**Course code: CS241**

**Course title: Design and Analysis of Algorithm**

**SYLLABUS**

**Module I - Algorithms and Complexity**

Introduction, Algorithm Complexity and various cases using Insertion Sort, Asymptotic Notations, Time complexity of Recursive Algorithm, Solving Recurrences using Iterative, Recursion Tree and Master Theorem. **(8L)**

**Module II - Divide and Conquer**

Discussion of basic approach using Binary Search, Merge Sort , Quick Sort , Selection in Expected linear time, Maximum Subarray , Matrix Multiplication , Introduction of Transform and Conquer and AVL Tree . **(8L)**

**Module III - Dynamic Programming**

Introduction and Approach, Rod Cutting, LCS, Optimal BST, Transitive closure and All-pair Shortest Path, Travelling Salesperson Problem. (8L)

**Module IV - Greedy and other Design Approaches**

Introduction to greedy using fractional knapsack, Huffman Code, Minimum Spanning Tree – Prim and Kruskal, Single Source Shortest Path Dijkstra’s and Bellman-Ford, Introduction to Backtracking using N-Queens problem, Introduction to Branch and Bound using Assignment Problem or TSP. **(8L)**

**Module V - NP Completeness and Other Advanced Topics**

Non-deterministic algorithms – searching and sorting, Class P and NP, Decision and Optimization problem, Reduction and NPC and NPH, NP Completeness proof for: SAT, Max- Clique, Vertex Cover, Introduction to Randomized Algorithms, Introduction to Approximation Algorithms. (8L)

**Text Book:**

1. Cormen Thomas H. et al., Introduction to Algorithms. 3rd Edition, PHI Learning, latest edition.(T1)

**Reference Books:**

2 Horowitz E., Sahani, Fundamentals of Computer Algorithms, Galgotia Publication Pvt. Ltd. (R1)

3 Dave and Dave, Design and Analysis of Algorithms, 2nd Edition, Pearson. (R2)

4 Goodrich, Tamassia. Algorithm Design. Wiley. (R3)

**QUESTION BANK**

**Department: CSE Class: BE Question Bank No:**

**Semester: VI Subject: Design and Analysis of Computer Algorithms**

**Subject Code: CP6101**

**Date of Question Bank given to Students: SP09**

**MODULE-I**

**Algorithm design and analysis fundamentals**

1.a)Discuss the steps of program development.

b)Discuss the concepts of top-down program development.

c) Discuss the concept of structured flowcharts.

2.a) “The perfomance of an algorithm varies with the nature/ order of the input data” . Justify.

b) Elaborate the notions of Best case, worst case and average case of time complexities of an algorithm, using a suitable example e.g. MERGESORT.

c) Prove correctness of INSERTIONSORT, BUBBLESORT.

3. a) What do you mean by space-time trade-off?

b) Why iteration is preferable over recursion? When recursion is useful?

4. What do you mean by growth function of an algorithm? How asymptotic notations are helpful in representing the upper bound and the lower bound of growth of functions?

5. 6. Define the Asymptotic notations. How they are helpful in expressing and comparing the efficiency of algorithms.

7. Consider the function f(n) = ½ n3 +½ n2 .

What is the maximum order of growth of the function?

OR Determine the upper bound on growth of the function.

OR Represent the order of the fucntion in O-notation.

8. Estimate the lower bound on growth of the fucntion f(n) = n3 log n.

OR What is the minimum order of growth of the function?

OR Represent the order of the fucntion in -notation.

9. Represent the order of the fucntion in -notation f(n) = 3 log n + log log n

10. Express the growth of the following functions using O,, and  notations:

n n n

i) (n+a)b ii)  i=1i iii) i=1log (i) iv) i=1ik.

v) 2n3 + 4n2 log(n)

11. Verify whether f(n)= 3n 2 + 100 n + 6 is ( n3).

12. Analyse the run time of the following algorithms

1. Loop1(n) 3) Loop3

S:= 0 P:= 1

for i := 1 to n do for i := 1 to 2n do

S := S + i. P:= P \* I

1. Loop2(n) 4) Loop2

S:= 0 P:= 1

for i := 1 to n do for i := 1 to n2 do

for j:= 1 to i do for j:= 1 to i do S := S + i. S:= S+i

13. a) How many multiplications are performed by the function for finding Factorial()?

1. How many additions are performed between pairs of matrix elements by the function Add() for adding matrix elements?
2. Write function to reverse a linked list. Prove its correctness. Determine its time complexity.
3. How many Swap operations are performed by the function for Transpose of matrix elements?
4. Algorithm A uses 10 n logn operations, while algorithm B uses n2 operations. Determine the value nb such that A is better than B for n >= n0

14. a)Write a recursive function for sequential search. Analyse its compelxity. Compare it with binary search.

b) Suppose that each node in a BST also has the field *leftsize*. Design an algorithm to insert an element x into such a BST. The complexity of your algorithm should be O(h), where h is the height of the search tree. Verify it.

15. Write following algorithms and determine their time complexities:

a) a recursive algorithm for finding both the maximum and the minimum elements in an array A of n elements.

b) an O(n) time algorithm for finding the maximum element in a give list. Can we have a better algorithm for the purpose.

c) finding path between two given vertices in a graph.

d) recursive algorithm to list permutations of the elements of a given set.

17. Prove that running time of an algorithm is (g(n)), iff the worst case running time is (g(n)).

18. If f(n) and g(n) are asymptotically non negative functions, then using the definition of the -notation prove that

max (f(n), g(n)) = (f(n) + g(n))

or f(n) + g(n) =  (min(f(n), g(n)))

19. For f(n) and g(n), asymptotically non negative functions

if f(n) = (g(n)), then g(n) = (f(n)). Verify.

20. a) Give an example of a positive function f(n), such that f(n) is neither O(n) nor (n).

b) Let f(n) and g(n) be functions where n is a natural number. Define the O-notation and state examples where f(n) = O(g(n)) but g(n) != O(f(n)).

21. An array A contains n-1 unique integers in the range [0, n-1], that is there is one number from this range that is not in A. Design an O(n)-time algorithm for finding that number. You are allowed to use only O(1) additional space besides the array itself.

22. Give a recursive algorithm to compute the product of two positive integers m and n using only addition.

23. Suppose that each row of an n x n array A consists of 1’s and 0’s such that, in any row I of A, all the 1’s come before any 0’s in that row. Suppose further that the number of 1’s in row I is at least the number in row i+1, for i= 0,1, …,n-2. Assuming A is already in memory, describe a method running in O(n) time ( not O(n2) time ) for finding the row of A that contains the most 1’s.

24. Solve the following recurrences

1) T(n) = (1) if n = 1

2T(n/2) + (n) if n > 1

2) T(n) = 1 if n = 1

T(n-1) +n if n > 1

3) T(n) = 1 if n = 0

2T(n-1) + 2n f n > 1

25. Solve the following recurrence by master method

1) T(n) = 9T(n/3) + n 2) T(n) = T( 2n/3 +1)

3) T(n) = 2T(n/2) + nlog n 4) T(n) = 2T(n/2) + n3

5) T(n) = 2T(1/√n) + n2

26. a) What do you mean by randomized algorithms? Discuss with example the use of randomization.

b) Write the randomized algorithms for following:

i) selection ii) primality test iii) quicksort

**Standard Techniques of Algorithm Design and typical problems.**

**MODULE-II**

**Divide and Conquer**

1. What do you mean by divide and conquer technique? Explain it with an example.
2. a) Prove the theorem- “ If n is in the range [2k-1, 2k], then BinSearch makes at most k element comparisons for a successful search and either k-1 or k comparisons for an unsuccessful search., i.e. the time for a successful search is O(log n) and for an unsuccessful search is (logn).
3. Suppose that each row of an n size array A consists of 1’s and 0’s such that, in any row I of A, all the 1’s come before any 0’s in that row. Assuming A is already in memory, describe a method running in O( lgn ) time for finding the position of A that contains the last 1.
4. Suppose that each row of an n x n array A consists of 1’s and 0’s such that, in any row I of A, all the 1’s come before any 0’s in that row. Assuming A is already in memory, describe a method running in O(n lgn) time ( not O(n2) time ) for finding the row of A that contains the most 1’s.
5. Discuss an O(n log n) algorithm for sorting an array of n elements.

*Hint:( Merge Sort )*

b) Show how MergeSort sorts a data set A= ( 310, 285, 179, 652, 351, 423, 861,254). Can we say that the time for MergeSort is O(n log n)?

1. Devise a binary search algorithm that splits the set not into two sets of (almost) equal sizes but into two sets, one of which is twice the size of the other. How does this algorithm compare with binary search?
2. A way to sort a file of n records is to scan the file, merge consecutive pairs of size one, then merge pairs of size two, and so on. Write an algorithm that carries out this process. Show how the algorithm works on a data set A= ( 310, 285, 179, 652, 351, 423, 861,254).
3. Compare the sorting methods MergeSort1 and QuickSort2 (ref: TextBook Algorithm 3.10 and 3.14 ). Can the two algorithms be made better by use of Insertion-Sort for smaller size of input? Justify.
4. Write an algorithm that multiplies two n x n matrices using less than O(n3) operations. Determine the precise number of multiplications, additions, and array element accesses. [Hint: Strassen’s Matrix Multiplication method]
5. Let x[1:n] and y[1:n] contain two sets of integers, each sorted in nondecreasing order. Write an algorithm that finds the median of te 2n combined elements. What is the time complexity of your algorithm?
6. The k-th *quantiles* of an n-element set are the k-1 elements from the set that divide the sorted set into k equal-sized sets. Give an O(n logk) time algorithm to list the kth quantiles of a set.
7. Given two vectors X=(x1,…, xn) and Y=(y1,…, yn), X<Y if there exists an i, 1<=i<=n, such that xj = yj for 1<=j<I and xi<yi. Given m vectors each of size n, write an algorithm that determines the minimum vector. Analyze time complexity

**MODULE-III**

**Dynamic Programming**

1. Solve the following using dynamic programming:

Maximize: z= 11y1+ 32 y2 + 58 y3

Subject to: y1+3y2+5y3 <= 8

With: all variables nonnegative and integral.

1. a) Discuss the essence of the Dynamic Programming method of solving problems. Explain the concept of “Principle of optimality” and discuss how the problems obeying this principle can be solved?

b) Give a dynamic programming solution to the 0/1 knapsack problem. What is the complexity of this algorithm?

1. State the similarity and differences between the techniques of dynamic programming and divide and conquer. Use an example of a problem for each of the techniques and explain the solution.
2. Justify how the following problems befit the principle of optimality. Hence give a dynamic programming based formulation for each.

i) 0/1 knapsack ii) all pair Shortest paths iii) Multistage Graph

iv) Traveling Salesman problem.

1. Using the concept of principle of optimality, solve the following traveling salesman problem

A B C D

A 0 4 5 7

B 6 0 10 11

C 9 5 0 12

D 8 7 10 0

1. The principle of optimality does not hold for every problem whose solution can be viewed as the result of a sequence of decisions. Find two problems for which the principle does not hold.
2. Discuss an algorithm with example to show how a minimum cost path can be found for a multistage graph, applying the principle of optimality.
3. Explain the dynamic programming technique based formulation for the problem of all pair shortest path. Illustrate its working with an example.
4. Explain the dynamic programming technique based formulation for the problem of optimal binary search tree. Illustrate its working with an example.
5. Explain the dynamic programming technique based formulation for the problem of flow shop scheduling. Illustrate its working with an example.

**BackTracking**

1. State the basic principles of problem solving with backtracking. Show how the problem of 0/1 knapsack be solved with backtracking.
2. Write a backtracking algorithm that finds all m-colorings of a graph. Explain how the same problem can be solved using branch and bound method.
3. Explain the methods of ‘Backtracking’ and of ‘Branch and Bound’. What type of problems can be solved with these methods? Explain with an example, the similarity and differences between these methods.
4. Explain an algorithm for arranging N-queens in non-attacking positions on an NxN chessboard. Comment on how your strategy reduces the number of comparisons.
5. Give a step-to-step trace for solving the n-queen problem for 4 queens.
6. Discuss the backtracking method to solve 8-puzzle problem.
7. Discuss the backtracking method to solve sum of subset problem.

**MODULE-IV**

**Greedy Method**

1. Discuss the basic principle of the greedy method of solving problem. Explain the notions- “*ordering paradigm*” and “*subset paradigm*”. What types of problems can be solved with this method.
2. a) Design an algorithm for the fractional Knapsack problem. Analyze the time complexity of Greedy Knapsack algorithm.

OR

Find an optimal solution to the knapsack instance n =7, m = 15, (p1,p2,p3,…,p7) = (10,5,15,7,6,18,3), and (w1,w2,…,w7) = (2,3,5,7,1,4,1), such that fraction of any object can not be taken.

1. If p1/w1>= p2/w2>= ….>= pn/wn, then Greedy Knapsack generates an optimal solution to the given instance of the knapsack problem.
2. Consider the knapsack problem. We add the requirement that xi = 0,1 , 1<= I<= n; that is an object is either included or not included into the knapsack. We wish to solve the problem

n

max i=1pi xi

subject to n

i=1wi xi <= m and xi = 0 or 1, 1<=i<=n

One greedy strategy is to consider the objects in order of nonincreasing desity pi /wi , and add the object into the knapsack if it fits. Show that this strategy doesn’t necessarily yield an optimal solution.

1. One greedy strategy is to consider the objects in order of nonincreasing desity pi /wi , and add the object into the knapsack if it fits. Show that this strategy doesn’t necessarily yield an optimal solution.
2. Discuss the time complexity of the Tree Vertex Splitting ( TVS) algorithm.
3. a) You are given a set of n jobs. Associated with each job i is a processing time ti and a deadline di by which it must be completed. A feasible schedule is a permutation of the jobs such that if the jobs are processed in that order, then each job finishes by its deadline. Define a greedy schedule to be one in which the jobs are processed in nondecreasing order of deadlines.

b) What is the solution generated by the function JS when n = 7, P = ( 3,5,20,18,1,6,30), and D = (1,3,4,3,2,1,2)?

1. Devise an algorithm that finds a minimum-weight spanning tree for any given weighted graph in time O( |V| + |E| ). Illustrate its function with an example.
2. Prove the correctness of Prims’s algorithm, Kruskal algorithm i.e. Prove that Prim’s and Kruskal’s methods generate minimum-cost spanning trees.
3. State an algorithm for determining whether an undirected graph has a cycle.
4. Let G be a simple graph with n vertices and m edges. Explain why O(log m) is O( log n).
5. Discuss the DFS and BFS schemes for traversing a graph. Determine their time complexities.
6. Devise an algorithm for finding *Reflexive transitive closure* A\* for a given graph A.
7. Show how the greedy method can be applied to find the optimal merge patterns for given data sets. Find an optimal binary merge pattern for ten files whose lengths are 28,32,12,5,84,53,91,35,3, and 11.
8. Discuss a greedy solution to find Single source shortest paths. Illustrate its function with a suitable example.

**Branch and Bound**

1. Explain the basic technique of branch and bound. How does it compare with backtracking?
2. State the basic principles of problem solving with branch and bound. Show how the problem of 0/1 knapsack be solved with this technique.
3. Discuss the backtracking method to solve 15-puzzle problem.
4. Give a solution for the Traveling salesman problem using the Branch and Bound method. Compare its efficiency with that of the solution obtained with dynamic programming method.
5. Devise an algorithm to find all Hamiltonian tours with minimum cost for a given weighted graph. Justify the correctness of your algorithm and comment on its efficiency.
6. Compare and contrast the following techniques of problem solving, i.e. discuss the basic principles of the following techniques of algorithm design; Comment on what types of problems can be solved with each of them.
   1. Divide and Conquer ii)Greedy algorithm iii)Dynamic Programming

iv) Backtracking v)Branch and Bound

1. Discuss the algorithms for insertion and deletion of nodes in

1. B-Tree 2. AVL Tree

1. What do you mean by hill climbing technique? What are subgoal techniques of problem solving?
2. Define the terms P, NP, NP hard, NP Complete. What is the problem of satisfiability? What is its role in the NP-considerations?
3. Explain how the problem of determining Hamiltonian tours in a graph is NP-complete problem.
4. State and justify whether the problem of traveling salesperson belongs to the set of NP-complete problems.
5. Write Short notes on
   * 1. NP-Complete Problem b)NP-Hard Problem
6. Set Cover problem

**MODULE-V**

**NP-Completeness etc**