JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) PULIVENDULA

Department of Computer Science & Engineering

IIIrd –I B.TECH-DESIGN AND ANALYSIS OF ALGORITHMS

Lesson Plan

Course Title	**:**	**DESIGN AND ANALYSIS OF ALGORITHM**			
Course Structure	**:**	Lectures	Tutorials	Practicals	Credits
		4	0	0	4
Course Coordinator	**:**	Mr K.V.Siva Prasad Reddy, Assistant Prof(Adhoc)			
Team of Instructors	**:**	Mr. G.Murali (HOD)			

I. Course Overview

Introduction to fundamental techniques for designing and analyzing algorithms, including asymptotic analysis; divide-and-conquer algorithms and recurrences; greedy algorithms; data structures; dynamic programming; graph algorithms; and randomized algorithms.

II. Prerequisite(s):

Level	Credits	Periods / Week	Prerequisites
UG	4	4	Design And Analysis of Algorithms

III. Assessment:

FORMATIVE ASSESMENT	
Mid Semester Test I for 40 Marks in first 21/2 units is conducted at the starting of 9th week. Mid Semester Test II for 40 Marks in next 21/2 is conducted at the end of the course work. Average of two tests is taken as final	30 Marks
Total (Formative)	30 Marks
SUMMATIVE ASSESMENT	
End Semester Examination in all units is conducted for 60 Marks	70 marks
Grand Total	100 Marks

IV. Course objectives:

Upon completion of this course, students will be able to do the following:

- Analyze the asymptotic performance of algorithms.
- Write rigorous correctness proofs for algorithms.
- Demonstrate a familiarity with major algorithms and data structures.
- Apply important algorithmic design paradigms and methods of analysis.
- Synthesize efficient algorithms in common engineering design situations.

V. Course Outcomes:

Students who complete the course will have demonstrated the ability to do the following:

- Argue the correctness of algorithms using inductive proofs and invariants.
- Analyze worst-case running times of algorithms using asymptotic analysis.
- Describe the divide-and-conquer paradigm and explain when an algorithmic design situation calls for it. Recite algorithms that employ this paradigm. Synthesize divide-and-conquer algorithms. Derive and solve recurrences describing the performance of divide-and-conquer algorithms.
- Describe the dynamic-programming paradigm and explain when an algorithmic design situation calls for it. Recite algorithms that employ this paradigm. Synthesize dynamic-programming algorithms, and analyze them.
- Describe the greedy paradigm and explain when an algorithmic design situation calls for it. Recite algorithms that employ this paradigm. Synthesize greedy algorithms, and analyze them.
- Explain the major graph algorithms and their analyses. Employ graphs to model engineering problems, when appropriate. Synthesize new graph algorithms and algorithms that employ graph computations as key components, and analyze them.
- Explain the different ways to analyze randomized algorithms (expected running time, probability of error). Recite algorithms that employ randomization. Explain the difference between a randomized algorithm and an algorithm with probabilistic inputs.
- Analyze randomized algorithms. Employ indicator random variables and linearity of expectation to perform the analyses. Recite analyses of algorithms that employ this method of analysis.
- Explain what amortized running time is and what it is good for. Describe the different methods of amortized analysis (aggregate analysis, accounting, potential method). Perform amortized analysis.
- Explain what competitive analysis is and to which situations it applies. Perform competitive analysis.
- Compare between different data structures. Pick an appropriate data structure for a design situation.
- Explain what an approximation algorithm is, and the benefit of using approximation algorithms. Be familiar with some approximation algorithms, including algorithms that are PTAS or FPTAS. Analyze the approximation factor of an algorithm.

VI. Program outcomes:

a. An ability to apply knowledge of software engineering, literature survey, module specifications, and computer science and engineering theory in the modeling and design of computer-based systems to real-world problems (fundamental engineering analysis skills)

b An ability to design and conduct experiments, as well as to analyze and interpret the software information (information retrieval skills)

c An ability to design , implement, and evaluate a computer-based system, process, component, module or program to meet desired needs, within realistic constraints such as economic, environmental, social, political, health and safety, manufacturability, and sustainability (Creative Skills) requirements.

d An ability to function effectively on multi-disciplinary teams (team work).

e An ability to analyze a problem, identify, formulate and use the appropriate computing and engineering requirements for obtaining its solution (engineering problem solving skills).

f An understanding of professional, ethical, legal, security and social issues and responsibilities (professional integrity)

g An ability to communicate effectively both in writing and orally (speaking / writing skills) with customers (stakeholders).

h The broad education necessary to analyze the local and global impact of computing and engineering solutions on individuals, organizations, and society (engineering impact assessment skills).

i Recognition of the need for, and an ability to engage in continuing professional development and life-long learning (continuing education awareness).

j A Knowledge of contemporary issues (social awareness).

k An ability to use current techniques, skills, and tools necessary for computing and engineering practice (practical engineering analysis skills).

l An ability to apply design and development principles in the construction of software and hardware systems of varying complexity (software hardware interface).

m An ability to recognize the importance of professional development by pursuing postgraduate studies or face competitive examinations and research works that offer challenging and rewarding careers in computing (successful career and immediate employment).

VII. Syllabus:

UNIT-I

Introduction to Algorithm, Algorithm specification , performance analysis, Asymptotic Notations

Divide and Conquer :General Method, Binary search, finding the maximum and minimum ,merge sort, Quick sort ,Selection , Stressen's matrix multiplication ,Analysis of Divide and Conquer run time Recurrence Relations.

UNIT-II

Greedy Method: General Method ,Knapsack problem , job scheduling with deadlines, minimum cost spanning trees-Prim's and Kruskal's Algorithm, Optimal storage on tapes ,single source shortest paths- Dijkastra's ,Bell Man ford And Wars hall's Algorithm.

Dynamic Programming: General Method, Multi stage graphs , All-pairs shortest paths, optimal binary search trees,0/1 knapsack ,the travelling sales person problem.

UNIT-III

Basic Traversal and search techniques: Techniques for binary trees, techniques for graphs ,AND/OR Graphs, connected components and spanning trees, Bi-Connected components and Dfs

Back tracking: General Method,8-queens problem ,sum of subsets problem, graph coloring and Hamiltonian cycles, knapsack problem.

UNIT-IV

Branch bound: The method ,Travelling salesperson,0/1 knapsack problem, efficiency considerations

Lower Bound Theory: Comparision trees of sorting and Searching , Lower bound through reductions-multiplying triangular matrices ,inverting a lower trainger matrix,computing the transitive closure.

UNIT-V

Network Flow Problems:

NP-Hard and NP-Complete problems:Np Hardness ,Scheduling Problems ,NP-Completeness, Cook's theorem (with out proof),Reductions for clique Decision problem ,Chromatic number decision problem .

Text Books

1. "Fundamentals of computer Algorithms:,Ellis Horowitz,s.satraj sahani and raja sekharan,2nd edition,university press.
2.Algorithm design –Jon Kleinberg and Eva Tardos,pearson.
3. Anany Levitin, "Introduction to the Design and Analysis of Algorithms", Third Edition, Pearson Education, 2012.

References

4. :Design and Analysis of algorithms",Aho,Ullman and Hopcroft,pearson education.

5. "Algorithms"-Richard Johnson baugh and Marcus Schaefer,pearson Education.
6 .Thomas H.Cormen, Charles E.Leiserson, Ronald L. Rivest and Clifford Stein, "Introduction to Algorithms", Third Edition, PHI Learning Private Limited, 2012.

7. Donald E. Knuth, "The Art of Computer Programming", Volumes 1& 3 Pearson Education, 2009. Steven S. Skiena, "The Algorithm Design Manual", Second Edition, Springer, 2008.

VIII. Course Plan:

The course plan is meant as a guideline. There may probably be changes.

Session No	Topics to be covered	**Dates**	**Ref, Page No**
1	Introduction, Algorithm, Notion of algorithm	03/07/2019	1 (1-7) 2(5-11)
2	Fundamentals of Algorithmic Problem Solving-steps in designing and analyzing an algorithm	03/07/2019	1 (9-16) 2(30-39)
3	***Important Problem Types-Sorting.searching, string processing, graph, geometric, numeric and combinatorial problems***	06/07/2019	1(18-23)
4	Fundamentals of the Analysis of Algorithm Efficiency, Analysis Framework, measuring input size, units for measuring running time,	10/07/2019	1(41-45)
5	Analysis Framework, Orders of growth, worst, best and average case analysis, recapitulation of Analysis Framework.	13/07/2019	1(45-50) 2(23-29)
6	Asymptotic Notations and its properties- Informal, Big-oh, Big-omega, Big-theta notation	13/07/2019	1(52-58) 2(43-53) 4(31-57)
7	Mathematical analysis for Non-recursive algorithms –General plan for analyzing time efficiency	17/07/2019	1(61-67)
8	Mathematical analysis for Recursive algorithms –General plan for analyzing time efficiency	17/07/2019	1(70-76) 2(65-75)
9	Sample problems	17/07/2019	1(8,16-18,)
10	Brute Force-selection.bubble sort,	20/07/2019	1(97-101)
11	Brute Force -sequential search,brute force string manipulation	24/07/2019	1(104-106)
12	Closest-Pair and Convex-Hull Problems	27/07/2019	1(108-113)
13	Traveling Salesman Problem - Knapsack Problem	27/07/2019	1(115-119) 3(401-405)

14	Assignment problem	27/07/2019	1(119-121)
			
			4(498-501)

15	Divide and conquer methodology master theorem	31/07/2019	1(169-172) 2(93-97),5

16	Merge sort,Quick sort	03/08/2019	1(172-181)
			
			2(170-182)
			
			4(120-129)

17	Binary search-Binary tree traversal and properties.	07/08/2019	1(181-185)
			4(132-139)

18	Multiplication of Large Integers – Strassen's Matrix Multiplication	07/08/2019	1(186-191),
			2(75-82)
			4(135-137)

19	Closest-Pair and Convex-Hull Problems. By divide and conquer rule	10/08/2019	1(192-197)
20	Brute Force-selection.bubble sort,	10/08/2019	1(283-287)
21	Brute Force -sequential search,brute force string manipulation	14/08/2019	1(287-290)
22	Closest-Pair and Convex-Hull Problems	14/08/2019	1(217-225)

23	Traveling Salesman Problem - Knapsack Problem	28/08/2019	1(226-237)
			4(210-212)

24	Assignment problem	28/08/2019	1(241-255)
			
			2(397-403)

25	Divide and conquer methodology master theorem	31/08/2019	1(249-257)
			
			2(425-427)
			
			4(427-431)

26	Merge sort,Quick sort	04/09/2019	1(315-322)
			
			2(634-636)

27	Binary search-Binary tree traversal and properties.	07/09/2019	1(325-331)
			
			2(631-633)

28	Multiplication of Large Integers – Strassen's Matrix Multiplication	07/09/2019	1(333-337)

			2(658-662)
29	Closest-Pair and Convex-Hull Problems. By divide and conquer rule	11/09/2019	1(338-343)

30	Brute Force-selection.bubble sort,	11/09/2019	1(345-351)
			
			2(846-850)

31	Brute Force -sequential search,brute force string manipulation	14/09/2019	1(351-359)
			
			2(864-878)

32	Closest-Pair and Convex-Hull Problems	14/09/2019	1(361-369)
			
			2(708-714)

33	Traveling Salesman Problem - Knapsack Problem	18/09/2019	1(369-371)
			
			4(258-262)

34	Assignment problem	18/09/2019	1(372-375)
			
			2(732-735)

35	Divide and conquer methodology master theorem	25/09/2019	1(375-378)
			
			2(732-735)
			
			4(217-222)

36	Merge sort,Quick sort	25/09/2019	1(380-381),
37	Binary search-Binary tree traversal and properties.	28/09/2019	1(381-383)
38	Multiplication of Large Integers – Strassen's Matrix Multiplication	05/10/2019	1(387-392)
39	Closest-Pair and Convex-Hull Problems. By divide and conquer rule	05/09/2019	1(394-397)
40	Limitations of Algorithm Power- Lower-Bound Arguments-Methods for establishing lower bounds	09/10/2019	1(401-409)

41	Decision Trees- Decision Trees for sorting and
	searching in sorted arrays.
		12/10/2019	1(423-425)
			
			4(231-238)

42	P, NP and NP-Complete Problems	16/10/2019	1(426-430)

43	Coping with the Limitations of algorithm power-backtracking	19/10/2019	1(432-436)

44	n-Queens problem – Hamiltonian Circuit Problem-
	Subset sum problem
		19/10/2019	1(436-440)
			
			4(427-430)
			
			4(533-537)

45	Branch and Bound – Assignment problem	23/10/2019	1(778-788)
			
			2(1048-1053)

46	Knapsack Problem – Traveling Salesman Problem	26/10/2019	1(789-792)
			
			2(1049-1052)

47	Approximation Algorithms for NP – Hard Problems – Traveling Salesman problem – Knapsack problem.	26/10/2019	1(793-795)

IX. Mapping course outcomes leading to the achievement of the program outcomes:

Course Outcomes	**Program Outcomes**												
	A	**b**	**C**	**D**	**E**	**f**	**G**	**h**	**i**	**J**	**k**	**l**	**m**
1	S	H											
2			H		S								
3			H										
4				H									S
5					H								S
6					H								
7			H										
8											H		
9								H					
10								H					

S = Supportive **H = Highly Related**

Justification of Course syllabus covering Course Outcomes:

By covering the syllabus a student can understand the various algorithmic techniques and easily solved any type problem.Student is able to develop the case analysis and performance analysis through no of notations.

Justification of CO's –PO's Mapping Table:

By mapping CO-1 to the PO's A & B which are related to the course CO1: The student is able to analyze and Implement Problems

By mapping CO-2 to the PO's C & E, which are related to the course CO2: The student is able to analyze the problem and solutions using various software architecture analyzing approaches.

By mapping CO-3 to the PO's C which are related to the course CO3: The student is able to understand the purpose of different software engineering project teams.

By mapping CO-4 to the PO's D & S which are related to the course CO4: The student is able to understand the Purpose of different software engineering architecture plans.

By mapping CO-5 to the PO's E & S which are related to the course CO5: The student is able to understand the Purpose of different architecture description languages.

By mapping CO-6 to the PO's E which are related to the course CO6: The student is able to understand the concept of architecture implementation and various goals of analysis.

By mapping CO-7 to the PO's C which are related to the course CO7: The student is able to different conceptual and technical skills in the analysis and design.

By mapping CO-8 to the PO's K which are related to the course CO8: The student is able to understand the software engineering CASE tools.

By mapping CO-9 to the PO's H which are related to the course CO9: The student is able to understand the purpose of why we are going for ADL and its implementation.

By mapping CO-10 to the PO's H which are related to the course CO10: The student is able to develop small projects using software architecture concepts and this knowledge will help him in further studies and industry needs.