Geethanjali College of Engineering and Technology

Cheeryal (V), Keesara (M), Ranga Reddy District – 501 301 (T.S)

DESIGN AND ANALYISIS OF ALGORITHMS

COURSE FILE

DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

(2015-2016)

Faculty In charge

Mr.D.Venkateswarulu
Mr.M.Srinivas

HOD-CSE
Dr. S.NagenderKumar

S.No	Topic	Page. No.
1	Cover Page	
2	Syllabus copy	
3	Vision of the Department	
4	Mission of the Department	
5	PEOs and Pos	
6	Course objectives and outcomes	
7	Brief notes on the importance of the course and how it fits into the curriculum	
8	Prerequisites if any	
9	Instructional Learning Outcomes	
10	Course mapping with POs	
11	Class Time Table	
12	Individual Time Table	
13	Lecture schedule with methodology being used/adopted	
14	Detailed notes	
15	Additional topics	
16	University Question papers of previous years	
17	Question Bank	
18	Assignment Questions	
19	Unit wise Quiz Questions and long answer questions	
20	Tutorial problems	
21	Known gaps ,if any and inclusion of the same in lecture schedule	
22	Discussion topics , if any	
23	References, Journals, websites and E-links if any	
24	Quality Measurement Sheets	
A	Course End Survey	
B	Teaching Evaluation	
25	Student List	
26	Group-Wise students list for discussion topic	

Geethanjali College of Engineering and Technology

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(Name of the Subject/Lab Course): Design and Analysis of Algorithms

(JNTU CODE: A40508) Programme: UG

Branch: CSE	Version No:
Year: II	Document Number: GCET/CSE
Semester: 2	No. of Pages:

Classification status (Unrestricted/Restricted)

Distribution List:

Prepared by :

1) Name : D.Venkateswarlu & M.Srinivas

2) Sign :

3) Design : Associate Prof./ Associate Prof

4) Date : 20.12.2014

Verified by :	*For Q.C only
1) Name :	1)Name :
2) Sign :	2) Sign :
3) Design :	3) Design :
4) Date :	4) Date :

Approved by (HOD) :

1) Name:

2) Sign :

3) Date :

Course file contents

II Year B.Tech. CSE II-Sem
DAA SYLLABUS

UNIT-I

INTRODUCTION: Algorithm, Pseudo code for expressing algorithms, Performance analysis, Time complexity and space complexity, Asymptotic Notations: O notation, Omega notation, theta notation, and little o notation .probabilistic analysis and amortized complexity

DIVIDE AND CONQUER: General method, applications – binary search, merge sort, quick sort, Stassen's matrix multiplication.

UNIT-II

SEARCHING AND TRAVERSAL TECHNIQUES : Efficient non-recursive binary tree traversal algorithms, disjoint set operations, union and find algorithms, spanning trees, graph traversals- BFS and DFS , AND/OR graphs, game tree, Connected components, bi-Connected components.

UNIT-III

GREEDY-METHOD : General method, Applications-Job sequencing with deadlines , 0/1 knapsack problem, minimum cost spanning tree, single source shortest path problem.

DYNAMIC PROGRAMMING: General method **,** applications-multistage grapes, optimal binary search trees, 0 /1 knapsack problem, all pairs shortest paths problem, traveling sales person problem , reliability design problem.

UNIT-IV

BACK TRACKING: General method, applications: n-queens problem, sum of sub set problem, graph coloring problem, Hamiltonian cycles.

BRANCH and BOUND: General method, applications: traveling sales person problem, 0 /1 knapsack problem, LC branch and bound, FIFO branch and bound solution

UNIT-V

NP-Hard and NP-Complete problems: Basic concepts, non deterministic algorithms, NP-hard and NP-complete classes, NP- Hard problems,Cook's theorem.

TEXT BOOKS:

1. HOROWITZ and S.SAHNI: "Fundamentals of Algorithms" Galgotia
2. Introduction to algorithms, 2nd edition ,t.H.Cormen, C.E. Leiserson, C.Stein.

Reference Books :

1. Algorithm Design, Micheal P. Goordicoh, Roborto Tamassia
2. The Design and Analysis of Computer Algorithms – Aho/Hop Croft/Ullman – Low Price edition

3. Vision of the Department

To produce globally competent and socially responsible computer science engineers contributing to the advancement of engineering and technology which involves creativity and innovation by providing excellent learning environment with world class facilities.

4. Mission of the CSE Department

1. To be a center of excellence in instruction, innovation in research and scholarship, and service to the stake holders, the profession, and the public.

2. To prepare graduates to enter a rapidly changing field as a competent computer science engineer.

3. To prepare graduate capable in all phases of software development, possess a firm understanding of hardware technologies, have the strong mathematical background necessary for scientific computing, and be sufficiently well versed in general theory to allow growth within the discipline as it advances.

4. To prepare graduates to assume leadership roles by possessing good communication skills, the ability to work effectively as team members, and an appreciation for their social and ethical responsibility in a global setting.

5. PROGRAM EDUCATIONAL OBJECTIVES OF CSE:

PEO-1. To provide graduates with a good foundation in mathematics, sciences and engineering fundamentals required to solve engineering problems that will facilitate them to find employment in industry and / or to pursue postgraduate studies with an appreciation for lifelong learning.

PEO-2. To provide graduates with analytical and problem solving skills to design algorithms, other hardware / software systems, and inculcate professional ethics, inter-personal skills to work in a multi-cultural team.

PEO-3. To facilitate graduates get familiarized with state of the art software / hardware tools, imbibing creativity and Innovation that would enable them to develop cutting-edge technologies of multi-disciplinary nature for societal development.

PROGRAMME OUTCOMES OF CSE:

1. An ability to apply knowledge of mathematics, science and engineering to develop and analyze computing systems.
2. An ability to analyze a problem and identify and define the computing requirements appropriate for its solution under given constraints.
3. An ability to perform experiments to analyze and interpret data for different applications.
4. An ability to design, implement and evaluate computer-based systems, processes, components or programs to meet desired needs within realistic constraints of time and space.
5. An ability to use current techniques, skills and modern engineering tools necessary to practice as a CSE professional.
6. An ability to recognize the importance of professional, ethical, legal, security and social issues and addressing these issues as a professional.
7. An ability to analyze the local and global impact of systems /processes /applications /technologies on individuals, organizations, society and environment.
8. An ability to function in multidisciplinary teams.
9. An ability to communicate effectively with a range of audiences.
10. Demonstrate knowledge and understanding of the engineering, management and economic principles and apply them to manage projects as a member and leader in a team.
11. A recognition of the need for and an ability to engage in life-long learning and continuing professional development
12. Knowledge of contemporary issues.
13. An ability to apply design and development principles in producing software systems of varying complexity using various project management tools.
14. An ability to identify, formulate and solve innovative engineering problems.

6. Course Outcomes

At the end of the course students will be able to

A40508.1. Explain, model, and analyze a given software problem as an algorithm.
A40508.2. Analyze algorithms and estimate their best-case, worst-case and average-case behavior.
A40508.3. Formulate the space needs for the implementation of an algorithm.
A40508.4. Investigate whether the algorithm found is the most efficient.
A40508.5. Prove the correctness of an algorithm.
A40508.6. Explain good principles of algorithm design and apply the same to real word problems;
A40508.7. Explain basic techniques for designing algorithms, including the techniques of recursion, Divide-and-Conquer, and Greedy and apply the same to various problems.
A40508.8. Explain advanced techniques for designing algorithms, including dynamic programming and Backtracking and apply the same to various problems.
A40508.9. Differentiate deterministic and non deterministic algorithms.
A40508.10. Categorize algorithms as NP-Hard, NP-complete

7. Brief notes on the importance of the course and how it fits into the curriculum

The course provides a solid foundation in algorithm design and analysis. Specifically, the students acquire the basic knowledge of graph and matching algorithms, and design algorithms using greedy strategy, divide and conquer approach, dynamic programming, and max flow - min cut theory. This course enables the student to analyze asymptotic runtime complexity of algorithms including formulating recurrence relations. It provides basic knowledge of computational complexity, approximation and randomized algorithms.

The student will have an ability to apply knowledge of mathematics, science and engineering to develop and analyze computing systems, to analyze a problem and identify and define the computing requirements appropriate for its solution under given constraints, to perform experiments to analyze and interpret data for different applications, to design, implement and evaluate computer-based systems, processes, components or programs to meet desired needs within realistic constraints of time and space and to identify, formulate and solve innovative engineering problems.

8. Prerequisites if any

1. C Programming
2. Data Structures

9. Instructional Learning outcomes

Unit-1:

1. Use proper conventions for writing an algorithm.
2. Differentiate different types of Asymptotic notations used for representing complexity of an algorithm
3. Use asymptotic notation to formulate the time and space requirements of algorithms.
4. Perform probabilistic analysis on a given algorithm
5. Perform Amortized analysis on a given algorithm

Unit-2:

1. Apply disjoint set operations
2. Use union and find algorithms
3. Draw spanning trees
4. Determine connected and bi-
5. connected components

Unit-3:

1. Apply the technique of divide and conquer.
2. Write algorithms using divide and conquer technique.
3. Use divide and conquer technique for sorting (quick sort,merge sort)
4. Use divide and conquer technique for matrix multiplication
5. Apply the technique of greedy technique.
6. Write algorithm for sequencing jobs with deadlines using greedy technique.
7. Apply greedy technique to solve 0/1 knapsack problem
8. Find minimum spanning trees using prims and krushkals technique based on greedy approach
9. apply greedy approach for Single source shortest problem
10. Apply the principal of optimality.
11. Write algorithm for matrix chain multiplication
12. Use principal of optimality for optimal binary search trees
13. Solve 0/1 knapsack problem, All pair shortest path algorithm, travelling salesman problem using dynamic programming
14. Do reliability design of resources using principal of optimality.

Unit-4:

1. Use backtracking technique.
2. Analyze which problems can be solved using backtracking technique.
3. Write algorithm to solve 8 queens problem using backtracking.
4. Apply backtracking for sum of subsets problem.
5. Determine the proper coloring of the graph and Hamiltonian cycles using backtracking
6. Apply the technique of LC branch and bound and FIFO branch and bound.
7. Write branch and bound solution for travelling salesman problem
8. Solving 0/1 Knapsack problem using LC branch and bound and FIFO branch and Bound.

Unit-5:

1. Categorize algorithms as NP-Hard and NP-Complete
2. Distinguish deterministic and non deterministic algorithms.
3. Analyze cooks theorem.

10. Course mapping with POs

DAA	PEO1,PEO2	PO1,PO2,PO3,PO4,PO7,PO8,PO10,PO13,PO14

Course Code and Title	**POS**													
A40508 Design and Analysis of Algorithms	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A40508.1 Explain, model, and analyze a given software problem as an algorithm.	**2**	**2**	**2**	**2**			**1**	**1**		**1**			**1**	**2**
A40508.2 Analyze algorithms and estimate their best-case ,worst-case and average-case behavior.	**2**	**2**	**2**	**2**			**1**	**1**		**1**			**1**	**1**
A40508.3 Formulate the space needs for the implementation of an algorithm.	**2**	**2**	**2**	**2**			**1**	**1**		**1**			**1**	**1**
A40508.4 Investigate whether the algorithm found is the most efficient.	**2**	**2**	**2**	**2**			**1**	**1**		**1**			**1**	**1**
A40508.5 Prove the correctness of an algorithm.	**2**	**2**	**2**	**2**			**1**	**1**		**1**			**1**	**1**
A40508.6 Explain good principles of algorithm design and apply the same to real word problems;	**2**	**2**	**2**	**2**			**1**	**1**		**1**			**1**	**2**
A40508.7 Explain basic techniques for designing algorithms, including the techniques of recursion, divide-and-conquer, and greedy and apply the same to various problems.	**2**	**2**	**2**	**2**			**1**	**1**		**1**			**1**	**2**
A40508.8 Explain advanced techniques for designing algorithms, including dynamic programming and Backtracking	**2**	**2**	**2**	**2**			**1**	**1**		**1**			**1**	**2**

11. Class Time Table

13. Lecture schedule with methodology being used/adopted

II-CSE-A

S.No.	No. of Periods	Topics to be covered	Regular/ Additional	Teaching Aids Used LCD/OHP/BB	Expected Date(s)
UNIT-I					
1	1	Introduction to subject course objectives and outcomes	Regular	BB	07.12.2015
2	1	Algorithm, Pseudo code for expressing algorithms	Regular	BB	08.12.2015
3	1	Writing algorithm for sample problems	Regular	BB	
4	1	Performance analysis: Space complexity.	Regular	BB	08.12.2015
5	1	Time complexity: using variable count	Regular	BB	09.12.2015
6	1	Time complexity: using frequency count table	Regular	BB	
7	1	Asymptotic Notation: Big oh, Omega, Theta & Little oh notation	Regular	BB,OHP/LCD,Q&A	10.12.2015
8	1	Examples on Asymptotic Notations and Theorems	**Tutorial-1**	BB,Q&A Learning by Doing,	11.12.2015
9	1	Divide and Conquer: General Method	Regular	BB,OHP/LCD,Q&A,E-Tutorials	14.12.2015
10	1	Recurrence Relations : Substitution method	Regular	BB,OHP/LCD, Internet, Q&A	15.12.2015
11	1	Applications: Binary Search	Regular	BB,OHP/LCD, Learning By Doing, E-Tutorials, Internet, Q&A	16.12.2015
12	1	Merge sort: Example and Algorithm	Regular	BB,OHP/LCD	17.12.2015
13	1	Sorting techniques	**Tutorial-2**	BB, Q&A,Learning by Doing	18.12.2015
14	1	Quick sort: Example Algorithm explanation	Regular	BB	21.12.2015
15	1	Analysis of Quick Sort: Best, Worst and Average cases	Regular	BB,OHP/LCD, Learning By Doing	22.12.2015
16	1	Divide &Conquer Approach to Matrix Multiplication	Regular	BB,OHP/LCD, Learning By Doing	23.12.2015
17	1	Strassen's Matrix multiplication	Regular	BB,OHP/LCD, Learning By Doing	24.12.2015
18	1	Solving Recurrence Relations	**Tutorial-3**	BB, Q&A,Learning by Doing	
19	1	Probabilistic Analysis	Regular	BB,OHP/LCD, Learning By Doing	28.01.2015
20	1	Amortized Complexity	Regular	BB,OHP/LCD, Learning By Doing	29.12.2015
UNIT-II					
21	1	Binary Tree Traversals –inorder, preorder and Post order Recursive algorithms	Regular	BB	04.01.2016

22	1	Non Recursive Tree Traversal algorithms	Regular	BB	
23	1	Disjoint Sets: operations and representation	Regular	BB,OHP/LCD	05.01.2016
24	1	Simple Union and Find algorithms	Regular	BB,OHP/LCD	06.01.2016
25	1	Weighted Union and Collapsing Find algorithms	Regular	BB,OHP/LCD	07.01.2016
26	1	Binary Tree traversals, operations on Disjoint sets	**Tutorial-4**	BB, Q&A,Learning by Doing	
27	1	Graphs – BFS and DFS	Regular	BB,OHP/LCD	08.01.2016
28	1	Bi connected components - Articulation Point	Regular	BB,OHP/LCD, Learning By Doing	11.01.2016
29	1	Determining articulation points thru dfn()	**Tutorial-5**	BB, Q&A,Learning by Doing	12.01.2016
30	1	Algorithm for Articulation point, Biconnected components	Regular	BB,OHP/LCD, Learning By Doing	
31	1	AND/OR Graphs	Regular	BB,OHP/LCD	
32	1	Game Trees	Regular	BB,OHP/LCD	
UNIT-III					
33	1	Greedy Method: General method	Regular	BB	18.01.2016
34	1	Applications: Knapsack Problem	Regular	BB	19.01.2016
35	1	minimum cost spanning trees	Regular	BB	20.01.2016
36	1	Prim's Algorithm	Regular	BB	21.01.2016
37	1	Kruskal's Algorithm	Regular	BB	22.01.2016
38	1	Knapsack problem and Minimum cost spanning trees	**Tutorial-6**	BB, Q&A,Learning by Doing	
39	1	single source shortest path problem	Regular	BB,OHP/LCD ,Learning By Doing	25.01.2016
40	1	Job Sequencing with deadlines	Regular	BB	27.01.2016
41	1	Dynamic Programming: General Method Principle of Optimality	Regular	BB	28.01.2016
42	1	Applications- Multistage Graph	Regular	BB,OHP/LCD, Learning By Doing	29.01.2016
43	1	Optimal Binary Search Trees(OBST) - Dynamic Programming approach	Regular	BB,OHP/LCD, Learning By Doing	30.01.2016
44	1	OBST- Example, Algorithm	Regular	BB,OHP/LCD, Learning By Doing	
45	1	Multistage graph – Backward approach, OBST	**Tutorial-7**	BB, Q&A,Learning by Doing	
46	1	0/1 knapsack problem	Regular	BB	22.02.2016
47	1	All pairs shortest path problem	Regular	BB,OHP/LCD, Learning By Doing	23.02.2016
48	1	traveling sales person problem	Regular	BB	24.02.2016
49	1	Reliability design	Regular	BB	25.02.2016
50	1	0/1 knapsack problem, Reliability Design All pairs shortest path problem	**Tutorial-8**	BB, Q&A,Learning by Doing	
UNIT -IV					
51	1	Backtracking : General method	Regular	BB	26.02.2016
52	1	State space tree – Terminology, fixed vs variable tuple size formulation	Regular	BB	
53	1	Recursive & Iterative Backtracking algorithm	Regular	BB	

54	1	State space tree Backtracking approach to problems	**Tutorial-9**	BB, Q&A,Learning by Doing	
55	1	Applications- n-queens problem	Regular	BB	29.02.2016
56	1	Graph coloring	Regular	BB	01.03.2016
57	1	Hamiltonian cycles			
58	1	sum of subsets problem	Regular	BB	03.03.2016
59	1	Determining Bounding function Generating state space tree	**Tutorial-10**	BB	04.03.2016
60	1	Branch & Bound: General Method	Regular	BB	07.03.2016
61	1	LC Search: 15- Puzzle Problem	Regular	BB	08.03.2016
62	1	Job Sequencing with Deadlines: Branch & Bound	Regular	BB	10.03.2016
63		Job Sequencing with Deadlines: FIFOBB, LIFOBB, LCBB	**Tutorial-11**	BB	
64	1	Applications - 0/1 knapsack problem FIFOBB	Regular	BB,OHP/LCD	11.03.2016
65	1	0/1 Knapsack Problem- LCBB	Regular	BB	14.03.2016
66	1	Traveling Sales Person(TSP)- Reduced Cost Matrix	Regular	BB	15.03.2016
67	1	TSP- Optimal Solution thru LCBB	Regular	BB	16.03.2016
68	1	TSP Another example	**Tutorial-12**	BB	21.03.2016
UNIT-V: NP-Hard and NP-Complete problems					
69	1	Tractable vs Intractable problems	Regular	BB,OHP/LCD	23.03.2016
70	1	Non Deterministic algorithms	Regular	BB,OHP/LCD	25.03.2016
71	1	More examples on Non Deterministic algorithms	Regular	BB,OHP/LCD	28.03.2016
72	1	P and NP Problems- NP Hard & NP-Complete classes	Regular	BB,OHP/LCD	04.04.2016
73	1	NP-Hard problems- Reducibility	Regular	BB,OHP/LCD	06.04.2016
74	1	Cook's Theorem	Regular	BB,OHP/LCD	09.04.2016

II-CSE-B

S.No.	No. of Periods	Topics to be covered	Regular/ Additional	Teaching Aids Used LCD/OHP/BB	Expected Date(s)
UNIT-I					
1	1	Introduction to subject course objectives and outcomes	Regular	BB	07.12.2015
2	1	Algorithm, Pseudo code for expressing algorithms	Regular	BB	08.12.2015
3	1	Writing algorithm for sample problems	Regular	BB	
4	1	Performance analysis: Space complexity.	Regular	BB	08.12.2015
5	1	Time complexity: using variable count	Regular	BB	09.12.2015
6	1	Time complexity: using frequency count table	Regular	BB	
7	1	Asymptotic Notation: Big oh, Omega, Theta & Little oh notation	Regular	BB,OHP/LCD,Q&A	10.12.2015

8	1	Examples on Asymptotic Notations and Theorems	**Tutorial-1**	BB,Q&A Learning by Doing,	11.12.2015
9	1	Divide and Conquer: General Method	Regular	BB,OHP/LCD,Q&A,E-Tutorials	14.12.2015
10	1	Recurrence Relations : Substitution method	Regular	BB,OHP/LCD, Internet, Q&A	15.12.2015
11	1	Applications: Binary Search	Regular	BB,OHP/LCD, Learning By Doing, E-Tutorials, Internet, Q&A	16.12.2015
12	1	Merge sort: Example and Algorithm	Regular	BB,OHP/LCD	17.12.2015
13	1	Sorting techniques	**Tutorial-2**	BB, Q&A,Learning by Doing	18.12.2015
14	1	Quick sort: Example Algorithm explanation	Regular	BB	21.12.2015
15	1	Analysis of Quick Sort: Best, Worst and Average cases	Regular	BB,OHP/LCD, Learning By Doing	22.12.2015
16	1	Divide &Conquer Approach to Matrix Multiplication	Regular	BB,OHP/LCD, Learning By Doing	23.12.2015
17	1	Strassen's Matrix multiplication	Regular	BB,OHP/LCD, Learning By Doing	24.12.2015
18	1	Solving Recurrence Relations	**Tutorial-3**	BB, Q&A,Learning by Doing	
19	1	Probabilistic Analysis	Regular	BB,OHP/LCD, Learning By Doing	28.01.2015
20	1	Amortized Complexity	Regular	BB,OHP/LCD, Learning By Doing	29.12.2015
UNIT-II					
21	1	Binary Tree Traversals –inorder, preorder and Post order Recursive algorithms	Regular	BB	04.01.2016
22	1	Non Recursive Tree Traversal algorithms	Regular	BB	
23	1	Disjoint Sets: operations and representation	Regular	BB,OHP/LCD	05.01.2016
24	1	Simple Union and Find algorithms	Regular	BB,OHP/LCD	06.01.2016
25	1	Weighted Union and Collapsing Find algorithms	Regular	BB,OHP/LCD	07.01.2016
26	1	Binary Tree traversals, operations on Disjoint sets	**Tutorial-4**	BB, Q&A,Learning by Doing	
27	1	Graphs – BFS and DFS	Regular	BB,OHP/LCD	08.01.2016
28	1	Bi connected components - Articulation Point	Regular	BB,OHP/LCD, Learning By Doing	11.01.2016
29	1	Determining articulation points thru dfn()	**Tutorial-5**	BB, Q&A,Learning by Doing	12.01.2016
30	1	Algorithm for Articulation point, Biconnected components	Regular	BB,OHP/LCD, Learning By Doing	
31	1	AND/OR Graphs	Regular	BB,OHP/LCD	
32	1	Game Trees	Regular	BB,OHP/LCD	
UNIT-III					
33	1	Greedy Method: General method	Regular	BB	18.01.2016
34	1	Applications: Knapsack Problem	Regular	BB	19.01.2016
35	1	minimum cost spanning trees	Regular	BB	20.01.2016
36	1	Prim's Algorithm	Regular	BB	21.01.2016
37	1	Kruskal's Algorithm	Regular	BB	22.01.2016
38	1	Knapsack problem and Minimum cost spanning trees	**Tutorial-6**	BB, Q&A,Learning by Doing	

39	1	single source shortest path problem	Regular	BB,OHP/LCD ,Learning By Doing	25.01.2016
40	1	Job Sequencing with deadlines	Regular	BB	27.01.2016
41	1	Dynamic Programming: General Method Principle of Optimality	Regular	BB	28.01.2016
42	1	Applications- Multistage Graph	Regular	BB,OHP/LCD, Learning By Doing	29.01.2016
43	1	Optimal Binary Search Trees(OBST) - Dynamic Programming approach	Regular	BB,OHP/LCD, Learning By Doing	30.01.2016
44	1	OBST- Example, Algorithm	Regular	BB,OHP/LCD, Learning By Doing	
45	1	Multistage graph – Backward approach, OBST	**Tutorial-7**	BB, Q&A,Learning by Doing	
46	1	0/1 knapsack problem	Regular	BB	22.02.2016
47	1	All pairs shortest path problem	Regular	BB,OHP/LCD, Learning By Doing	23.02.2016
48	1	traveling sales person problem	Regular	BB	24.02.2016
49	1	Reliability design	Regular	BB	25.02.2016
50	1	0/1 knapsack problem, Reliability Design All pairs shortest path problem	**Tutorial-8**	BB, Q&A,Learning by Doing	
UNIT -IV					
51	1	Backtracking : General method	Regular	BB	26.02.2016
52	1	State space tree – Terminology, fixed vs variable tuple size formulation	Regular	BB	
53	1	Recursive & Iterative Backtracking algorithm	Regular	BB	
54	1	State space tree Backtracking approach to problems	**Tutorial-9**	BB, Q&A,Learning by Doing	
55	1	Applications- n-queens problem	Regular	BB	29.02.2016
56	1	Graph coloring	Regular	BB	01.03.2016
57	1	Hamiltonian cycles			
58	1	sum of subsets problem	Regular	BB	03.03.2016
59	1	Determining Bounding function Generating state space tree	**Tutorial-10**	BB	04.03.2016
60	1	Branch & Bound: General Method	Regular	BB	07.03.2016
61	1	LC Search: 15- Puzzle Problem	Regular	BB	08.03.2016
62	1	Job Sequencing with Deadlines: Branch & Bound	Regular	BB	10.03.2016
63		Job Sequencing with Deadlines: FIFOBB, LIFOBB, LCBB	**Tutorial-11**	BB	
64	1	Applications - 0/1 knapsack problem FIFOBB	Regular	BB,OHP/LCD	11.03.2016
65	1	0/1 Knapsack Problem- LCBB	Regular	BB	14.03.2016
66	1	Traveling Sales Person(TSP)- Reduced Cost Matrix	Regular	BB	15.03.2016
67	1	TSP- Optimal Solution thru LCBB	Regular	BB	16.03.2016
68	1	TSP Another example	**Tutorial-12**	BB	21.03.2016
UNIT-V: NP-Hard and NP-Complete problems					
69	1	Tractable vs Intractable problems	Regular	BB,OHP/LCD	23.03.2016
70	1	Non Deterministic algorithms	Regular	BB,OHP/LCD	25.03.2016
71	1	More examples on Non Deterministic algorithms	Regular	BB,OHP/LCD	28.03.2016
72	1	P and NP Problems- NP Hard & NP-Complete classes	Regular	BB,OHP/LCD	04.04.2016

73	1	NP-Hard problems- Reducibility	Regular	BB,OHP/LCD	06.04.2016
74	1	Cook's Theorem	Regular	BB,OHP/LCD	09.04.2016

II-CSE-C

S.No.	No. of Periods	Topics to be covered	Regular/ Additional	Teaching Aids Used LCD/OHP/BB	Expected Date(s)
UNIT-I					
1	1	Introduction to subject course objectives and outcomes	Regular	BB	07.12.2015
2	1	Algorithm, Pseudo code for expressing algorithms	Regular	BB	08.12.2015
3	1	Writing algorithm for sample problems	Regular	BB	
4	1	Performance analysis: Space complexity.	Regular	BB	08.12.2015
5	1	Time complexity: using variable count	Regular	BB	09.12.2015
6	1	Time complexity: using frequency count table	Regular	BB	
7	1	Asymptotic Notation: Big oh, Omega, Theta & Little oh notation	Regular	BB,OHP/LCD,Q&A	10.12.2015
8	1	Examples on Asymptotic Notations and Theorems	**Tutorial-1**	BB,Q&A Learning by Doing,	11.12.2015
9	1	Divide and Conquer: General Method	Regular	BB,OHP/LCD,Q&A,E-Tutorials	14.12.2015
10	1	Recurrence Relations : Substitution method	Regular	BB,OHP/LCD, Internet, Q&A	15.12.2015
11	1	Applications: Binary Search	Regular	BB,OHP/LCD, Learning By Doing, E-Tutorials, Internet, Q&A	16.12.2015
12	1	Merge sort: Example and Algorithm	Regular	BB,OHP/LCD	17.12.2015
13	1	Sorting techniques	**Tutorial-2**	BB, Q&A,Learning by Doing	18.12.2015
14	1	Quick sort: Example Algorithm explanation	Regular	BB	21.12.2015
15	1	Analysis of Quick Sort: Best, Worst and Average cases	Regular	BB,OHP/LCD, Learning By Doing	22.12.2015
16	1	Divide &Conquer Approach to Matrix Multiplication	Regular	BB,OHP/LCD, Learning By Doing	23.12.2015
17	1	Strassen's Matrix multiplication	Regular	BB,OHP/LCD, Learning By Doing	24.12.2015
18	1	Solving Recurrence Relations	**Tutorial-3**	BB, Q&A,Learning by Doing	
19	1	Probabilistic Analysis	Regular	BB,OHP/LCD, Learning By Doing	28.01.2015
20	1	Amortized Complexity	Regular	BB,OHP/LCD, Learning By Doing	29.12.2015
UNIT-II					
21	1	Binary Tree Traversals –inorder, preorder and Post order Recursive algorithms	Regular	BB	04.01.2016
22	1	Non Recursive Tree Traversal algorithms	Regular	BB	
23	1	Disjoint Sets: operations and representation	Regular	BB,OHP/LCD	05.01.2016

24	1	Simple Union and Find algorithms	Regular	BB,OHP/LCD	06.01.2016
25	1	Weighted Union and Collapsing Find algorithms	Regular	BB,OHP/LCD	07.01.2016
26	1	Binary Tree traversals, operations on Disjoint sets	**Tutorial-4**	BB, Q&A,Learning by Doing	
27	1	Graphs – BFS and DFS	Regular	BB,OHP/LCD	08.01.2016
28	1	Bi connected components - Articulation Point	Regular	BB,OHP/LCD, Learning By Doing	11.01.2016
29	1	Determining articulation points thru dfn()	**Tutorial-5**	BB, Q&A,Learning by Doing	12.01.2016
30	1	Algorithm for Articulation point, Biconnected components	Regular	BB,OHP/LCD, Learning By Doing	
31	1	AND/OR Graphs	Regular	BB,OHP/LCD	
32	1	Game Trees	Regular	BB,OHP/LCD	
UNIT-III					
33	1	Greedy Method: General method	Regular	BB	18.01.2016
34	1	Applications: Knapsack Problem	Regular	BB	19.01.2016
35	1	minimum cost spanning trees	Regular	BB	20.01.2016
36	1	Prim's Algorithm	Regular	BB	21.01.2016
37	1	Kruskal's Algorithm	Regular	BB	22.01.2016
38	1	Knapsack problem and Minimum cost spanning trees	**Tutorial-6**	BB, Q&A,Learning by Doing	
39	1	single source shortest path problem	Regular	BB,OHP/LCD ,Learning By Doing	25.01.2016
40	1	Job Sequencing with deadlines	Regular	BB	27.01.2016
41	1	Dynamic Programming: General Method Principle of Optimality	Regular	BB	28.01.2016
42	1	Applications- Multistage Graph	Regular	BB,OHP/LCD, Learning By Doing	29.01.2016
43	1	Optimal Binary Search Trees(OBST) - Dynamic Programming approach	Regular	BB,OHP/LCD, Learning By Doing	30.01.2016
44	1	OBST- Example, Algorithm	Regular	BB,OHP/LCD, Learning By Doing	
45	1	Multistage graph – Backward approach, OBST	**Tutorial-7**	BB, Q&A,Learning by Doing	
46	1	0/1 knapsack problem	Regular	BB	22.02.2016
47	1	All pairs shortest path problem	Regular	BB,OHP/LCD, Learning By Doing	23.02.2016
48	1	traveling sales person problem	Regular	BB	24.02.2016
49	1	Reliability design	Regular	BB	25.02.2016
50	1	0/1 knapsack problem, Reliability Design All pairs shortest path problem	**Tutorial-8**	BB, Q&A,Learning by Doing	
UNIT -IV					
51	1	Backtracking : General method	Regular	BB	26.02.2016
52	1	State space tree – Terminology, fixed vs variable tuple size formulation	Regular	BB	
53	1	Recursive & Iterative Backtracking algorithm	Regular	BB	
54	1	State space tree Backtracking approach to problems	**Tutorial-9**	BB, Q&A,Learning by Doing	
55	1	Applications- n-queens problem	Regular	BB	29.02.2016

56	1	Graph coloring	Regular	BB	01.03.2016
57	1	Hamiltonian cycles			
58	1	sum of subsets problem	Regular	BB	03.03.2016
59	1	Determining Bounding function Generating state space tree	**Tutorial-10**	BB	04.03.2016
60	1	Branch & Bound: General Method	Regular	BB	07.03.2016
61	1	LC Search: 15- Puzzle Problem	Regular	BB	08.03.2016
62	1	Job Sequencing with Deadlines: Branch & Bound	Regular	BB	10.03.2016
63		Job Sequencing with Deadlines: FIFOBB, LIFOBB, LCBB	**Tutorial-11**	BB	
64	1	Applications - 0/1 knapsack problem FIFOBB	Regular	BB,OHP/LCD	11.03.2016
65	1	0/1 Knapsack Problem- LCBB	Regular	BB	14.03.2016
66	1	Traveling Sales Person(TSP)- Reduced Cost Matrix	Regular	BB	15.03.2016
67	1	TSP- Optimal Solution thru LCBB	Regular	BB	16.03.2016
68	1	TSP Another example	**Tutorial-12**	BB	21.03.2016
UNIT-V: NP-Hard and NP-Complete problems					
69	1	Tractable vs Intractable problems	Regular	BB,OHP/LCD	23.03.2016
70	1	Non Deterministic algorithms	Regular	BB,OHP/LCD	25.03.2016
71	1	More examples on Non Deterministic algorithms	Regular	BB,OHP/LCD	28.03.2016
72	1	P and NP Problems- NP Hard & NP-Complete classes	Regular	BB,OHP/LCD	04.04.2016
73	1	NP-Hard problems- Reducibility	Regular	BB,OHP/LCD	06.04.2016
74	1	Cook's Theorem	Regular	BB,OHP/LCD	09.04.2016

II-CSE-D

S.No.	No. of Periods	Topics to be covered	Regular/ Additional	Teaching Aids Used LCD/OHP/BB	Expected Date(s)
UNIT-I					
1	1	Introduction to subject course objectives and outcomes	Regular	BB	07.12.2015
2	1	Algorithm, Pseudo code for expressing algorithms	Regular	BB	08.12.2015
3	1	Writing algorithm for sample problems	Regular	BB	
4	1	Performance analysis: Space complexity.	Regular	BB	08.12.2015
5	1	Time complexity: using variable count	Regular	BB	09.12.2015
6	1	Time complexity: using frequency count table	Regular	BB	
7	1	Asymptotic Notation: Big oh, Omega, Theta & Little oh notation	Regular	BB,OHP/LCD,Q&A	10.12.2015
8	1	Examples on Asymptotic Notations and Theorems	**Tutorial-1**	BB,Q&A Learning by Doing,	11.12.2015
9	1	Divide and Conquer: General Method	Regular	BB,OHP/LCD,Q&A,E-Tutorials	14.12.2015
10	1	Recurrence Relations : Substitution method	Regular	BB,OHP/LCD, Internet, Q&A	15.12.2015

11	1	Applications: Binary Search	Regular	BB,OHP/LCD, Learning By Doing, E-Tutorials, Internet, Q&A	16.12.2015
12	1	Merge sort: Example and Algorithm	Regular	BB,OHP/LCD	17.12.2015
13	1	Sorting techniques	**Tutorial-2**	BB, Q&A,Learning by Doing	18.12.2015
14	1	Quick sort: Example Algorithm explanation	Regular	BB	21.12.2015
15	1	Analysis of Quick Sort: Best, Worst and Average cases	Regular	BB,OHP/LCD, Learning By Doing	22.12.2015
16	1	Divide &Conquer Approach to Matrix Multiplication	Regular	BB,OHP/LCD, Learning By Doing	23.12.2015
17	1	Strassen's Matrix multiplication	Regular	BB,OHP/LCD, Learning By Doing	24.12.2015
18	1	Solving Recurrence Relations	**Tutorial-3**	BB, Q&A,Learning by Doing	
19	1	Probabilistic Analysis	Regular	BB,OHP/LCD, Learning By Doing	28.01.2015
20	1	Amortized Complexity	Regular	BB,OHP/LCD, Learning By Doing	29.12.2015
UNIT-II					
21	1	Binary Tree Traversals –inorder, preorder and Post order Recursive algorithms	Regular	BB	04.01.2016
22	1	Non Recursive Tree Traversal algorithms	Regular	BB	
23	1	Disjoint Sets: operations and representation	Regular	BB,OHP/LCD	05.01.2016
24	1	Simple Union and Find algorithms	Regular	BB,OHP/LCD	06.01.2016
25	1	Weighted Union and Collapsing Find algorithms	Regular	BB,OHP/LCD	07.01.2016
26	1	Binary Tree traversals, operations on Disjoint sets	**Tutorial-4**	BB, Q&A,Learning by Doing	
27	1	Graphs – BFS and DFS	Regular	BB,OHP/LCD	08.01.2016
28	1	Bi connected components - Articulation Point	Regular	BB,OHP/LCD, Learning By Doing	11.01.2016
29	1	Determining articulation points thru dfn()	**Tutorial-5**	BB, Q&A,Learning by Doing	12.01.2016
30	1	Algorithm for Articulation point, Biconnected components	Regular	BB,OHP/LCD, Learning By Doing	
31	1	AND/OR Graphs	Regular	BB,OHP/LCD	
32	1	Game Trees	Regular	BB,OHP/LCD	
UNIT-III					
33	1	Greedy Method: General method	Regular	BB	18.01.2016
34	1	Applications: Knapsack Problem	Regular	BB	19.01.2016
35	1	minimum cost spanning trees	Regular	BB	20.01.2016
36	1	Prim's Algorithm	Regular	BB	21.01.2016
37	1	Kruskal's Algorithm	Regular	BB	22.01.2016
38	1	Knapsack problem and Minimum cost spanning trees	**Tutorial-6**	BB, Q&A,Learning by Doing	
39	1	single source shortest path problem	Regular	BB,OHP/LCD ,Learning By Doing	25.01.2016
40	1	Job Sequencing with deadlines	Regular	BB	27.01.2016
41	1	Dynamic Programming: General Method Principle of Optimality	Regular	BB	28.01.2016
42	1	Applications- Multistage Graph	Regular	BB,OHP/LCD, Learning By Doing	29.01.2016

43	1	Optimal Binary Search Trees(OBST) - Dynamic Programming approach	Regular	BB,OHP/LCD, Learning By Doing	30.01.2016
44	1	OBST- Example, Algorithm	Regular	BB,OHP/LCD, Learning By Doing	
45	1	Multistage graph – Backward approach, OBST	**Tutorial-7**	BB, Q&A,Learning by Doing	
46	1	0/1 knapsack problem	Regular	BB	22.02.2016
47	1	All pairs shortest path problem	Regular	BB,OHP/LCD, Learning By Doing	23.02.2016
48	1	traveling sales person problem	Regular	BB	24.02.2016
49	1	Reliability design	Regular	BB	25.02.2016
50	1	0/1 knapsack problem, Reliability Design All pairs shortest path problem	**Tutorial-8**	BB, Q&A,Learning by Doing	
UNIT -IV					
51	1	Backtracking : General method	Regular	BB	26.02.2016
52	1	State space tree – Terminology, fixed vs variable tuple size formulation	Regular	BB	
53	1	Recursive & Iterative Backtracking algorithm	Regular	BB	
54	1	State space tree Backtracking approach to problems	**Tutorial-9**	BB, Q&A,Learning by Doing	
55	1	Applications- n-queens problem	Regular	BB	29.02.2016
56	1	Graph coloring	Regular	BB	01.03.2016
57	1	Hamiltonian cycles			
58	1	sum of subsets problem	Regular	BB	03.03.2016
59	1	Determining Bounding function Generating state space tree	**Tutorial-10**	BB	04.03.2016
60	1	Branch & Bound: General Method	Regular	BB	07.03.2016
61	1	LC Search: 15- Puzzle Problem	Regular	BB	08.03.2016
62	1	Job Sequencing with Deadlines: Branch & Bound	Regular	BB	10.03.2016
63		Job Sequencing with Deadlines: FIFOBB, LIFOBB, LCBB	**Tutorial-11**	BB	
64	1	Applications - 0/1 knapsack problem FIFOBB	Regular	BB,OHP/LCD	11.03.2016
65	1	0/1 Knapsack Problem- LCBB	Regular	BB	14.03.2016
66	1	Traveling Sales Person(TSP)- Reduced Cost Matrix	Regular	BB	15.03.2016
67	1	TSP- Optimal Solution thru LCBB	Regular	BB	16.03.2016
68	1	TSP Another example	**Tutorial-12**	BB	21.03.2016
UNIT-V: NP-Hard and NP-Complete problems					
69	1	Tractable vs Intractable problems	Regular	BB,OHP/LCD	23.03.2016
70	1	Non Deterministic algorithms	Regular	BB,OHP/LCD	25.03.2016
71	1	More examples on Non Deterministic algorithms	Regular	BB,OHP/LCD	28.03.2016
72	1	P and NP Problems- NP Hard & NP-Complete classes	Regular	BB,OHP/LCD	04.04.2016
73	1	NP-Hard problems- Reducibility	Regular	BB,OHP/LCD	06.04.2016
74	1	Cook's Theorem	Regular	BB,OHP/LCD	09.04.2016

14. Detailed notes

UNIT-I

DEFINITIONS:

TIME COMPLEXITY: The time complexity of an algorithm is the amount of computer time it needs to run to completion.

SPACE COMPLEXITY: The space complexity of an algorithm is the amount of memory it needs to run to completion.

ASYMPTOTIC NOTATION:

BIG Oh Notation: The function f(n) =O (g (n)) if and only if there exists positive constants c and n0 such that f(n)<= c*g (n) for all n, n>= n0

Omega Notation: The function f(n) =Ω(g (n)) if and only if there exists positive constants c and n0 such that f(n)>= c*g (n) for all n, n>= n0

Theta Notation: The function f (n) =θ(g (n)) if and only if there exists positive constants c1.c2 and n0 such that c1g (n) <= f (n) <= c2g (n) for all n, n > = n0

Control Abstraction: By control Abstraction we mean a procedure whose flow of control is clear but whose primary operations are specified by other procedures whose precise meanings are left undefined.

Divide and conquer: The divide and conquer paradigm breaks the problem into several sub problems that are similar to the original problem but are smaller in size, solve the sub problems recursively, and then combine these solutions to create a solution to the original problem.

Merge Sort:

Its worst-case time complexity is O (n log n)

Given a sequence of n elements the general idea is to split them into two sets this division continues recursively until each sub part has only single element. Each set is individually sorted and the resulting sequences are merged to produce a single sorted sequence of n elements. Procedure MERGESORT describes this process using recursion and a procedure MERGE that merges together two sorted lists.

Quick Sort:

Best Case is O (n log n)
Average Case is O (n log n).
Worst Case is O (n^2).

In quick sort the division into two sub arrays is made so that the sorted sub arrays do not need to be merged later. This is accomplished by rearranging the elements in a [1:n] such that a [I]<=a [j] for all I between 1 and m and all j between m+1 and n for some m, 1<=m<=n. Thus, the elements a [1:m] and a [m+1:n] can be independently sorted.

This rearrangement is called partitioning.

Strassen's matrix multiplication: Let A and B be two n*n matrices the product C=A*B is also an n*n matrix .To compute C [I, J] we require n multiplications. As the matrix C has n^2 elements the time for resulting matrix multiplication algorithm is given by

$\theta(n^3)$. Since matrix multiplications are more expensive then matrix additions, we can attempt to reformulate the equations so as to have fewer multiplications and possibly more additions.

Divide and conquer:

The divide and conquer paradigm breaks the problem into several sub problems that are similar to the original problem but are smaller in size, solve the sub problems recursively, and then combine these solutions to create a solution to the original problem.

Merge Sort:

its worst-case time complexity is O (n log n)

Given a sequence of n elements the general idea is to split them into two sets this division continues recursively until each sub part has only single element. Each set is individually sorted and the resulting sequences are merged to produce a single sorted sequence of n elements. Procedure MERGESORT describes this process using recursion and a procedure MERGE that merges together two sorted lists.

Quick Sort:

Best Case is O (n log n)
Average Case is O (n log n).
Worst Case is O (n^2).

In quick sort the division into two sub arrays is made so that the sorted sub arrays do not need to be merged later. This is accomplished by rearranging the elements in a [1:n] such that a [I]<=a [j] for all I between 1 and m and all j between m+1 and n for some m, 1<=m<=n. Thus, the elements a [1:m] and a [m+1:n] can be independently sorted.
This rearrangement is called partitioning.

Strassen's matrix multiplication:

Let A and B be two n*n matrices the product C=A*B is also an n*n matrix .To compute C [I, J] we require n multiplications. As the matrix C has n^2 elements the time for resulting matrix multiplication algorithm is given by
$\theta(n^3)$. Since matrix multiplications are more expensive then matrix additions, we can attempt to reformulate the equations so as to have fewer multiplications and possibly more additions.
For 2*2 matrix generally we require 8 multiplications and 4 additions. But by using Strassen's multiplication we require only 7 multiplications and 18 additions.

Knapsack problem: We are given n objects and a knapsack. Object j has a weight w_j and the knapsack has the capacity m .If a fraction n_j $0<=n_j<=1$ of object j is placed into knapsack then a profit of $p_j x_j$ is earned .The objective is to obtain a filling of the knapsack that maximizes the total weight of all chosen objects to be at most m.

Minimum spanning tree: Let G = (V, E) ^{be} an undirected connected graph. A sub graph T = (V, E) of G is a spanning tree of G if and only if t is a tree.

Minimum cost spanning tree can be obtained using prim's algorithm and kruskhal's algorithm.

For 2*2 matrix generally we require 8 multiplications and 4 additions. But by using Strassen's multiplication we require only 7 multiplications and 18 additions.

Knapsack problem: We are given n objects and a knapsack. Object j has a weight w_j and the knapsack has the capacity m .If a fraction n_j $0<=n_j<=1$ of object j is placed into knapsack then a profit of $p_j x_j$ is earned .The objective is to obtain a filling of the knapsack that maximizes the total weight of all chosen objects to be at most m.

Minimum spanning tree: Let G = (V, E) be an undirected connected graph. A sub graph T = (V, E) of G is a spanning tree of G if and only if t is a tree.
Minimum cost spanning tree can be obtained using prim's algorithm and kruskhal's algorithm.

UNIT-II
SEARCHING AND TRAVERSAL TECHNIQUES

Tree Traversals: There are three types of tree traversals. They are

INORDER TRAVERSAL
PREORDER TRAVERSAL
POSTORDER TRAVERSAL

T is a binary tree. Each node of t has three fields:
lchild, data, rchild.

```
Algorithm Inorder (t)
{
        if ( t ≠ 0) then
        {
                Inorder (t→ lchild)
                Visit (t)
                Inorder (t→rchild)
        }
}
Algorithm Preorder (t)
{
        if ( t ≠ 0) then
        {
                Visit (t)
                Preorder (t→ lchild)
                Preorder (t→rchild)
        }

}
Algorithm Postorder (t)
{
        if ( t ≠ 0) then
        {
                Postorder (t→ lchild)
                Visit (t)
                Postorder (t→rchild)
        }

}
```

GRAPH TRAVERSALS:

There are two types of graph traversals

i) Breadth First Search.

ii) Depth First Search.

A breadth first search of G is carried out beginning at vertex v. For any node i, visited [i] =1 if i has already been visited. The graph G and array visited [] are global; visited [] is initialized to zero.

```
Algorithm BFS (v)
{
u:= v;
Visited [v]: =1;
        repeat
            {
                for all vertices w adjacent from u do
                 {
                        if (visited [w] =0) then
                        {
                                Add w to q;
                                Visited[w]:=1;
                        }
                }
                if q is empty then return;
                Delete u from q;
        } until (false)
}

Algorithm DFS(v)
{
 Visited [v]: =1;
 for each vertex w adjacent from v  do
   {
       if  ( visited[w]=0) then DFS (w)

   }

}
```

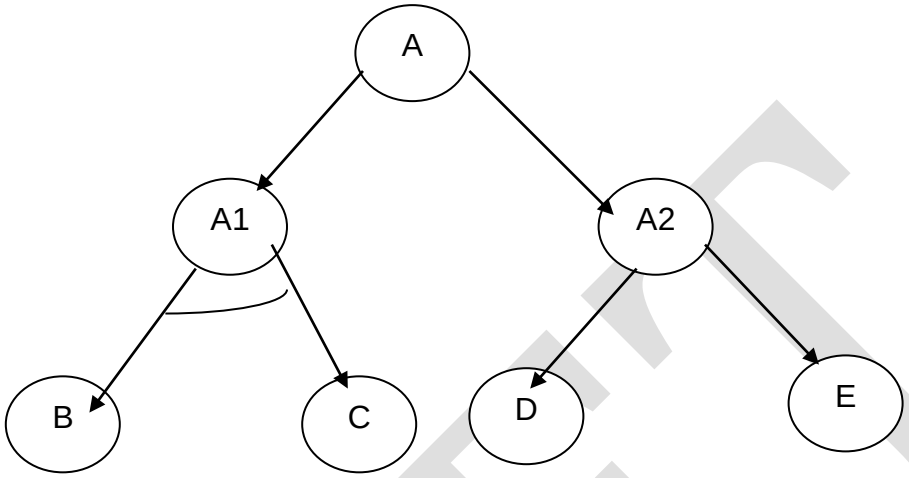
AND/OR GRAPHS:

The breakdown of a complex problem into several sub problems can be represented by a decimal graph like structure in which nodes represent problems and descendents of nodes represent the sub problem associated with them.

All nodes can be made to be such that their solution requires either all descendents to be solved or any one descendent to be solved. Nodes of first type are called AND nodes and those of the latter type are called OR nodes.

Nodes with no descendents are called terminal. Terminal nodes represent primitive problems and are marked either solvable or not solvable. Solvable terminal nodes are represented by rectangles. An AND/OR graph need not always be a tree.

EXAMPLE: AND/OR GRAPH



Nodes A1 and A2 of above figure are OR nodes.
Nodes B and C are AND nodes.
If we have to complete sub problem A1 we should complete its two sub problems B and C. There fore A1 is AND node.

GAME TREE:

A game tree is defined as a pictorial representation of all valid, finite length sequences of board configuration (C1, C2…Cm) of a game. Each valid sequence with Cm being a terminal configuration is called an Instance of board configuration. A board configuration is set be valid if

i) Ci is starting configuration of the game.
ii) Ci, 0 <I<m are non-terminal configurations.
iii) Ci + 1 is obtained from Ci by legal move made by player 1 If I is odd and by player 2 If I is even.

It is assume that there are finite numbers of legal moves. A legal move consists of removing an agreed number of toothpicks from the file in the "NIM" game. For instance. The degree any node in a game tree is at most equal to the number of distinct legal move. The depth of game tree is a length of longest instance of the game. Game trees are useful in determining the next move a player should make.

ARTICULATION POINT:

A vertex V in a connected graph is an articulation point if and only if the deletion of vertex V together with all edges incident to V disconnects the graph into two or more components.

Unit-3

DYNAMIC PROGRAMMING

DEFINITIONS:

MULTISTAGE GRAPHS

A multistage graph G=(V, E) is a directed graph in which the vertices are partitioned into k>2 disjoint sets V_i ,1<=i<=k. In addition if (U, V) is an edge in E then $U \in V_i$ and $V \in V_{i+1}$ for some i, 1<=i<=k. The sets V_1 and V_k are such that $|V_1|=|V_k|=1$. Let s and t respectively be the vertices in V_1 and V_k. The vertex S is the source and T is the sink. Let C (i, j) be the cost of edge (i, j). The cost of the path from s to t is the sum of the cost of the edges on the path.

The multistage graph problem is to find a minimum cost path from S to T. Each set V_i defines a stage in the graph. Because of the constraints on E, every path from S to T starts in stage1 goes to stage2 then to stage3 and so on and eventually terminates at stage k.

Multistage graph problem can be solved using either forward approach or backward approach.

ALL PAIRS SHORTEST PATH:

The all pair shortest path problem is to determine a matrix A such that A (i, j) is the length of a shortest path from i to j. If this problem is solved by n single source shortest path problems, the matrix A may be obtained in O (n^3) time.
An alternate solution to the problem using dynamic programming of the O (n^3) is as given.

The shortest path originates at vertex 'i' and goes through some intermediate vertices and terminates at vertex 'j'. This path contains no cycle for if there is a cycle then this may be deleted without increasing the path length. If 'k' and from 'k' to 'j' must be shortest paths from 'i' to 'k' and from 'k' to 'j'. Otherwise, the 'i' to 'j' going through no vertex of index greater than 'k'. We get

$$A(i, j) = \min \{\min \{ A^{k-1}(i, k) + A^{k-1}(k, j)\}, C(i, j)\}$$

OPTIMAL BINARY SEARCH TREE:

In obtaining a cost function for binary search tree, it is useful to add a fictious node in place of every empty sub tree in the search tree. Such nodes are called External nodes and are drawn square. All other nodes are internal nodes. If a binary search tree represents n identifiers then there will be exactly n internal nodes and n+1 external nodes. Every internal node represents a point where a successful search may terminate. Every external node represents a point where an unsuccessful search may terminate. If a successful search terminates at an internal node at level L. Hence the expected cost contribution from the internal node for ai is p (i) * level (a_i). If the failure node for E_i is at level l then only l-1 iterations are made hence cost contribution of this node is Q(i)*(level(E_i)-1).

The preceding discussion leads to the following formula for expected cost of binary search tree.

$$\sum_{1<=i<=n} p(i)*level(a_i) + \sum_{0<=i<=n} q(i)*(level(E_i)-1)$$

We define optimal binary search tree for the identifier set (a1, a2.... an) to be a binary search tree for which above equation is minimum.

0/1 KNAPSACK:

A set to the knapsack problem may be obtained by making a sequence of decisions on the variables X_1, X_2, X_3 .....X_n. A decision on variable X_i involves deciding which of the values 0 or 1 is to be assigned to it. Let us assume that decision on the X_i are made in the order X_n, X_{n-1}..... X_1.

Following a decision on X_n we may be in one of the two possible states. The capacity remaining in knapsack is M and no profit has accrued or the capacity remaining is M- W_n and a profit of P_n has accrued. It is clear that remaining decisions xn-1,... x1 must be optimal with respect the problem state resulting from the decision on xn. Hence the principle of optimality holds.

RELIABILITY DESIGN:

In this section we look at an example of how to use Dynamic programming to solve a problem with a multiplicative optimization function. The problem is to design a system, which is composed of several devices connected in series. Let ri be the reliability device Di.

Then the reliability of the entire system is Π ri. Even if the individual devices are very reliable, the reliability of system may not be very good.

For e.g. If n=10 and ri=. 99,1<=I<=10 then Π ri=. 904. Hence it is desirable to duplicate devices. Multiple copies of the same device type are connected in parallel through the use of switching circuits. The switching circuits determine which devices in any group are functioning properly. They then make use of one such device at each stage.

If stage i contains mi copies of device Di then the probability that all mi have a malfunction is $(1-ri)^{mi}$. Thus if ri=. 99 and mi=2 the stage reliability becomes .9999

TRAVELLING SALESMAN PROBLEM:

Let G=(V, E) be a directed graph with edge cost Cij . Cij is defined such that Cij>0 for a1l i and j and Cij=∞ if <I, j>∉ E. Let (V)= n and assume n>1. A tour of G is a directed cycle that includes every vertex in V. the cost of the tour is the sum of the cost of the edges on that tour. The traveling sales man problem is to find a tour of minimum cost. Let g (i, s) be the length of shortest path starts at vertex 1 going through all vertices in s and terminating at vertex 1.

G (1,v- {1}) is the length of an optimal salesman tour. From the principle of optimality it follows that g(I,s)=min{Cij + g(j,s-{j})}

BRANCH AND BOUND:

The term branch and bound refers to all state space search methods in which all children of the E-node are generated before any other live node can become the E-node. We have already seen two graph search strategies, BFS and D-search, in which the exploration of a new node cannot begin until the node currently

being explored is fully explored. Both of these generalize to branch and bound strategies. In branch and bound terminology, a BFS like state space search will be called FIFO search, as the list of live nodes is a FIFO list. A D-search like state space search will be called LIFO search as the list of live nodes is a LIFO. Bounding functions are used to help avoid the generation of sub trees that do not contain an answer node.

LC search:

The search for an answer node can often be speeded by using an intelligent ranking function c^ (.) for live nodes. The next E-node is selected on the basis of this ranking function. The ideal way to assign ranks would be on the basis of the additional computational effort needed to reach an answer node from the live node.

BOUNDING: A branch-and- bound method searches a states space tree using any search mechanism in which all the children of the E-node are generated before another node becomes the E-node. We assume that each answer node x has cost c (x) associated with it and that a minimum cost answer node is to be found. Three common strategies are FIFO, LIFO and LC. A cost function c^ (.) such that c^ (x)<=c (x) is used to provide lower bounds on solutions obtainable from any node x. If upper is an upper bound on the cost of a minimum-cost solution, Then all live nodes x with c^ (x)> upper may be killed as all answer nodes reachable from x have cost c (x)>= c^ (x)> upper. The starting value for upper can be obtained from some heuristic or can be set to infinity. Clearly, so long as the initial value for upper space is no less than the cost of minimum-cost answer node the above rules to kill live nodes will not result in the killing of a live node that can be reach a minimum cost answer node. Each time an answer is found, the value of upper can be updated.

FIFO BRANCH AND BOUND:

In implementing FIFO branch - and -bound algorithm, it is not economical to kill live nodes with c^ (x) > upper each time upper is updated. This is so because live nodes are in the queue in the order in which they were generated. Hence, nodes with c^(x) > upper are distributed in some random way in the queue. Instead, live nodes with c^(x) > upper can be killed when they are about to become E-nodes.

ALGEBRIC PROBLEMS:

In this topic we shift our attention away from the problems we have dealt with previously to concentrate on methods for dealing with numbers and polynomials. Though computer have the ability already built –in to manipulate integer and reals, they are not directly equipped to manipulated symbolic mathematical expressions such as polynomials. One must determined a way to represent then and then write procedures that perform the desire operations. A system that allows for the manipulations of mathematical expressions is called a mathematical symbol manipulation system. The systems have been fruitfully used to solve variety scientific problems for many years. The technique study here often let to efficient ways to implement the operations offered by these systems.

Assume we have an input I that is a member of set S1 and a function f (I) that describes what must be computed. Usually the output f (I) is also a member of S1.Though a method may exist for computing f (I) using operations on elements in S1, this method may be inefficient .The algebraic transformation technique suggest that we alter the input into to another form to produce a member of set S2.The set S2 contains exactly the same elements as S1 except it assumes a different representation for them. Once the answer in S2 is computed, an inverse transformation perform is to yield the result in set S1.

Backtracking:

Backtracking algorithms determine problem solutions by systematic searching the solution space for the given problem instance. Using a tree organization from the solution space facilitates this search. For a given solution space many tree organization may be possible.

The basic idea is to build up the solution vector one component at a time and to use modified criterion functions Pi (x1…xi) sometimes called bounding functions) to test whether the vector being formed has any chance of success. The major advantage of this method is this: If it is realized that the partial vectors can be ignored entirely.

Many of the problems we solve using backtracking require that all the solutions satisfy a complex set of constraints. For any problem these constraints can be divided into two categories: explicit and implicit.

Explicit Constraints:

Explicit constraints are rules that restrict each Xi to take on values from given set.
Common examples of explicit constraints are
$x_i \geq 0$ Si ={all non negative real numbers}
x_i =0 or 1 or Si ={0,1}
$i_i \leq x_i \leq u_i$ or Si = {a: $i_i \leq a \leq u_i$ }
The explicit constraints depend on the particular instance I of the problem being solved. All tuples that satisfy the explicit constraints define a possible solution space for i.

Implicit Constraints: The implicit constraints are rules that determine which of the tuples in the solution space of I satisfy the criterion function. Thus implicit Constraints describe the way in which the x_i must relate to each other.

8-Queen problem:
Consider a chessboard and try to find final all ways to place 8 attacking queens.

We can let (X1…Xn) represent a solution in which Xi is the column of the i^{th} row where the i^{th} queen is placed. The Xi will all be distinct since no two queens' can be placed in the same column. Suppose two queens are placed at position (i, j) & (k, l) then they are on the same diagonal any if

$$|J - l| = |i-k|$$

All solutions to the 8- queens problem can therefore be represented as 8-tuples
(X_1,……., X8) . The explicit constraints using this formulation s are S_i={1,2,3,4,5,6,7,8}, 1≤ i≤ n . Therefore the solution space consists of 8^8 8- tuples.
The implicit constraints for this problem are that no two Xi can be the same. The realization reduces the size of the solution space from 8^8 tuples to 8! Tuples.
Expressed as an 8 –tuple, the solution in figure is (4,6,8,2,7,1,3,5).

1	2	3	4	5	6	7	8
			Q				
					Q		
							Q

	Q						
						Q	
Q							
		Q					
				Q			

SUM OF SUBSETS:

Suppose we are given n distinct positive numbers usually called weights and we desire to find all combinations of these numbers whose sums are m.
This is called sum of subsets problem.

GRAPH COLORING:

Let G be a graph and m be a +ve integer. We want to find Whether the nodes of G can be colored in such a way that no two adjacent nodes have the same color yet only m colors are used. If d is the degree of the graph than it can be colored with d+1 colors. The m-colorability optimization problem asks for the smallest integer m for which the graph G can be colored. This integer is referred to as the chromatic number of the graph

UNIT-V

NP-Hard and NP-Complete problems: In this section the examine the operations on polynomials. As we search for efficient algorithms, we see examples of another design strategy called algebraic simplification. It refers to the process of re expressing computational formulas so that the required the number of operations to compute these formulas is minimized. One issue we ignore where is the numerical stability of the resulting algorithms.

A uni-variate polynomial is generally written ass

$A(x)=a_nx^n + a_{n-1}x^{n-1} +\ldots\ldots+ a_1x + a_0$

Where x is indeterminate and a_i may be integers, floating point numbers are more generally elements of commutative ring or a field. If $a_n \neq 0$ then is called the degree of A.

When considering the representation of a polynomial by its coefficients there are at least two alternatives. The first calls for storing the degree followed by degree +1 coefficients.

$(n, a_n , a_{n-1} ,\ldots\ldots a_1 , a_0)$

This is termed the dense representation because it explicitly stores all coefficients whether or they are zero.

The second representation calls for storing only each non-zero coefficient and its corresponding exponent; for example, if all the a_i are nonzero, then the polynomial is stored as

$(n, a_n ,n-1, a_{n-1} ,\ldots\ldots 1,a_1 ,0, a_0)$

This is termed as sparse representation because the storage depends directly on the number of nonzero terms and not on degree.

For a polynomial of degree n , all of whose coefficients are nonzero, this second representation requires roughly twice the storage of the first. However, that is the worst case. For high-degree polynomials with few nonzero terms, the second representation is many times better than the first.

HORNER's rule: Polynomial can be expressed as

$A(x) = (\ldots((a_n x + a_{n-1}) x + a_{n-2}) x + \ldots.. a_1)x + a_0)$

It requires n multiplications, n additions, and n+1 assignments.

NP-Hard and NP-Complete Problems

Aims:

- To describe SAT, a very important problem in complexity theory;
- To describe two more classes of problems: the **NP**-Hard and **NP**-Complete problems.

33.2. NP-Hard and NP-Complete Problems

33.2.1. NP-Hard Problems

- We say that a decision problem P_i is ***NP**-hard* if *every* problem in **NP** is polynomial-time reducible to P_i.
- In symbols,

  P_i is **NP**-hard if, for every $P_j \in \mathbf{NP}$, $P_j \xrightarrow{\text{poly}} P_i$.
- Note that this doesn't require P_i to be in **NP**.
- Highly informally, it means that P_i is 'as hard as' all the problems in **NP**.
  - If P_i can be solved in polynomial-time, then so can all problems in **NP**.
  - Equivalently, if any problem in **NP** is ever proved intractable, then P_i must also be intractable.

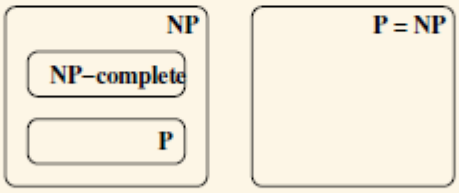
33.2.2. NP-Complete Problems

- We say that a decision problem P_i is ***NP**-complete* if
  - it is **NP**-hard and
  - it is also in the class **NP** itself.
- In symbols, P_i is **NP**-complete if P_i is **NP**-hard and $P_i \in \mathbf{NP}$
- Highly informally, it means that P_i is one of the hardest problems in **NP**.

- So the **NP**-complete problems form a set of problems that may or may not be intractable but, whether intractable or not, are all, in some sense, of equivalent complexity.
- If anyone ever shows that an **NP**-complete problem is tractable, then
  - every **NP**-complete problem is also tractable
  - indeed, every problem in **NP** is tractable

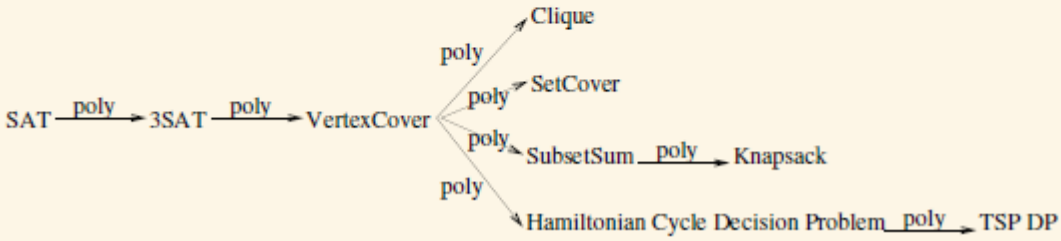
  and so $\mathbf{P} = \mathbf{NP}$.
- If anyone ever shows that an **NP**-complete problem is intractable, then
  - every **NP**-complete problem is also intractable

  and, of course, $\mathbf{P} \neq \mathbf{NP}$.
- So there are two possibilities:

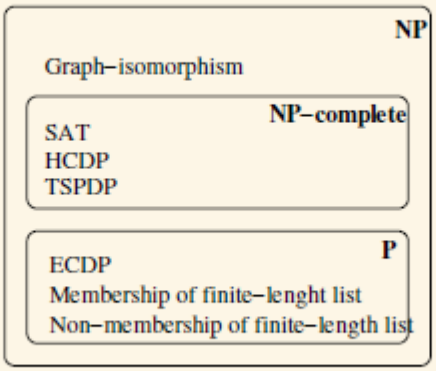


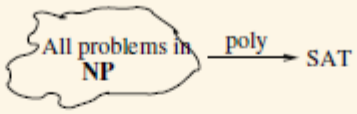
  We don't know which of these is the case.
- But this gives Computer Scientists a clear line of attack. It makes sense to focus efforts on the **NP**-complete problems: they all stand or fall together.
- So these sound like very significant problems in our theory. But how would you show that a decision problem is **NP**-complete?

- Starting with SAT and using Method 2, numerous problems have been shown to be **NP**-complete.
- Without going into the details of the problems or the reductions themselves, here is a picture that shows a few of the polynomial-time reductions that have been found.



- Here's a picture showing some actual decision problems.





- The proof is beyond the scope of this course and the result goes by the name of the Cook-Levin Theorem

33.2.4. How to Show Other Problems are NP-Complete

- We have one problem that is proven to be **NP**-complete, where the proof is done generically and 'from scratch'. Showing that other problems are **NP**-complete is easier.
- How to show decision problem P_i is **NP**-complete (Method 2)
  - First, confirm it is a decision problem.
  - Then show P_i is in **NP**.
  - Then show that P_i is **NP**-hard by taking just one problem P_j that is already known to be **NP**-complete and showing that $P_j \xrightarrow{\text{poly}} P_i$
- Why does the latter show P_i to be **NP**-hard?

  If P_j is **NP**-complete, then we know that P_j is **NP**-hard (by the definition of **NP**-complete), i.e. every problem in **NP** is polynomially-reducible to P_j.
  But, if every problem in **NP** is polynomially-reducible to P_j and P_j is polynomially-reducible to P_i then, by transitivity, every problem in **NP** is polynomially-reducible to P_i.

15. Additional Topics

No additional topics

16. University question papers

Code No: RR-210504

Set No: 1

II B.Tech I-Semester Regular Examinations, November 2004

DESIGN AND ANALYSIS OF ALGORITHMS

(Common to Computer Science and Engineering, Information Technology and Computer Science and Systems Engineering)

Time: 3 hours **Max. Marks: 80**

Answer any FIVE questions

All questions carry equal marks

---

1. Define time complexity. Describe different notations used to represent these complexities. Illustrate.

2. a) Analyze the average case time complexity of Quick sort.
   b) If k is a non-negative constant, then show that the solution to the given recurrence equation, for n a power of 2 is $T(n) = 3kn^{\log 3} - 2kn$.
   $T(n) = k, \quad n=1$
   $3T(n/2)+kn, \quad n>1$

3. a) Write Prim's algorithm under the assumption that the graphs are represented by adjacency lists.
   b) Analyze precisely the computing time and space requirements of this new version of Prim's algorithm using adjacency lists.

4. a) What are Dictionaries? Explain.
   b) What is a balanced tree? Differentiate between 2-3 trees and AVL trees.

5. a) What do you mean by forward and backward approach of problem solving in Dynamic programming?
   b) What are the differences between the Greedy and Dynamic programming methods of problem solving?

6. a) Present an algorithm for depth first search traversal. Explain with an example.
   b) Write a detailed note on breadth first traversal.

7. a) What is graph coloring? Present an algorithm which finds all m-colorings of a graph.
   b) Draw the state space tree for m-closing graph using a suitable graph.

8. Present a program schema for a FIFO Branch & Bound search for a Least-Cost answer node.

###

Set No: 2

Code No: RR-210504

II B.Tech I-Semester Regular Examinations, November 2004
DESIGN AND ANALYSIS OF ALGORITHMS
(Common to Computer Science and Engineering, Information Technology and Computer Science and Systems Engineering)

Time: 3 hours **Max. Marks: 80**

Answer any FIVE questions
All questions carry equal marks
---

1. Define space complexity. Explain the same with an example.

2. a) Compare Merge sort & Quick sort for the given data sets.
10, 30, 15, 45, 25, 30, 35, 20, 30, 40, 50
b) Compare their time complexities.

3. Explain the Kruskal's algorithm with an example and analyze its time complexity.

4. a) Write a procedure DIVIDE(b,T) to implement SPLIT(b,s) instruction which partitions a 2-3 tree T about a leaf 'b' so that all leaves to the left of 'b' and 'b' itself is in one 2-3 tree and leaves to the right of 'b' are in a second 2-3 tree.
b) Prove that the above procedure takes time O(height(T)) and the order of the tree is preserved.

5. Discuss the dynamic programming solutions for the problems of reliability design and traveling sales person problem.

6. a) Show that the in-order and post order sequences of a binary tree uniquely define the binary tree.
b) Write a detailed note on depth-first traversal.

7. Explain in detail how the technique of backtracking can be applied to solve the 8 queen's problem. Present the required algorithms.

8. Write a program schema for a LIFO Branch & Bound for a Least-Cost answer node.

###

Set No: 3

Code No: RR-210504

II B.Tech I-Semester Regular Examinations, November 2004
DESIGN AND ANALYSIS OF ALGORITHMS
(Common to Computer Science and Engineering, Information Technology and Computer Science and Systems Engineering)

Time: 3 hours **Max. Marks: 80**

Answer any FIVE questions
All questions carry equal marks

---

1. Write the non-recursive algorithm for finding the Fibonacci sequence and derive its time complexity.

2. a) Design a Divide and Conquer algorithm for computing the number of levels in a binary tree.
   b) Compute the efficiency of the above algorithm.

3. Formulate an algorithm for a spanning tree problem in terms of a sequence of set operations in which take G as the undirected graph ; S as the undirected tree; V as the number of vertices; E as the number of edges T as a set used to collect the edges of the final minimum spanning tree; C as the cost function for the graph G given by $\{ e \Sigma \varepsilon E_1 C(e)$ for the sub-graph $G1 = (V_1,E_1)$ of G. Use set VS for the vertex set of the forest to write the minimum cost spanning tree algorithm.

4. a) Write a pseudo code for constructing 2-3 trees for a given list of n integers.
   b) Explain the above algorithm for a list of 10 integers.

5. a) Using Divide and Conquer approach coupled with the set generation approach, show how to obtain an $O(2^{n/2})$ algorithm for 0/1 Knapsack problem.
   b) Develop an algorithm that uses this approach to solve the 0/1 Knapsack problem.
   c) Compare the run time and storage requirements of this approach.

6. a) Write a non-recursive algorithm for the pre-order traversal of a binary tree T, using stacks.
   b) What are the time and space requirements of your algorithm?

7. Define the following terms: state space, explicit constraints, implicit constraints, problem state, solution states, answer states, live nod, E-node, dead node, bounding functions.

8. Draw the portion of a state space tree generated by FIFOBB, LCBB and LIFOBB for the job sequencing with deadlines instance n=5, $(p_1,p_2,\ldots,p_5)=(6,3,4,8,5)$, $(t_1,t_2,\ldots,t_5)=(2,1,2,1,1)$ and $(d_1,d_2,\ldots,d_5)=(3,1,4,2,4)$. What is the penalty corresponding to an optimal solution? Use a variable tuple size formulation and ĉ (.) and u (.).

###

Code No: RR-210504 **Set No: 4**

II B.Tech I-Semester Regular Examinations, November 2004
DESIGN AND ANALYSIS OF ALGORITHMS
(Common to Computer Science and Engineering, Information Technology and Computer Science and Systems Engineering)

Time: 3 hours **Max. Marks: 80**

Answer any FIVE questions
All questions carry equal marks

---

1. A complex valued matrix X is represented by a pair of matrices (A,B) where A and B contain real values. Write an algorithm that computes the product of two complex valued matrices (A,B) and (C,D) where (A,B) * (C,D) = (A+iB) * (C+iD) = (AC-BD) + i (AD+BC). Determine the number of additions and multiplications if the matrices are all n×n.

2. a) Devise a version of Merge sort algorithm which performs sorting in-place.
   b) Devise a ternary search algorithm which first tests the element at position n/3 for equality with some value x and then possibly checks the element at 2n/3 and either discovers x or reduces the set size to one third of the original. Compare this with the binary search.

3. Explain the Job sequencing with dead line algorithm and also find the solution for the instance n=7, $(P_1,P_2,\ldots,P_7)=(3,5,20,18,1,6,30)$ and $(D_1,D_2,\ldots,D_7)=(1,3,4,3,2,1,2)$.

4. a) Write an algorithm for checking whether an array H [1,2,….,n] is a heap or not.
   b) Determine the time efficiency of the above algorithm.

5. Using the algorithm OBST, compute W(i,j), R(i,j) and C(i,j), 0<=i<j<=4 for the identifier set (a_1,a_2,a_3,a_4)=(end, goto, print, stop) with p(1)=1/20, p(2)=1/5, p(3)=1/10, p(4)=1/20; q(0)=1/5, q(1)=1/10, q(2)=1/5, q(3)=1/20 and q(4)=1/20. Using the R(i,j)'s construct the OBST.

6. Write a non-recursive algorithm for the inorder traversal of binary tree T. Each node has 4 fields: LCHILD, DATA, PARENT, RCHILD. Your algorithm should take no more than O(1) additional space and O(n) time for an n-node tree. Show that this is the time taken by your algorithm.

7. Compare and contrast
   a) Brute force approach Vs Backtracking
   b) Fixed Vs variable tuple size formulation.

8. Consider the LCBB traveling sales person algorithm described using the dynamic state space tree formulation. Let A and B be nodes. Let B be the child of A. If the edge (A,B) represents the inclusion of edge <i,j> in the tour, then in the reduced matrix for B all entries in row i and column j are set to ∞. In addition, one more entry is set to ∞. Obtain an efficient way to determine this entry.

###

Code No: RR-210504

II B.Tech I-Semester Regular Examinations, November 2004

DESIGN AND ANALYSIS OF ALGORITHMS

(Common to Computer Science and Engineering, Information Technology and Computer Science and Systems Engineering)

Time: 3 hours **Max. Marks: 80**

Answer any FIVE questions

All questions carry equal marks

---

1. Define time complexity. Describe different notations used to represent these complexities. Illustrate.

2. a) Analyze the average case time complexity of Quick sort.
   b) If k is a non-negative constant, then show that the solution to the given recurrence equation, for n a power of 2 is $T(n) = 3kn^{\log 3} - 2kn$.
   $T(n) = k, \quad n=1$
   $3T(n/2)+kn, \quad n>1$

3. a) Write Prim's algorithm under the assumption that the graphs are represented by adjacency lists.
   b) Analyze precisely the computing time and space requirements of this new version of Prim's algorithm using adjacency lists.

4. a) What are Dictionaries? Explain.
   b) What is a balanced tree? Differentiate between 2-3 trees and AVL trees.

5. a) What do you mean by forward and backward approach of problem solving in Dynamic programming?
   b) What are the differences between the Greedy and Dynamic programming methods of problem solving?

6. a) Present an algorithm for depth first search traversal. Explain with an example.
   b) Write a detailed note on breadth first traversal.

7. a) What is graph coloring? Present an algorithm which finds all m-colorings of a graph.
   b) Draw the state space tree for m-closing graph using a suitable graph.

8. Present a program schema for a FIFO Branch & Bound search for a Least-Cost answer node.

###

Set No: 2

Code No: RR-210504

II B.Tech I-Semester Regular Examinations, November 2004
DESIGN AND ANALYSIS OF ALGORITHMS
(Common to Computer Science and Engineering, Information Technology and Computer Science and Systems Engineering)

Time: 3 hours **Max. Marks: 80**

Answer any FIVE questions
All questions carry equal marks
---

1. Define space complexity. Explain the same with an example.

2. a) Compare Merge sort & Quick sort for the given data sets.
10, 30, 15, 45, 25, 30, 35, 20, 30, 40, 50
c) Compare their time complexities.

3. Explain the Kruskal's algorithm with an example and analyze its time complexity.

4. a) Write a procedure DIVIDE(b,T) to implement SPLIT(b,s) instruction which partitions a 2-3 tree T about a leaf 'b' so that all leaves to the left of 'b' and 'b' itself is in one 2-3 tree and leaves to the right of 'b' are in a second 2-3 tree.
b) Prove that the above procedure takes time O(height(T)) and the order of the tree is preserved.

5. Discuss the dynamic programming solutions for the problems of reliability design and traveling sales person problem.

6. a) Show that the inorder and post order sequences of a binary tree uniquely define the binary tree.
c) Write a detailed note on depth-first traversal.

7. Explain in detail how the technique of backtracking can be applied to solve the 8 queen's problem. Present the required algorithms.

8. Write a program schema for a LIFO Branch & Bound for a Least-Cost answer node.

###

Code No: RR-210504

Set No: 2

II B.Tech I-Semester Regular Examinations, November 2004
DESIGN AND ANALYSIS OF ALGORITHMS
(Common to Computer Science and Engineering, Information Technology and Computer Science and Systems Engineering)

Time: 3 hours **Max. Marks: 80**

Answer any FIVE questions
All questions carry equal marks

---

1. Write the non-recursive algorithm for finding the Fibonacci sequence and derive its time complexity.

2. a) Design a Divide and Conquer algorithm for computing the number of levels in a binary tree.
   b) Compute the efficiency of the above algorithm.

3. Formulate an algorithm for a spanning tree problem in terms of a sequence of set operations in which take G as the undirected graph ; S as the undirected tree; V as the number of vertices; E as the number of edges T as a set used to collect the edges of the final minimum spanning tree; C as the cost function for the graph G given by $\{ e \Sigma \varepsilon E_1 C(e)$ for the sub-graph G1 = (V_1,E_1) of G. Use set VS for the vertex set of the forest to write the minimum cost spanning tree algorithm.

4. a) Write a pseudo code for constructing 2-3 trees for a given list of n integers.
   b) Explain the above algorithm for a list of 10 integers.

5. a) Using Divide and Conquer approach coupled with the set generation approach, show how to obtain an $O(2^{n/2})$ algorithm for 0/1 Knapsack problem.
   b) Develop an algorithm that uses this approach to solve the 0/1 Knapsack problem.
   c) Compare the run time and storage requirements of this approach.

6. a) Write a non-recursive algorithm for the pre-order traversal of a binary tree T, using stacks.
   b) What are the time and space requirements of your algorithm?

7. Define the following terms: state space, explicit constraints, implicit constraints, problem state, solution states, answer states, live nod, E-node, dead node, bounding functions.

8. Draw the portion of a state space tree generated by FIFOBB, LCBB and LIFOBB for the job sequencing with deadlines instance n=5, $(p_1,p_2,\ldots,p_5)=(6,3,4,8,5)$, $(t_1,t_2,\ldots,t_5)=(2,1,2,1,1)$ and $(d_1,d_2,\ldots,d_5)=(3,1,4,2,4)$. What is the penalty corresponding to an optimal solution? Use a variable tuple size formulation and ĉ (.) and u (.).

###

Code No: RR-210504

Set No: 4

II B.Tech I-Semester Regular Examinations, November 2004
DESIGN AND ANALYSIS OF ALGORITHMS
(Common to Computer Science and Engineering, Information Technology and Computer Science and Systems Engineering)

Time: 3 hours **Max. Marks: 80**

Answer any FIVE questions
All questions carry equal marks

---

1. A complex valued matrix X is represented by a pair of matrices (A,B) where A and B contain real values. Write an algorithm that computes the product of two complex valued matrices (A,B) and (C,D) where (A,B) * (C,D) = (A+iB) * (C+iD) = (AC-BD) + i (AD+BC). Determine the number of additions and multiplications if the matrices are all n×n.

2. a) Devise a version of Merge sort algorithm which performs sorting in-place.
   b) Devise a ternary search algorithm which first tests the element at position n/3 for equality with some value x and then possibly checks the element at 2n/3 and either discovers x or reduces the set size to one third of the original. Compare this with the binary search.

3. Explain the Job sequencing with dead line algorithm and also find the solution for the instance n=7, $(P_1,P_2,\ldots,P_7)=(3,5,20,18,1,6,30)$ and $(D_1,D_2,\ldots,D_7)=(1,3,4,3,2,1,2)$.

4. a) Write an algorithm for checking whether an array H [1,2,….,n] is a heap or not.
   b) Determine the time efficiency of the above algorithm.

5. Using the algorithm OBST, compute W(i,j), R(i,j) and C(i,j), 0<=i<j<=4 for the identifier set (a_1,a_2,a_3,a_4)=(end, goto, print, stop) with p(1)=1/20, p(2)=1/5, p(3)=1/10, p(4)=1/20; q(0)=1/5, q(1)=1/10, q(2)=1/5, q(3)=1/20 and q(4)=1/20. Using the R(i,j)'s construct the OBST.

6. Write a non-recursive algorithm for the inorder traversal of binary tree T. Each node has 4 fields: LCHILD, DATA, PARENT, RCHILD. Your algorithm should take no more than O(1) additional space and O(n) time for an n-node tree. Show that this is the time taken by your algorithm.

7. Compare and contrast
   c) Brute force approach Vs Backtracking
   d) Fixed Vs variable tuple size formulation.

8. Consider the LCBB traveling sales person algorithm described using the dynamic state space tree formulation. Let A and B be nodes. Let B be the child of A. If the edge (A,B) represents the inclusion of edge <i,j> in the tour, then in the reduced matrix for B all entries in row i and column j are set to ∞. In addition, one more entry is set to ∞. Obtain an efficient way to determine this entry.

17. Question Bank

Unit-1

1. a) Define an algorithm. Describe the Characteristics of the algorithm.
   b) What do you mean by the input size of a problem? Explain its significance.
2. Describe the Performance analysis in detail.
3. a) Define time Complexity. Describe different notations used to represent these complexities.
   b) Write the non-recursive algorithm for finding the Fibonacci sequence and derive its time complexity.
4. a) Write an algorithm to find a largest of given 'n' numbers. Derive its time complexity using big 'oh' notation.
   b) Explain about amortization.
5. a) Show that $f(n)+g(n) = o(n^2)$ where $f(n)=3n^2-n+4$ and $g(n)= n \log n+5$
   b) Show that $f(n) =4n^2 – 64n + 288 =\Omega(n^2)$.
6. a) Write and explain the control abstraction for divide and conquer.
   b) Explain Binary Search and give its algorithm.
7. a) Sort the following elements using Merge Sort: 10, 30, 15, 45, 25, 30, 35, 20, 30, 40, 50
   b) Write the Algorithm for Merge sort and derive its time complexity.
8. a) Write the algorithm for quick sort.
   b) Derive its time complexity for Best, Worst and Average cases.
9. a) Trace the quick sort algorithm to sort the list C, O, L, L, E, G, E in alphabetical order.
   b) Give an instance, where the quick sort algorithm has worst case time complexity.
10. a) If K is a non negative constant, then prove that the recurrence

$$T(n)= \begin{cases} K & n = 1 \\ 3T(n/2) + kn & n > 1 \end{cases}$$

has the following solution (for n a power of 2); $T(n) = 3kn \log 3 – 2kh$.

b) Solve the following recurrence relation using substitution method.

$$T(n) = \begin{cases} 1 & n \leq 4 \\ 2T(\sqrt{n}) + \log n & n > 4 \end{cases}$$

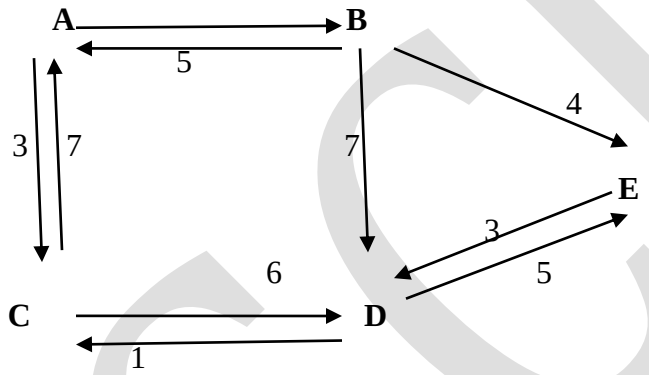
Unit-II

1. a) List various tree traversal techniques. Explain with an example.
   b) Write non-recursive algorithm for in-order binary tree traversal.
2. Develop algorithms for UNION and FIND using weighting rule and collapsing rule respectively.
3. a) Write and explain BFS and DFS Algorithms for Graphs.
   b) Write BFS and DFS order for the following Graph.
4. a) What are biconnected components? Explain.
   b) Write an algorithm to determine the articulation points and the biconnected components of a graph G.

5. a) Explain in detail AND/OR Graphs.
   b) (Game Trees)

Unit-III

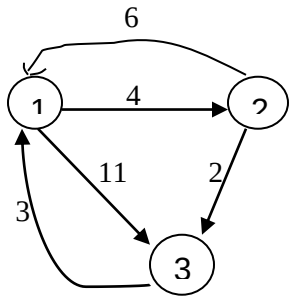
1. a) Explain the terms feasible solution, optimal solution and objective function.
   b) Find an optimal solution to the knapsack instance n=7 m=15.
   (P1, P2…..P7)= (10,5,15,7,6,18,3), and (W1, W2,…..W7) = (2,3,5,7,1,4,1).
2. a) Explain the prims algorithm with the appropriate example.
   b) Write the prims algorithm to find the minimum cost spanning tree.
3. a) What is a spanning tree? What are the applications of it.
   b) Find the shortest path for the given graph by using single source shortest path.



4. a) Explain in the control abstraction for greedy method.
   b) Write kruskal's algorithm to find the maximum spanning tree.
5. a) Explain the algorithm for Job sequencing with deadlines. Applying the same find the solution for the instance n=4, (P1, P2…..P4) = (100.10, 15.27), and (d1, d2….d4) = (2, 1, 2, 1).
6. a) What is Principle of optimality? Explain its significance.
   b) Multistage Graphs
7. a) Define an Optimal Binary Search Tree (OBST)
   b) Construct an OBST when n=4, such that a1< a2< a3< a4. With (q0, q1, q2, q4) = (1/4, 3/16, 1/16, 1/16, 1/16) and (p1, p2, p3, p4) = (1/4, 1/8, 1/16, 1/16).
8. a) Define merge and purge rules.
   b) Consider the Knapsack instance n=6, m=165, (p1, p2…..p6) = (w1, w2…w6) = (100, 50, 20, 10, 7, 3). Generate the S^i sets containing the pair (pi, wi) and thus find the optimal solution.

9. a) Explain the OBST algorithm.
   b) Write an Algorithm for 0/1 Knapsack problem using Dynamic programming.

10. a) Write an algorithm of all pairs shortest path problem.
    b) Find the shortest path between all pairs of nodes in the following Graph.



11. a) What is Travelling Sales Person problem and what are its applications?
    b) Find the shortest tour of a TSP for following instance using Dynamic programming.

	A	B	C	D
A	0	10	15	20
B	5	0	9	10
	6	13	0	12
D	8	8	9	0

12. a) Explain in detail the Reliability Design problem.
    b) Design three stage system with device types D1, D2, D3. The costs are Rs. 30, Rs. 15 and Rs. 20 respectivly. The Costs of the system into be not more than Rs. 105. The reliability of each device type is 0.9, 0.8 and 0.5 respectivly.

UNIT-IV

1. a) Compare and contrast Brute Force approach and Back tracking.
   b) Write the control abstraction of Backtracking.

2. a) Define the following terms. Explicit constraints, Implicit constraints, state space tree, problem state, Answer state, live node, E-node, dead node. Back tracking and Branch and Bound.

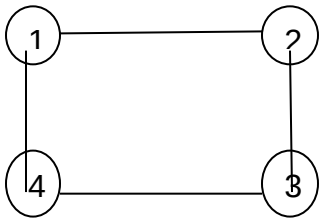
3. a) Compare and contrast fixed vs variable tuple size formulation.
   b) Draw the tree organization of the 4-queens solution space.Number the nodes using
      i) BFS ii) DFS iii) D-Search

4. a) Explain in detail how Backtracking can be applied to solve the 8-queens problem.
   b) Write an algorithm for n-queens problem.

5. a) Explain in detail the Sum of Subsets problem with an example.
   b) Write recursive Backtracking algorithm for Sum of Subsets problem.

6. Consider the following instance of Sum of Subsets problem:
   W={15, 7, 20, 5, 18, 10, 12} and m=35. Find all possible Sub Sets of w that sum to m. Draw the portion of the state space tree that is generated.

7. a) Draw the state space tree for the following graph when m=3(m-Coluring graph problem).



b) Devise a Backtracking algorithm for m-Coloring graph problem.

8. a) Define Hamiltonian cycle. Give an example.
b) Write a Backtracking algorithm fro finding all Hamiltonian cycles in a given graph.

9. a) Define the term Branch and Bound and Explain it with an example.
b) Write an algorithm for LC-Search.

10. Draw the portion of the State Space Tree generated by FIFO BB, LCBB and LIFOBB for the job sequensing with deadlines instance, n=4, (p1, d1, t1) = (5, 1, 1), (p2, d2, t2) = (10, 3, 2), (p3, d3, t3) = (6, 2, 1) and (p4, d4, t4) = (3, 1, 1). What is the penalty corresponding to an optional solution.

11. Draw the portion the State Space Tree generated by LCBB and FIFOBB for the following Knapsack instance.
N=4, (p1, p2, p3, p4) = (10, 10, 12, 18), (w1, w2, w3, w4) = (2, 4, 6, 9) and m=15.

12. Consider the Travelling Salesperson instance defined by the cost matrix.

$$\begin{pmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{pmatrix}$$

a) Obtain the reduced cost matrix.
b) Using a State Space Tree formulation, obtain the portion of the State Space tree that will be generated by LCBB. Label each node by its $\hat{C}$ value, write out the reduced matrices corresponding to each of these nodes.

13. Solve the above problem using reduced matrix method and the Dynamic State Space Tree approach.

UNIT – V

1. Write about the theory of NP-Completeness.
2. Write shortnotes on Deterministic and Non-Deterministic algorithm with an example.
3. Explain Non Deterministic search and sorting algorithms.
4. Write about Decision and Optimization problems with examples.
5. a) Explain the Clique problem and write the algorithm for the same.
   b) Write a Non-Deterministic Knapsack algorithm.
6. Explain the Satisifiability problem and write the algorithm for the same.
7. a) With a neat diagram, explain the relevance of NP-hard and NP-complete problems.
   b) What is meant by Halting problem. Explain with an example.
8. State and explain Cook's Theorem.

18. Assignment sheets

Assignment-1

1. Express the following function in Big oh ,omega and theta notations
a) $4*2^n$ b) $10\log n+6$ c) $10n^2+5n$
2. Write the non recursive algorithm for finding the Fibonacci sequence and derive the complexity
3. Given an array of n elements write an algorithm to remove all duplicates from the array. Give its time and space complexity
4. Sort the following elements using quick sort
   11, 8, 10, 6, 19, 12, 7, 14

Assignment-2

1. Present an algorithm height union that uses the height rule for union operation instead of the weighted rule. The rule is defined as below
   If the height of a tree i is less than that of the tree j then make j the parent of i otherwise make i the parent of j.
2. Prove that the maximum time and space taken by algorithm BFS on any graph G with 'n' vertices and 'e' edges is $T(n,e)=\Theta(n+e)$, $S(n,e)= \Theta(n)$, if G is represented by the adjacency list and $T(n,e)= \Theta(n^2)$ and $S(n,e)= \Theta(n)$.
3. Present the behavior of weighted union on the following sequence of union starting from the initial configuration as p[i]=-count[i]=-1, 1<=i<=8,
Union(1,2), Union(3,4), Union(5,6) , Union(7,8), Union(1,3), Union(5,7), Union(1,5)

Assignment-3

1. a) What is the solution generated by the function job sequencing when n=7 ,
(P1,P2,….P7)=(3,5,20,18,1,6,30) and (d1,d2,….d7)=(1,3,4,3,2,1,2)
b) What is Greedy method and discuss its applications

2. a) Solve the following 0/1 Kanpsack problem using Dynamic Programming P=(11,21,31,33) , W=(2,11,22,15), m=40 , n=4.
b) Consider a four stage system with r1=0.9, r2=0.8, r3=0.5 and r4=0.6 and c1=30,c2=15,c3=20 and c4=10, where the total cost of the system is C<=135. Find the reliability design for the system.

3.Consider nn=5 and a1,a2,a3,a4,a5 identifiers , the values of p's and q's are given as p(1:5)=(0.15,0.10,0.05,0.10,0.20), q(0:4)=(0.05,0.10,0.05,0.05,0.05,0.10). Construct the Optimal Binary Search Tree(OBST).

Assignment-4

1.Find the optimal travelling salesperson tour using dynamic programming

0	5	2	3
4	0	1	5
4	2	0	3
7	6	8	0

2. Let w = {6, 15, 20, 10, 11, 18, 29} and m=35. Find all possible subsets of w that sum to m. Draw the portion of the state space tree that is generated

3. Draw the portion of state space tree generated by LCBB for the knapsack instances: n = 4; (P1; P2;P3 ; P4) = (10; 10; 12; 18); (w1w2;w3;w4) = (2;4; 6; 9) and M = 15

4. Solve the following 0/1 Knapsack problem using dynamic programming P= (10, 20, 30, 32), W= (3, 10, 21, 14), C=35, n=4.

19. Objective Questions

Unit-I

1. The time complexity of Strassen's Matrix multiplication algorithm is__ [ ]

   a) $O(\log n^7)$ b) $O(\log 7^n)$ c) $O(7^{\log n})$ d) None

2. If algorithm takes $O(n^2)$, it is faster for sufficiently larger n than if it had taken _____ [ ]

   a)O(log n) b) O(n) c) O(n log n) d) $O(2^n)$

3. The worst case time complexity of Quick sort is_____ [ ]

   a) O(n log n) b) $O(n^2)$ c) O(log n) d) None

4. The Average Time Complexity of Binary Search is_____ [ ]

   a) O(n log n) b) $O(n^2)$ c) O(log n) d) None

5. The Average Time Complexity of Merge Sort is _____ [ ]

   a) O(n log n) b) $O(n^2)$ c) O(log n) d) None

6. Which notation gives the lower bound of the function f(n)._____ [ ]

   a) O b) o c) w d)NONE

7. In Quick sort what is the characteristic of the pivot element_____

8. Time Complexity, T(p), of a program is the sum of _____ and _____

9. The f(n)= w(g(n)), if _____

10. Sort the following in ascending order.
    $O(n)$, $O(1)$, $O(2^n)$, $(\log n)$, $O(n^2)$, $O(n \log n)$, $O(n^3)$, $O(3^n)$

    _____

11. If P the actual problem is divided into k sub problems $P_1, P_2, \ldots P_k$ then the computing time of P, T(n) is given by
    _____
    where n_i is the number of input present in P_i.

12. Step by step procedure to solve a problem is known as_____

13. Define profiling.

14. The maximum stack space needed by Quick sort is _____

15. The data structure used to execute the recursive procedures is _____

16. The condition for merge sort is_____

17. The time Complexity of conventional matrix multiplication algorithm is_____

18. Define Algorithm

19. The amount of computer memory required to execute a program is known as_____

20. Write the recurrence relation to find the time complexity of recursive fibonacci series program_____.

Unit-II

1. The maximum no. of nodes in a 2-3 tree of Height **k** is _____

2. Write Weighted rule for Union.

3. Write the Collapsing rule for Find.

4. What is the time complexity of Weighted Union algorithm._____

5 In Reliability design problem U_i is called as [ ]
   a. The upper bound on no. of copies
   b. The lower bound on no. of copies
   c. Reliability d) none

6 The objective function of Traveling sales person problem is [ ]
   a. Cost of the tour must be maximum
   b. Time of the tour must be minimum
   c. Time of the tour must be maximum
   d. Cost of the tour must be minimum

7 The node for which the children is currently being generated is known as
a) Dead node b)Enode [ ]
c) Live node d) None

8 The node for which all the children are generated is known as
a)Dead node b)Enode [ ]
c)Live node d)None

9 The node for which the children is yet to be generated is known as
a)Dead node b)Enode [ ]
c)Live node d)None

10 ---------- Constraints are rules that restrict each x_i to take on values only from a given set [ ]
a)Explicit Constraints b)Implicit Constraints
c)Implied Constraints d)None

11 _____ determines which of the tuples (sets) in the solution space actually satisfy the criterion function [ ]
a) Explicit Constraints b)Implicit Constraints
c) Implied Constraints d)None

12 Depth First node generation with bounding functions is called___[ ]
a) Dynamic Programming b) Branch and Bound
c) Back Tracking d) None

13 _____ functions are used to kill the nodes of a state space tree [ ]
a) Explicit Functions b)Implicit Functions c) Bounding Functions d)None

14 Define Problem state.

15 Write the formula to test whether the two queens are in the same diagonal or not.

16 place(k, i) returns true if _____

17 Define Chromatic Number.

18 Number of nodes generated in the State Space Tree of a Graph Coloring problem with n vertices and with chromatic number m is _____

19 g(i , S)= min { }
_____

20 g(i , ϕ) = _____ [ ]
a) $C_i\phi$ b) Cϕi c) Cii d) C$\phi\phi$

21 In reliability design problem S_1^i = _____.

Unit-III

In greedy method only one decision sequence is ever generated. (True/False)

1. Multi-stage graph problem is classified under ……. category of problem.
   a) permutation selection problem
   b) subset selection problem
   c) both a and b
   d) none
2. If you construct a Binary search tree for the given identifier set {for,do,while,int,if}. Then the average no of comparisions required to search an identifier in the worst case is ……..
3. The time complexity of Heap Sort algorithm is [ ]

a) O(n log n)
b) O(log n)
c) $O(n^2)$
d) O(1)

4. In AVL Tree the Balancing factor of any node can be [ ]
a) 1,2,3
b) 1,0
c) 1,0,-1
d) None

5. In a 2-3 Tree each node should have _____ no of **keys.**[ ]
a) 2 b)3 c)5 d)None

6. In Sets the FIND operation returns [ ]
a) Parent node
b) Root node
c) Child node
d) None

7. In a Maximum Heap the value of a node is at least as large as its ____[ ]
a) Parent
b) Root
c) Sibling
d) Children

8. In _____ trees all the terminal nodes are at same level [ ]
a) 2-3 Trees
b) AVL Trees
c) Binary Trees
d) None

9. In Weighted Union Algorithm the set with minimum no. of nodes becomes
a) Root of the Resultant Tree [ ]
b) Child node at Level Two
c) AVL Tree
d) 2-3 Tree

10. The minimum no. of nodes in an AVL tree of Height ' **h**' is given by n(h)=_____

11. The minimum no. of nodes in a 2-3 tree of Height **k** is _____

12. Write the Objective function of Knapsack problem.

13. Define Optimal & feasible Solutions

14. Write the Control abstraction of Greedy Method.

Unit -IV

1. The time complexity of In-order Tree traversal algorithm is _____ [ ]
   a)Θ(n) b) Θ (log n) c) Θ (n^2) d) Θ (n^3)

2. The Space complexity of In-order Tree traversal algorithm is _____ [ ]
   a)O(n) b)O(log n) c)O(n^2) d)O(n^3)

3. The Space complexity of BFS algorithm is _____ [ ]
   a)Θ(n) b) Θ (log n) c) Θ (n^2) d) Θ (n^3)
4. The Time complexity of BFS algorithm(if adjacency matrix is used to represent the graph) is _____ [ ]
   a)Θ (n+E) b) Θ (log n) c) Θ (n^2) d) Θ (n^3)
5. The Time complexity of DFS algorithm(if adjacency list is used to represent the graph) is _____ [ ]
   a)Θ (n+E) b) Θ (log n) c) Θ (n^2) d) Θ (n^3)
6. The data structure used in BFS algorithm is [ ]
   a) Stack b)Queue c)Linked list d)None
7. The data structure used in DFS algorithm is [ ]
   a) Stack b)Queue c)Linked list d)None
8. _____ game is one in which there are no valid sequences of infinite length
   a)Infinite b)Finite c)Cricket d)None [ ]
9. Define the Instance of a Game Tree
10. The Problem reduction in computer can be represented by using [ ]
    a. Multistage graph
    b. Tree
    c. AND/OR graph
    d. OR/OR graph
11. In LC search the next E-node is selected based on _____ [ ]
    a) FIRST IN FIRST OUT b)LAST IN FIRST OUT
    c) RANKING FUNCTION d)All the ABOVE.
12. In LC search, the function of Least( ) is_____ [ ]
    a. To find the node with least rank
    b. To find least number of nodes
    c. To find the least value
    d. None
13. In 15-puzzle problem, g(x)= _____ [ ]
    a. The total number of Tiles.
    b. The number of non blank Tiles with even number
    c. The number of Tiles with ODD number
    d. The number of non blank Tiles not in their Goal Position.

14. Intelligent ranking function c(x)=_____ [ ]
    a. F(h(x)-g|(x)
    b. F(h(x))
    c. F(h(x))+g|(x)
    d. None

UNIT-V

1. The solution to the recurrence H(n) = n +root(n)· H([$\sqrt{n}$]) is [ ]

A. O(n).

B. O($\sqrt{n}$.).
C. O(n log log n).
D. O(log n).
E. O(log* n).

2. Given two graphs G1 and G2, deciding if one can delete k edges from G1 and get the graph G2 (we consider two graphs to be the same if one can rename the vertices of one graph to get the second graph) is
A. NP-complete.
B. Solvable in polynomial time.
C. NP-hard
D. None of the above.

3. Any problem that can be solved by an algorithm that uses only O(log n) bits of memory (in addition to the input, which resides in a read-only memory) can be solved using polynomial time." This statement is:
A. False.
B. True.
C. False only if P = NP.
D. True only if P = NP.

4.The problem Triple 2Coloring (deciding if the vertices of a graph G can be partitioned into three sets S, T, V , such that the induced subgraphs GS, GT and GV are each colorable by two colors) is
A. NP-Complete.
B. Solvable in polynomial time.
C. NP-Hard.
D. None of the above.

5 . The solution to the recurrence A(n) = A(log n) + 1 is:
A. O(1)
B. O(log log log n)
C. O(n log n).
D. O(n log n).

6). Given a boolean formula F of length n defined over 100 variables, deciding if F is satisfiable can be done in:
A. O(2n) time, and there is no faster algorithm.
B. O(log log n) time.
C. Polynomial time.
D. This is an NP-complete problem, and it cannot be solved.

20. Tutorial problems

Tutorial-1

1. Write an algorithm to evaluate a polynomial using Horner's Rule.
2. Given n Boolean variables x1, x2….xn. We wish to print all possible combinations of truth values they can assume. Write an algorithm to accomplish this.
3. Give both a Recursive and an Iterative algorithm to compute the binominal coefficient $\binom{n}{m}$
4. Ackermann's function A(m,n) is defined as follows.

$$A(m,n) = \begin{cases} n+1 & \text{if } m=0 \\ A(m-1, 1) & \text{if } n=0 \\ A(m-1, A(m, n-1)) & \text{otherwise} \end{cases}$$

Write both Recursive and Non Recursive algorithm for computing above function.

5. Determine the frequency counts for all state ments in the following two algorithm segments.

a)

```
1.  for  i: =    1 to n do
2.        for  j: =    1 to  i do
3.              for  k: =   1 to  j do
4.                    x: = x+1;
```

b)

```
        1. i: = 1;
2. while (i≤n) do
        3.   {
4.      x: = x+1;
        5.      i: = i+1;
        6.    }
```

6. Obtain the step count for the following algorithm using the Frequency method. Clearly show the step count table.

```
Algorithm mult (a, b, c, n)
{
for   i:= 1 to n do
    for j: = 1 to n do
{
c [i, j] :=  0;
for k := 1 to n do.
C [i, j] :=  c[i, j] + a[i, k] * b[k, j];
}
}
```

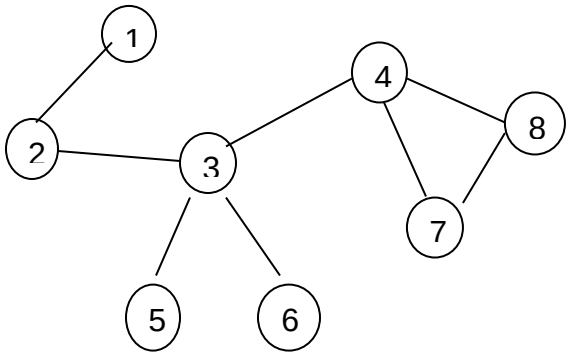
Tutorial-2

1. Show how Quicksort sorts the following sequences keys:
   5, 5, 8, 3, 4, 3, 2
2. Sort a file of n records which consists of scanning the file, merging consecutive pairs of size one, then merge pairs of size two, and so on. Write an algorithm that carries out this process. Show how your algorithm works on the data set (100, 300, 150, 450, 250, 350, 200, 400, 500).
3. Show how Quick sort sorts the following sequence of keys: 1, 1, 1, 1, 1, 1, 1 and 5, 5, 8, 3, 4, 3, 2
4. On which input data does the algorithm QuickSort exhibit its worst case behavior?

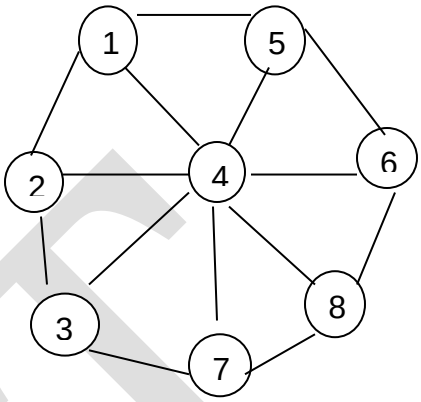
Tutorial-3

1. Present an algorithm Height union that uses the height rule for union operations instead of the weighting rule.

2. For the following graphs indentify the articulation points and draw the biconnected components.



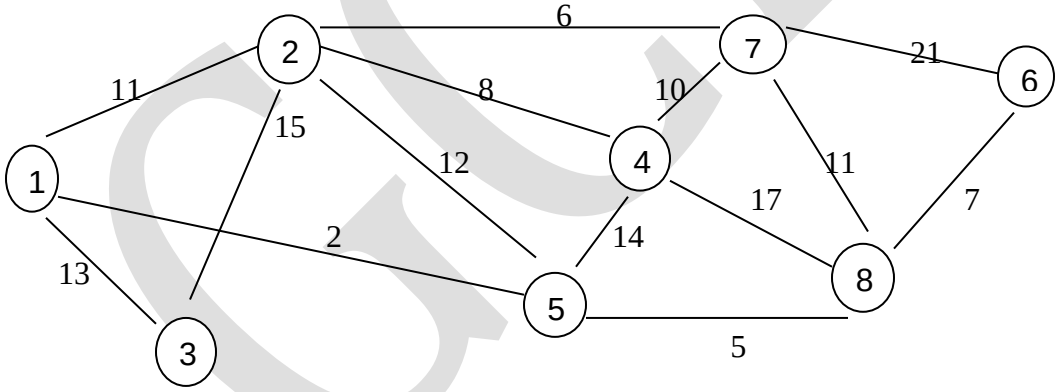
a)



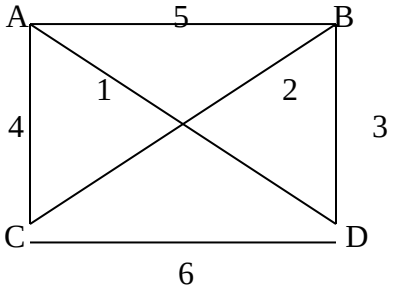
b)

Tutorial-4

1. Prove that if p1/w1≥, p2/w2≥,……≥pn/wn, then GreedyKnapsack generates an optimal solution to the given instance of the Knapsack problem.
2. Find an optimal solution to the Knapsack instance n=7, m=15, (p1, p2,….p7) = (10, 5, 15, 7, 6, 18, 3) and (w1, w2, w3…..w7) = (2,3,5,7,1,4,1).
3. Compute a minimum cost spanning tree for the following graph using
   a) Prim's Algorithm b) Kruskal's Algorithm.



4. Find the shortest path between all pairs of nodes in the following graph



Tutorial-5

1. Construct the Optimal Binary Search Tree when n=4 such that a1<a2<a3<a4 with
(q0, q1, q2, q3, q4) = (¼, 3/16, 1/16, 1/16, 1/16) and
(p1, p2, p3, p4) = (1/4, 1/8, 1/16, 1/16)
2. Find the optimal solution for the knapsack (p) instance when m=165, n=6, (p1,….p6) = (w1,…..w6) = (100, 50, 20, 10, 7, 3)
3. Consider the following Sum of Subsets problem instance. n=6, m=30, and w[1:6] ={5, 10, 12, 13, 15, 18}. Find all possible subsets of w that sum to m. Draw the portion of the state space tree that is generated.

4 Draw the portion of the state space tree generated by LCBB for the following Knapsack instances.n=5 (p1, p2….p5) = (10, 15, 6, 8, 4)

(w1, w2,……w5) = (4, 6, 3, 4,) and m=12

a) n=5 (p1, p2,…p5) = (w1, w2,….w5) = (4, 4, 5, 8, 9) and m=15.

21. Known gaps ,if any and inclusion of the same in lecture schedule : NO

22. Discussion topics

1. Time and Space complexity
2. Different Sorting algorithms
3. Applications of Divide and Conquer
4. Greedy technique applications
5. Dynamic programming
6. Comparison of Dynamic programming and Greedy technique
7. Application of Trees
8. Application of graphs
9. Tree and graph traversals
10. Applications of backtracking technique
11. Difference between LC Branch & Bound and FIFO branch and Bound
12. Deterministic and Non Deterministic algorithms

23. References , Journals, Websites and E-links

http://www.personal.kent.edu/~rmuhamma/Algorithms/algorithm.html
http://nptel.ac.in/courses/106101060/
http://freevideolectures.com/Course/2281/Design-and-Analysis-of-Algorithms#
http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap18.htm
http://www.cs.princeton.edu/courses/archive/spr11/cos423/Lectures/NP-completeness.pdf
https://www.ics.uci.edu/~eppstein/161/960312.html

24. Quality Measurement Sheets

Course End Survey

COURSE END SURVEY of II CSE-A **Faculty: A.Sree Lakshmi**

ACADEMIC YEAR : 2012-13	SEM : II	Date :22-04-2004
COURSE	DAA	CLASS : II CSE
FACULTY	A.SREE LAKSHMI	SECTION : A

Please evaluate on the following Scale:

Excellent(E)	Good(G)	Average(A)	Poor(P)	No Comment(NC)
5	4	3	2	1

SNO	QUESTIONAIRE	E 5	G 4	A 3	P 2	NC 1	Avg %
GENERAL OBJECTIVES:							
1)	Did the course achieve its stated objectives?	31	13	1	0	0	4.66(93.33%)
2)	Have you acquired the stated skills?	29	14	2	0	0	4.6(92%)
3)	Whether the syllabus content is adequate to achieve the objectives?	31	12	2	0	0	4.64(92.88%)
4)	Whether the instructor has helped you in acquiring the stated skills?	31	11	3	0	0	4.62(92.44%)
5)	Whether the instructor has given real life applications of the course?	30	12	3	0	0	4.6(92%)
6)	Whether tests, assignments, projects and grading were fair?	30	11	4	0	0	4.57(91.55%)
7)	The instructional approach (es) used was (were) appropriate to the course.	30	13	2	0	0	4.62(92.44%)
8)	The instructor motivated me to do my best work.	29	14	2	0	0	4.6(92%)
9)	I gave my best effort in this course.	27	15	3	0	0	4.53(90.66%)
10)	To what extent you feel the course outcomes have been achieved.	28	14	3	0	0	4.55(91.11%)
For Lab courses only							
11)	I was provided with adequate orientation and guidance for proceeding with laboratory activities.						
12)	The instructor(s) was (were) helpful in assisting with problems and difficulties in the lab.						
13)	Space & facilities were adequate for required activities of the lab.						
14)	Instructor provided material required for the lab.						
Please provide written comments							

a) What was the most effective part of this course

- Presentations very interesting to study topics
- Explanation
- Algorithms
- Teaching was good enough to understand the concepts

b) What are your suggestions, if any, for changes that would improve this course?

- Would be better if topics of similar methods are reduced

c)Given all that you learned as a result of this course, what do you consider to be most important?

- Implementation

d)Do you have any additional comments or clarifications to make regarding your responses to any particular survey item?

-

e)Do you have any additional comments or suggestions that go beyond issues addressed on this survey?

-

COURSE END SURVEY of II CSE-B Faculty: D.VENKATESWARLU

A. Y : 2014-15	SEM : II	Date :01-05-2015
COURSE	DAA	CLASS : II CSE
FACULTY	D.VENKATESWARLU	SECTION : B

Please evaluate on the following Scale:

Excellent(E)	Good(G)	Average(A)	Poor(P)	No Comment(NC)
5	4	3	2	1

SNO	QUESTIONAIRE	E 5	G 4	A 3	P 2	NC 1	Avg %
GENERAL OBJECTIVES:							
1)	Did the course achieve its stated objectives?	16	18	16	7	0	3.75(75.08)
2)	Have you acquired the stated skills?	13	23	14	7	0	3.73(74.73)
3)	Whether the syllabus content is adequate to achieve the objectives?	15	22	14	6	0	3.80(76.14)
4)	Whether the instructor has helped you in acquiring the stated skills?	12	22	14	9	0	3.64(72.98)
5)	Whether the instructor has given real life applications of the course?	12	18	18	9	0	3.57(71.57)
6)	Whether tests, assignments, projects and grading were fair?	13	18	17	9	0	3.61(72.28)
7)	The instructional approach (es) used was (were) appropriate to the course.	14	13	20	9	0	3.50(70.17)
8)	The instructor motivated me to do my best work.	16	14	19	8	0	3.66(73.33)
9)	I gave my best effort in this course.	15	18	17	7	0	3.71(74.38)
10)	To what extent you feel the course outcomes have been achieved.	18	13	18	8	0	3.75(74.38)
For Lab courses only							
11)	I was provided with adequate orientation and guidance for proceeding with laboratory activities.						
12)	The instructor(s) was (were) helpful in assisting with problems and difficulties in the lab.						
13)	Space & facilities were adequate for required activities of the lab.						
14)	Instructor provided material required for the lab.						
Please provide written comments							

b) What was the most effective part of this course

- Class Discussions
- Real Life examples given
- Learning subject
- Given more number of Assignments
- Computer Science Algorithm in Mathematical approach
- Everything is explained very clearly

b) What are your suggestions, if any, for changes that would improve this course?

- To conduct exams

c)Given all that you learned as a result of this course, what do you consider to be most important?

- Practicing
- To work day to day
- Application based algorithms(like graphs, trees etc.,)

d)Do you have any additional comments or clarifications to make regarding your responses to any particular survey item?

- To be consistent

COURSE END SURVEY of II CSE-C Faculty: A.Sree Lakshmi

ACADEMIC YEAR : 2012-13	SEM : II	Date :15-APR-2015
COURSE	DAA	CLASS : II CSE
FACULTY	A.SREE LAKSHMI	SECTION : C

Please evaluate on the following Scale:

Excellent(E)	Good(G)	Average(A)	Poor(P)	No Comment(NC)
5	4	3	2	1

SNO	QUESTIONAIRE	E 5	G 4	A 3	P 2	NC 1	Avg %
GENERAL OBJECTIVES:							
1)	Did the course achieve its stated objectives?	33	10	0	1	0	4.7(94%)
2)	Have you acquired the stated skills?	26	17	1	0	0	4.56(91.3%)
3)	Whether the syllabus content is adequate to achieve the objectives?	30	13	1	0	0	4.65(93.1%)
4)	Whether the instructor has helped you in acquiring the stated skills?	29	12	3	0	0	4.59(91.81%)
5)	Whether the instructor has given real life applications of the course?	33	10	0	1	0	4.7(94.09%)
6)	Whether tests, assignments, projects and grading were fair?	31	9	4	0	0	4.61(92.2%)
7)	The instructional approach (es) used was (were) appropriate to the course.	30	11	1	2	0	4.56(91.3%)
8)	The instructor motivated me to do my best work.	30	12	1	1	0	4.61(92.2%)
9)	I gave my best effort in this course.	33	9	2	0	0	4.7(94%)
10)	To what extent you feel the course outcomes have been achieved.	31	12	0	0	1	4.63(92.7%)
For Lab courses only							
11)	I was provided with adequate orientation and guidance for proceeding with laboratory activities.						
12)	The instructor(s) was (were) helpful in assisting with problems and difficulties in the lab.						
13)	Space & facilities were adequate for required activities of the lab.						
14)	Instructor provided material required for the lab.						
Please provide written comments							

c) What was the most effective part of this course

-

b) What are your suggestions, if any, for changes that would improve this course?

-

c)Given all that you learned as a result of this course, what do you consider to be most important?

-

d)Do you have any additional comments or clarifications to make regarding your responses to any particular survey item?

-

e)Do you have any additional comments or suggestions that go beyond issues addressed on this survey?

-

COURSE END SURVEY of II CSE-D Faculty: A.Sree Lakshmi

ACADEMIC YEAR : 2012-13	SEM : II	Date :APR-2015
COURSE	DAA	CLASS : II CSE
FACULTY	A.SREE LAKSHMI	SECTION : D

Please evaluate on the following Scale:

Excellent(E)	Good(G)	Average(A)	Poor(P)	No Comment(NC)
5	4	3	2	1

SNO	QUESTIONAIRE	E 5	G 4	A 3	P 2	NC 1	Avg %
GENERAL OBJECTIVES:							
1)	Did the course achieve its stated objectives?	27	17	2	0	0	4.5(90.86%)
2)	Have you acquired the stated skills?	24	21	1	0	0	4.5(90%)
3)	Whether the syllabus content is adequate to achieve the objectives?	25	18	3	0	0	4.47(89.56%)
4)	Whether the instructor has helped you in acquiring the stated skills?	33	12	1	0	0	4.69(93.9%)
5)	Whether the instructor has given real life applications of the course?	28	18	0	0	0	4.6(92.17%)
6)	Whether tests, assignments, projects and grading were fair?	30	15	1	0	0	4.63(92.60%)
7)	The instructional approach (es) used was (were) appropriate to the course.	28	14	4	0	0	4.52(90.43%)
8)	The instructor motivated me to do my best work.	33	11	2	0	0	4.67(93.47%)
9)	I gave my best effort in this course.	26	16	4	0	0	4.47(89.56%)
10)	To what extent you feel the course outcomes have been achieved.	25	18	3	0	0	4.47(89.56%)
For Lab courses only							
11)	I was provided with adequate orientation and guidance for proceeding with laboratory activities.						
12)	The instructor(s) was (were) helpful in assisting with problems and difficulties in the lab.						
13)	Space & facilities were adequate for required activities of the lab.						
14)	Instructor provided material required for the lab.						
Please provide written comments							

d) What was the most effective part of this course

- Logical Thinking
- Real Life Applications
- Regular follow up and clearing students doubts
- Algorithms
- Analyze algorithms and improve efficiency of algorithms
- Performance of Algorithms
- Better for understanding
- Motivative classes

b) What are your suggestions, if any, for changes that would improve this course?

- More indepth explanation of Branch and Bound
- Few algorithms were complex
- More Practical thinking to be developed & conduct tests
- Teaching with the help of videso to get a real view
- Giving more and more examples

c)Given all that you learned as a result of this course, what do you consider to be most important? • Basics • Techniques of alg design • Estimate Performance of Algorithms • Diff techniques of designing algorithms
d)Do you have any additional comments or clarifications to make regarding your responses to any particular survey item?
e)Do you have any additional comments or suggestions that go beyond issues addressed on this survey?

TEACHING EVALUATION

Course Assessment

Class:II CSE-A **A.Y: 2014-15**
Subject: DAA **Sem: II**
Faculty: A. SREE LAKSHMI

Assessment	Criteria Used	Attainment Level			Remarks
Direct(d)	**Theory:**				
	External Marks	81.14%	76.79%	76.79%	
	Internal Marks(Theory)	51%			
	Assignments	91.23%			
	Tutorials	82.8%			
	Lab:				
	Internal Marks	NA			
	External Marks	NA			
Indirect(id)	Course End Survey			92.04%	
Course Assessment(06*d+0.4*id)				82.87%	

Course Assessment

Class:II CSE-B **A.Y: 2014-15**
Subject: DAA **Sem: II**
Faculty: D.VENKATESWARLU

Assessment	Criteria Used	Attainment Level			Remarks
Direct(d)	**Theory:**				
	External Marks	89.47%	78.61%	78.61%	
	Internal Marks(Theory)	36%			
	Assignments	100%			
	Tutorials	89%			
	Lab:				

	Internal Marks	-----			
	External Marks	-----			
Indirect(id)	Course End Survey			92%	
Course Assessment(06*d+0.4*id)				83.96%	

Course Assessment

Class:II CSE-C **A.Y: 2014-15**
Subject: DAA **Sem: I**
Faculty: A. SREE LAKSHMI

Assessment	**Criteria Used**	**Attainment Level**			**Remarks**
Direct(d)	**Theory:**				
	External Marks	80%	66.905%	66.905%	
	Internal Marks(Theory)	30%			
	Assignments	82.5%			
	Tutorials	75.12%			
	Lab:				
	Internal Marks	NA	NA		
	External Marks	NA			
Indirect(id)	Course End Survey			92.7%	
Course Assessment(06*d+0.4*id)				77.22%	

Course Assessment

Class:II B.Tech II CSE-D **A.Y: 2014-15**
Subject: DAA **Sem: II**
Faculty: A.Sree Lakshmi

Assessment	**Criteria Used**	**Attainment Level**			**Remarks**
Direct(d)	**Theory:**				
	External Marks	88%	84.57%	84.57%	
	Internal Marks(Theory)	70.37%			
	Assignments	90.74%			
	Tutorials	89.17%			
	Lab:				
	Internal Marks	NA	NA		
	External Marks	NA			
Indirect(id)	Course End Survey			91%	
Course Assessment(06*d+0.4*id)				87.22%	