete solution to the problem is reached. On each step, the choice made must *be feasible, locally optimal and irrevocable.*

PART-B

1) Write a pseudo code for divide & conquer algorithm for merging two sorted arrays in to a single sorted one.Explain with example. (12)

2) Construct a minimum spanning tree using Kruskal's algorithm with your own example. (10)

3) Explain about Knapsack Problem with example

4) Explain Dijikstra algorithm (8)

5) Define Spanning tree.Discuss design steps in Prim's algorithm to construct minimum spanning
tree with an example. (16)

6) Explain Kruskal's algorithm. (8)

7) Explain about binary search with example?

UNIT III

1)Define Dynamic Programming

Dynamic programming is a technique for solving problems with overlapping problems. Typically, these subproblems arise from a recurrence relating a solution to a given problem with solutions to its smaller subproblems of the same type. Rather than solving overlapping subproblems again and again, dynamic programming suggests solving each of the smaller sub problems only once and recording the results in a table from which we can then obtain a solution to the original problem.

2) Define Binomial Coefficient

n

The Binomial Coefficient, denoted C(n,k) or k is the number of Combinations(subsets) of k elements from an n-element set(0<k<n). The name "binomial

coefficients" comes from the participation of these numbers in the so called binomial formula $(a+b)_n = (n,0)a_n + \ldots\ldots\ldots + C(n,i)a_{n-i}b_i + \ldots\ldots\ldots + C(n,n)b_n$

3) Define Transitive closure

The transitive closure of a directed graph with n vertices can be defined as the n by n Boolean matrix T = {ti,}, in which the element in the i_{th} row (1<i<n) and the j_{th} column (l<j<n) is 1 if there exists a non trivial directed path from the i_{th} vertex to j_{th} vertex ; otherwise , t_{ij} is 0.

4) Explain Warshalls algorithm

Warshall's algorithm constructs the transitive closure of a given digraph with n vertices through a series of n by n Boolean matrices $R_{(0)}$, ……, $R_{(k-l)}R_{(k)}$,…….., $R_{(n)}$
Each of these matrices provides certain information about directed paths in the digraph.

4) Explain All-pair shortest-paths problem

Given a weighted connected graph (undirected or directed), the all pairs shortest paths problem asks to find the distances(the lengths of the shortest path) from each vertex to all other vertices.

5) Explain Floyd's algorithm

It is convenient to record the lengths of shortest paths in an n by n matrix D called the distance matrix: the element d_{ij} in the i_{th} row and the jth column of this matrix indicates the length of the shortest path from the i_{th} vertex to the j_{th} vertex . We can generate the distance matrix with an algorithm that is very similar to warshall's algorithm. It is called Floyd's algorithm.

6) What does Floyd's algorithm do?

It is used to find the distances (the lengths of the shortest paths) from each vertex to all other vertices of a weighted connected graph.

7) Explain principle of Optimality

It says that an optimal solution to any instance of an optimization problem is composed of optimal solutions to its subinstances.

8) Explain Optimal Binary Search Trees

One of the principal application of Binary Search Tree is to implement the operation of searching. If probabilities of searching for elements of a set are known, it is natural to pose a question about an optimal binary search tree for which the average number of comparisons in a search is the smallest possible.

9) Explain Knapsack problem

Given n items of known weights $w_1, w_2 \ldots\ldots\ldots w_n$ and values $v_1, v_2 \ldots\ldots\ldots v_n$ and a knapsack of capacity W, find the most valuable subset of the items that fit into the knapsack.(Assuming all the weights and the knapsack's capacity are positive integers the item values do not have to be integers.)

10) Explain the Memory Function technique

The Memory Function technique seeks to combine strengths of the top down and bottom-up approaches to solving problems with overlapping subproblems. It does this by solving, in the top-down fashion but only once, just necessary sub problems of a given problem and recording their solutions in a table.

11) Explain "Traveling salesman problem"?

A salesman has to travel n cities starting from any one of the cities and visit the remaining cities exactly once and come back to the city where he started his journey in such a manner that either the distance is minimum or cost is minimum. This is known as traveling salesman problem.

PART-B

1) Solve the all pair shortest path problem for the diagraph with the weighted matrix given below:-

	a	b	c	d
a	0	∞	3	∞
b	2	0	∞	∞
c	∞	7	0	1
d	6	∞	∞	0(16)

2) Explain Warshall's & Floyd's Algorithm. (16)
3) Explain about Multistage graphs with example.
4) Define optimal binary search trees with example.
5) Explain 0/1 knapsack problem with example.
6) Discuss the solution for Travelling salesman problem using branch & bound technique. (16)

UNIT IV

l) Explain Backtracking

The principal idea is to construct solutions one component at a time and evaluate such partially constructed candidates as follows.

> If a partially constructed solution can be developed further without violating the problem's constraints, it is done by taking the first remaininig legitimate option for the next component.

> If there is no legitimate option for the next component, no alternatives for any remaining component need to be considered.

In this case, the algorithm backtracks to replace the last component of the partially constructed solution with its next option

2) Explain State Space Tree

If it is convenient to implement backtracking by constructing a tree of choices being made, the tree is called a state space tree. Its root represents an initial state before the search for a solution begins.

3) Explain promising and nonpromising node

A node in a state space tree is said to be promising if it corresponds to a partially constructed solution that may still lead to a complete solution;otherwise it is called nonpromising

4) Explain n-Queens problem

The problem is to place n queens on an n by n chessboard so that no two queens attack each other by being in the same row or same column or on the same diagonal.

5) Explain Subset-Sum Problem

We consider the subset-sum problem: Find a subset of a given set $S=\{S_1,S_2,.....S_n\}$ of n positive integers whose sum is equal to a given positive integer d.

6) Explain Branch and Bound Technique

Compared to backtracking, branch and bound requires

The idea to be strengthened further if we deal with an optimization problem, one that seeks to minimize or maximize an objective function, usually subject to some constraints.

7) Define Feasible Solution

A feasible solution is a point in the problem's search space that satisfies all the problem's constraints.

Ex: A Hamiltonian Circuit in the traveling salesman problem. A subset of items whose total weight does not exceed the knapsack's Capacity

8) Define Optimal solution

Is a feasible solution with the best value of the objective function

Eg: The shortest Hamiltonian Circuit
The most valuable subset of items that fit the knapsack

9)Mention two reasons to terminate a search path at the current node in a state-space tree of a branch and bound algorithm.

The value of the node's bound is not better than the value of the best solution seen so far.
The node represents no feasible solutions because the constraints of the problem are already violated.

10) Explain ″Graph coloring" problem.

The graph coloring problem asks us to assign the smallest number of colors to vertices of a graph so that no two adjacent vertices are the same color.

11) Explain Knapsack Problem

Find the most valuable subset of n items of given positive integer weights and values that fit into a knapsack of a given positive integer capacity.

PART-B

1. Explain the 8-Queen's problem & discuss the possible solutions. (16)
2. Solve the following instance of the knapsack problem by the branch & bound algorithm. (16)
3. Apply backtracking technique to solve the following instance of subset sum problem : S={1,3,4,5} and d=11 (16)
5. Explain subset sum problem & discuss the possible solution strategies using backtracking.(16)
6. Explain Graph coloring with example.
7. Explain about Knapsack Problem using back tracking with example.

UNIT V

1) Define tractable and intractable problems

Problems that can be solved in polynomial time are called tractable problems, problems that cannot be solved in polynomial time are called intractable problems.

2) Explain the theory of computational complexity

A problem's intractability remains the same for all principal models of computations and all reasonable input encoding schemes for the problem under consideration

3)Explain class P problems

Class P is a class of decision problems that can be solved in polynomial time by(deterministic) algorithms. This class of problems is called polynomial.

4)Explain undecidable problems

If the decision problem cannot be solved in polynomial time, and if the decision problems cannot be solved at all by any algorithm. Such problems are called Undecidable.

5) Explain the halting problem

Given a computer program and an input to it,determine whether the program will halt on that input or continue working indefinitely on it.

6) Explain class NP problems

Class NP is the class of decision problems that can be solved by nondeterministic polynomial algorithms.Most decision problems are in NP. First of all, this class includes all the problems in P. This class of problems is called Nondeterministic polynomial.

7)Explain NP-complete problems

A decision problem d is said to be NP-complete if
1) it belongs to class NP
2) every problem in NP is polynomially reducible to D.

8)When a decision problem is said to be polynomially reducible

A decision problem Dl is said to be polynomially reducible to a decision problem D2 if there exists a function t that transforms instances of Dl to instances ofD2 such that
i) t maps all yes instances of d1 to yes instances odf d2 and all no instances of dl to no instances ofd2
ii) t is computable by a polynomial time algorithm

9) Define a Heuristic

A heuristic is a common-sense rule drawn from experience rather than from a mathematically proved assertion.
Ex: Going to the nearest unvisited city in the traveling salesman problem is a good illustration for Heuristic

10) Explain NP-Hard problems

The notion of an NP-hard problem can be defined more formally by extending the notion of polynomial reducability to problems that are not necessary in class NP including optimization problems.

11)Define Traversals.

When the search necessarilyinvolves the examination of every vertex in the object being searched it is called a traversal.

12)List out the techniques for traversals in graph.

Breadth first search
Depth first search

13)What is articulation point.

A vertex v in a connected graph G is an articulation point if and only if the deletion of vertex v together with all edged incident to v disconnects the graph in to two or more nonempty components.

PART-B

1. Give a suitable example & explain the Breadth first search & Depth first search. (16)
2. Explain about biconnected components with example.
3. Briefly explain NP-Hard and NP-Completeness with examples.
4. Explain about 0/1 Knapsack Problem using branch and bound with example.