NARSIMHA REDDY ENGINEERING COLLEGE
(Autonomous)
Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad
Accredited by NBA & NAAC with A Grade

COURSE FILE

Program Name :

Name of the Course: DESIGN AND ANALYSIS OF ALGORITHM

Course Code : AM3102PC

Semester and Year : I/III

Faculty Name : DHANANJAY

S.No	Contents	Included
1	Vision, Mission, COs, POs, PSOs, PEOs	
2	Academic calendar	
3	Syllabus	
4	CO/PO mapping	
5	Nominal Rolls of the Students	
6	Time table	
7	Lesson Plan	
8	Unit wise Question Bank	
9	Old Question Papers	
10	Question Papers (CIA & SEE)	
11	Tutorial sheets	
12	Learning Methodologies: Experiential learning (Industrial visits, Internships, Mini Projects, Academic Projects, Guest Lectures, Student Workshops etc.,), Problem Solving methodologies (assignments, quiz, case study etc.) **Note:1. At least TWO learning Methodologies to be included in your course** **2. The above methodologies for illustration, you may add more**	
13	Subject notes/PPTs/self study material	
14	Feedback on Curriculum Design and development	
15	CO/PO attainment, analysis and Action taken report	

. **1. Department Vision**

To evolve as a center of excellence with international reputation by adapting the rapid advancements in the computer specialization fields.

2. Department Mission

To inculcate professional behavior, strong ethical values, innovative research capabilities and leadership abilities in the young minds so as to work with a commitment to the progress of the nation.

3. COURSE OUTCOMES:

1. Ability to analyze the performance of algorithms
2. Ability to choose appropriate data structures and algorithm design methods for a specified application
3. Ability to understand how the choice of data structures and the algorithm design methods impact the performance of programs

4.COURSE OBJECTIVES:

1. Introduces the notations for analysis of the performance of algorithms.
2. Introduces the data structure disjoint sets.
3. Describes major algorithmic techniques (divide-and-conquer, backtracking, dynamic programming, greedy, branch and bound methods) and mention problems for which each technique is appropriate;
4. Describes how to evaluate and compare different algorithms using worst-, average-, and best- case analysis.
5. Explains the difference between tractable and intractable problems, and introduces the problems that are P, NP and NP complete.

5 .PROGRAM EDUCATIONAL OBJECTIVES OF CSE:

PEO#1	To apply the knowledge of mathematics, basic sciences and engineering solving the real world computing problems to succeed in higher education and professional careers.
PEO#2	To develop the skills required to comprehend, analyze, design and create innovative computing products and solutions for real life problems.
PEO#3	To inculcate professional and ethical attitude, communication and teamwork skills, multi-disciplinary approach and an ability to relate computer engineering issues with social awareness.

PROGRAMME OUTCOMES OF CSE:

PO1. Engineering knowledge: Apply the knowledge of basic sciences and fundamental engineering concepts in solving engineering problems.

PO2. Problem analysis: Identify and define engineering problems, conduct experiments and investigate to analyze and interpret data to arrive at substantial conclusions.

PO3. Design/development of solutions: Propose an appropriate solution for engineering problems complying with functional constraints such as economic, environmental, societal, ethical, safety and sustainability.

PO4. Conduct investigations of complex problems: Perform investigations, design and conduct experiments, analyze and interpret the results to provide valid conclusions.

PO5. Modern tool usage: Select or create and apply appropriate techniques and IT tools for the design & analysis of the systems.

PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. Environment and sustainability: Demonstrate professional skills and contextual reasoning to assess environmental or societal issues for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multi-disciplinary situations.

PO10. Communication: Communicate effectively among engineering community, being able to comprehend and write effectively reports, presentation and give / receive clears instructions.

PO11. Project management and finance: Demonstrate and apply engineering & management principles in their own / team projects in multidisciplinary environment.

PO12. Life-long learning: Recognize the need for, and have the ability to engage in independent and lifelong learning.

1. Knowledge of contemporary issues.
2. An ability to apply design and development principles in producing software systems of varying complexity using various project management tools.
3. An ability to identify, formulate and solve innovative engineering problems.

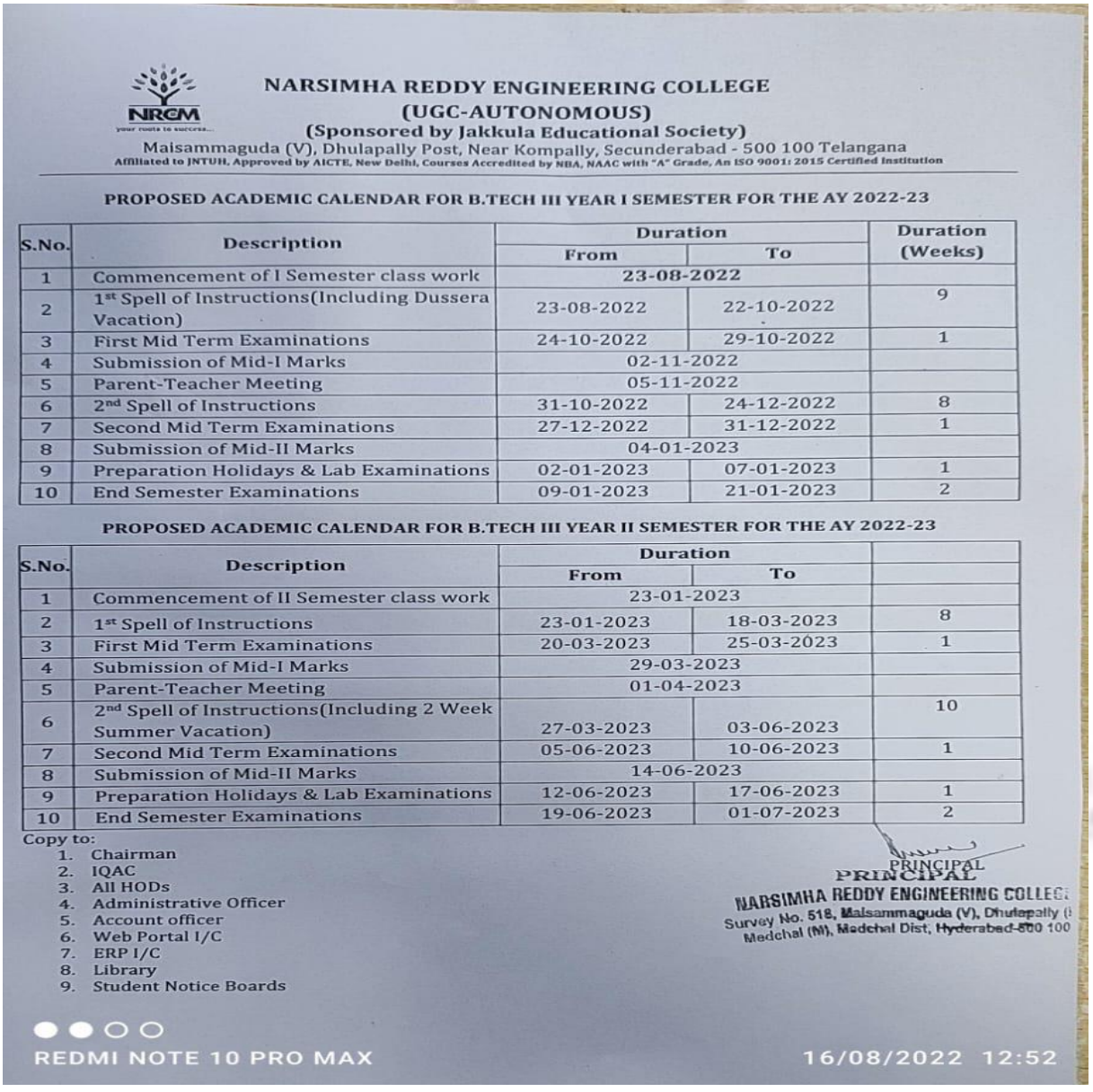
PROGRAM SPECIFIC OUTCOMES

PSO1: To provide effective and efficient real time solutions using acquired knowledge in various domains to crack problem using suitable mathematical analysis, data structure and suitable algorithm.

PSO2: To develop environmental and sustainable engineering solution having global and societal context using modern IT tools.

PSO3: To exhibit professional and leadership skills with ethical values dealing diversified projects with excellent communication and documentation qualities.

Academic Calendar



NARSIMHA REDDY ENGINEERING COLLEGE
(UGC-AUTONOMOUS)
(Sponsored by Jakkula Educational Society)
Maisammaguda (V), Dhulapally Post, Near Kompally, Secunderabad - 500 100 Telangana
Affiliated to JNTUH, Approved by AICTE, New Delhi, Courses Accredited by NBA, NAAC with "A" Grade, An ISO 9001: 2015 Certified Institution

PROPOSED ACADEMIC CALENDAR FOR B.TECH III YEAR I SEMESTER FOR THE AY 2022-23

S.No.	Description	Duration		Duration (Weeks)
		From	To	
1	Commencement of I Semester class work	23-08-2022		
2	1st Spell of Instructions(Including Dussera Vacation)	23-08-2022	22-10-2022	9
3	First Mid Term Examinations	24-10-2022	29-10-2022	1
4	Submission of Mid-I Marks	02-11-2022		
5	Parent-Teacher Meeting	05-11-2022		
6	2nd Spell of Instructions	31-10-2022	24-12-2022	8
7	Second Mid Term Examinations	27-12-2022	31-12-2022	1
8	Submission of Mid-II Marks	04-01-2023		
9	Preparation Holidays & Lab Examinations	02-01-2023	07-01-2023	1
10	End Semester Examinations	09-01-2023	21-01-2023	2

PROPOSED ACADEMIC CALENDAR FOR B.TECH III YEAR II SEMESTER FOR THE AY 2022-23

S.No.	Description	Duration		
		From	To	
1	Commencement of II Semester class work	23-01-2023		
2	1st Spell of Instructions	23-01-2023	18-03-2023	8
3	First Mid Term Examinations	20-03-2023	25-03-2023	1
4	Submission of Mid-I Marks	29-03-2023		
5	Parent-Teacher Meeting	01-04-2023		
6	2nd Spell of Instructions(Including 2 Week Summer Vacation)	27-03-2023	03-06-2023	10
7	Second Mid Term Examinations	05-06-2023	10-06-2023	1
8	Submission of Mid-II Marks	14-06-2023		
9	Preparation Holidays & Lab Examinations	12-06-2023	17-06-2023	1
10	End Semester Examinations	19-06-2023	01-07-2023	2

Copy to:
1. Chairman
2. IQAC
3. All HODs
4. Administrative Officer
5. Account officer
6. Web Portal I/C
7. ERP I/C
8. Library
9. Student Notice Boards

PRINCIPAL
PRINCIPAL
NARSIMHA REDDY ENGINEERING COLLEGE
Survey No. 518, Maisammaguda (V), Dhulapally (
Medchal (M), Medchal Dist, Hyderabad-500 100

REDMI NOTE 10 PRO MAX 16/08/2022 12:52

MAPPING OF COURSE OUTCOMES WITH PROGRAMME OUTCOMES:

CO/PO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO1 0	PO1 1	PO1 2
CS3102PC.1	3	3	3	-	-	-	-	-	2	2	-	2
CS3102PC.2	3	3	3	-	-	-	-	-	2	2	-	2
CS3102PC.3	3	2	3	-	-	-	-	-	2	2	-	2

MAPPING OF COURSE OUTCOMES WITH THE PROGRAM SPECIFIC OUTCOMES:

CO/PSO	PSO1	PSO2	PSO3
CS3102PC.1	3	3	3
CS3102PC.2	3	3	3
CS3102PC.3	3	3	3

5. Nominal Rolls of the Students

1	20X01A6601	AREDDY SATHVIK REDDY
2	20X01A6602	BADDAM MANIDEEP REDDY
3	20X01A6603	BOJJA SURESH
4	20X01A6604	CHOKKA DEVI VARA PRASAD
5	20X01A6605	DASOJU BHANU PRAKASH
6	20X01A6606	BURRA SRIKARUN
7	20X01A6607	EEDA THARUN REDDY
8	20X01A6608	JOSHUA RUFUS
9	20X01A6609	KATTA SAMYUKTHA
10	20X01A6610	K.AJAY
11	20X01A6611	MAMIDIPALLY MOUNIKA RAJESHWARI
12	20X01A6612	YEDDULA BHARGAVA REDDY
13	20X01A6613	NALLAVENI SIDDU ABHILASH
14	20X01A6614	NOMULA VIJAY
15	20X01A6615	PODDATURI ADITHYA
16	20X01A6616	PONNABOINA VINAYAK YADAV
17	20X01A6617	PANDILLAPALLI VISHNU VARDHAN REDDY
18	20X01A6618	SAKETH KULAKARNI
19	20X01A6619	SYEED AZHAR
20	20X01A6620	DUMPALA VENKATA SAI

21	21X05A6601	ANTIGARI SREEKANTH
22	21X05A6602	ANUGURTHI HARSHA TEJA
23	21X05A6603	ARRA POOJA
24	21X05A6604	BHUPATHI PURNA CHANDRA RAO
25	21X05A6605	BOMMA SAI GANESH
26	21X05A6606	CHARKA JITENDER DEVENDER
27	21X05A6607	CHILEKESI AJAY KUMAR
28	21X05A6608	CHILUMULA KRANTHI KUMAR
29	21X05A6609	DODLA VIVEK REDDY
30	21X05A6610	GANDRA NAGENDHAR
31	21X05A6611	GOLIPALLY DINESH REDDY
32	21X05A6612	GUVVA VEERESH
33	21X05A6613	KALAGANI BHAVITHA
34	21X05A6614	KOMATI REDDY NAVEEN REDDY
35	21X05A6615	KONDABOINA RADHA KRISHNA
36	21X05A6616	KOTA HEMA LATHA
37	21X05A6617	KOTAGIRI KRISHNA
38	21X05A6618	KUMMARI RAMESH
39	21X05A6619	MAGIRI DHARMA TEJA
40	21X05A6620	MANGALAPALLI VISHAL KUMAR
41	21X05A6621	MARIKANTI CHAITANYA ACHUTHA
42	21X05A6622	MOHAMMAD TAJUDDIN
43	21X05A6623	PAALINI MAHESH
44	21X05A6624	PALAKURTHI SAI KUMAR
45	21X05A6625	PANTLA SAI KUMAR
46	21X05A6626	R RESHMA
47	21X05A6627	RANGANAGARI KEERTHANA
48	21X05A6628	RANGANAMONI SATHISH
49	21X05A6629	S AKHIL
50	21X05A6630	S DHINAKAR
51	21X05A6631	SALIKALA ANIL
52	21X05A6632	SHAIK SHAREEF
53	21X05A6633	SINGAVARAPU CHANDRAKANTH
54	21X05A6634	SYED MOHAMMED SHAFI
55	21X05A6635	THUPAKULA VAMSHI KRISHNA
56	21X05A6636	N.DIVISHA
57	21X05A6637	SAI RAM
58	21X05A6638	CH.ADITYA

CLASS TIME TABLE: **CLASS: III/I CSE - AI&ML**

	1	2	3	4		5	6	7
HOUR/DAY	9:30AM - 10:20AM	10:20AM - 11:10AM	11:10AM - 12:00PM	12.00PM- 12.50PM	12:50PM – 1:40PM	1:40PM - 2:30PM	2:30PM - 3:20PM	3:20 PM - 4.10 PM
MON	AI ML				LUNCH			
TUE								
WED				AI ML				
THU		DAA LAB(AI ML)						AI ML
FRI			AI ML					
SAT								

INDIVIDUAL TIME TABLE

Mr.Y.DHANANJAY

	1	2	3	4		5	6	7
HOUR/DAY	9:30AM - 10:20AM	10:20 AM - 11:10 AM	11:10AM - 12:00PM	12.00PM- 12.50PM	12:50PM – 1:40PM	1:40PM - 2:30PM	2:30PM - 3:20PM	3:20PM - 4.10PM
MON	AI ML			CS	LUNCH	DAA LAB(DS)		
TUE		CS		DS				
WED		CS		AI ML			DS	
THU		DAA LAB(AI ML)				DS		AI ML
FRI		DS	AI ML					CS
SAT								

LESSON PLAN

Department: **CSE** Academic Year: **2022-23** Year of Branch: **III-I(CSE/EMER)**

Subject Code: **CS3102PC** Subject:**DESIGN&ANALYSISOFALGORITHMS**

Name of Faculty Member: **Dhananjay**

Number of Working Weeks:**21** Number of Periods per Week:**4**

Total Number of Periods: **60**

Lecture schedule with methodology being used/adopted(LESSION PLAN)

S.No	Tentative Date	Topics as per JNTUH Syllabus	Topic Actually Covered	Suggested Book	Method of Teaching
					BB/PPT
	UNIT – I **INTRODUCTION TO ALGORITHMS**				
1	23-08-2022	Algorithms Concept	Algorithms Concept	T1	BB
2	24-08-2022	Pseudo Code	Pseudo Code	T1	BB
3	25-08-2022	Space Complexity, Time Complexity	Space Complexity, Time Complexity	T1	BB
4	26-08-2022	Asymptotic Notation-Big Oh Notation, Omega Notation	Asymptotic Notation-Big Oh Notation, Omega Notation	T1	BB
5	29-08-2022	Asymptotic Notation-Theta Notation And Little Oh Notation	Asymptotic Notation-Theta Notation And Little Oh Notation	T1	BB
6	30-08-2022	Probabilistic Analysis	Probabilistic Analysis	T1	BB
7	01-09-2022	Amortized complexity	Amortized Complexity	T1	BB
8	02-09-2022	Introduction To Divide And Conquer Method	Introduction To Divide And Conquer Method	T1	BB
9	05-09-2022	Binary Search	Binary Search	T1	BB

10	06-09-2022	Quick Sort	Quick Sort	T1	PPT
11	07-09-2022	Strassen's Matrix Multiplication	Strassen's Matrix Multiplication	T1	BB
UNI-II **DISJOINT SETS**					
12	08-09-2022	Disjoint set operations,	Disjoint set operations,	T1	BB
13	09-09-2022	union and find algorithms	union and find algorithms	T1	PPT
14	13-09-2022	Backtracking: General method,	Backtracking: General method,	T1	BB
15	14-09-2022	Backtracking: General method,	Backtracking: General method,	T1	BB
16	15-09-2022	Backtracking: General method,	Backtracking: General method,	T1	PPT
17	16-09-2022	sum of subsets problem,	sum of subsets problem,	T1	BB
18	19-09-2022	sum of subsets problem,	sum of subsets problem,	T1	BB
19	20-09-2022	sum of subsets problem,	sum of subsets problem,	T1	BB
20	20-09-2022	n-queen’s problem,	n-queen’s problem,	T1	BB
21	21-09-2022	n-queen’s problem,	n-queen’s problem,	T1	BB
22	22-09-2022	n-queen’s problem,	n-queen’s problem,	T1	BB
23	23-09-2022	n-queen’s problem,	n-queen’s problem,	T1	BB
24	26-09-2022	sum of subsets problem,	sum of subsets problem,	T1	BB
25	27-09-2022	sum of subsets problem,	sum of subsets problem,	T1	BB
26	27-09-2022	graph coloring	graph coloring	T1	BB
27	28-09-2022	graph coloring	graph coloring	T1	BB

UNIT-III **DYNAMIC PROGRAMMING**					
29	30-09-2022	General method	Applications	T1	BB
30	10-10-2022	General method	Applications	T1	BB
31	11-10-2022	Optimal binary search trees	Optimal binary search trees	T1	BB
32	12-10-2022	Optimal binary search trees	Optimal binary search trees	T1	BB
33	13-10-2022	0/1 knapsack problem	0/1 knapsack problem	T1	BB
34	14-10-2022	0/1 knapsack problem	0/1 knapsack problem	T1	BB
35	17-10-2022	All pairs shortest path problem	All pairs shortest path problem	T1	BB
36	18-10-2022	Traveling sales person problem	Traveling sales person problem	T1	BB
37	19-10-2022	Reliability design	Reliability design	T1	BB
UNIT-IV **GREEDY METHOD**					
38	20-10-2022	General method	Applications	T1	BB
39	21-10-2022	Job sequencing with deadlines	Job sequencing with deadlines	T1	BB
40	25-10-2022	knapsack problem	knapsack problem	T1	BB
41	26-10-2022	knapsack problem	knapsack problem	T1	BB
42	27-10-2022	knapsack problem	knapsack problem	T1	BB
43	28-10-2022	Minimum cost spanning trees	Minimum cost spanning trees	T1	BB
44	03-11-2022	Minimum cost spanning trees	Minimum cost spanning trees	T1	BB
45	04-11-2022	Single source shortest	Single source shortest	T1	BB

		path problem.	path problem.		
46	07-11-2022	Single source shortest path problem.	Single source shortest path problem.	T1	BB
47	09-11-2022	Single source shortest path problem.	Single source shortest path problem.	T1	BB
UNIT-V **BRANCH AND BOUND**					
48	14-11-2022	General method	applications	T1	BB
49	15-11-2022	Travelling sales person problem	Travelling sales person problem	T1	BB
50	27-11-2022	LC Branch and Bound solution	LC Branch and Bound solution	T1	BB
51	30-11-2022	0/1 knapsack problem	0/1 knapsack problem	T1	BB
52	06-12-2022	FIFO Branch and Bound solution.	FIFO Branch and Bound solution.	T1	BB
53	07-12-2022	Basic concepts non deterministic algorithms	Basic concepts non deterministic algorithms	T1	BB
54	09-12-2022	NP - Hard and NP-Complete classes,	NP - Hard and NP-Complete classes,	T1	BB
55	12-12-2022	Cook's theorem.	Cook's theorem.	T1	BB
56	13-12-2022	Cook's theorem.	Cook's theorem.		

UNIT WISE OBJECTIVE TYPE QUESTION BANK,

SHORT AND LONG ANSWER TYPE QUESTIONS

UNIT-1

MULTIPLE CHOICE QUESTIONS

1. Theta notation expresses **[A]**

A) Tight bounds B) Upper bounds C) Lower bounds D) worst cases

2. Consider the flowing functions $f(n)=2^n$ g(n)=n! $h(n)=n^{\log n}$ **[B].**

Which of the following statements about the asymptotic behavior of f(n),g(n) and h(n) is true

A) f(n)=O(g(n)); g(n)=O(h(n)) B) f(n)=Ω(g(n)); g(n)=O(h(n))

C) g(n)=O(f(n)); h(n)=O(f(n)) D) h(n)=O(f(n));g(n)= Ω(f(n))

3. Function g is an upper bound on function f if and only if for all x [B]

A) g(x)<=f(x) B) g(x)>=f(x) C) g(x)=f(x) D) f(x)<g(x)

4. Which of the following is the best sorting algorithm [B]

A) Merge Sort B)Heap Sort C) Bubble Sort D)Quick Sort

5. Which of the following sorting technique have same O(n log n) complexity in all the cases?

A)Insertion Sort B) Quick Sort C) Merge Sort D) Selection Sort **[C]**

6. The running time of quick sort depends heavily on the selection of [D]

A) No of inputs B) Arrangement of elements in array

C) Size of elements D) Pivot element

7. _____is the maximum no of steps that can executed for a given parameter [B]

A) Best case B) Worst case C) Average case D) None

8. The number of a statement is executed is usually is referred as____ [B]

A) Complexity B) Frequency count C) both D) none

9. The Average time complexity of quick sort [B]

A) O (n) B) O (n log n) C) O (n) D) O (log n)

10. The time complexity stresses matrix multiplication [C]

A) 3 B) 1 C) 2 D) none

FILL IN THE BLANKS

1. **An algorithm must be always----------**

[Terminate, Executable, Well-Ordered]

2. **Design of an algorithm is needed----------**

[Pseudo code,Methodology]

3. **Reading the input from the user and printing the output to the user done by----------**

[Primitive operations]

4. **Transferring the values from the user to a variable or vice-verse is-------**

[Function]

5. **The running time of an algorithm is represented by following recurrence relation**:

$T(n)= T(n/3)_n + cn$ n<=3Otherwise ---------- represents the time complexity of the algorithm

[O(n)]

6.$\Sigma_{1<=k<=n}$ O (n) stands for order n is _____

[linear time]

7. Merging4 sorted files containing 50, 10, 25, and 15 records will take_____time.

[650]

8. Time complexity of quick sort in best case_____

[O(n log n)]

9. A_____is the expression of an algorithm in a programming language

[Program]

10._____algorithm do not always yield optimal solution

[greedy]

S.No	Questions	BT	CO	PO
	Part –A(Short Answer Questions)			
1	Define the term algorithm and state the criteria the algorithm	L1	CO1	PO1
2	Define order of an algorithm and the need to analyze the algorithm.	L4	CO1	PO2
3	Define asymptotic notations :big‘ Oh’, omega and theta?	L3	CO1	PO1
4	Distinguish between Algorithm and Pseudocode.	L2	CO2	PO1
5	State the best case and worst case analysis for binary search	L1	CO1	PO2
6	State the best case analysis of quick sort .	L4	CO2	PO2
7	Give the recurrence equation for the worst case behavior of merge sort	L1	CO3	PO2
8	Compute the average case time complexity of quick sort	L1	CO2	PO1
9	How the performance can be analyzed? Explain with the example.	L1	CO1	PO1
10	Describe best case, average case and worst case efficiency of an algorithm?	L2	CO2	PO1

		Part– B(Long Answer Questions)			
11	a)	Discuss various the asymptotic notations used for best case average case and worst case analysis of algorithms.	L4	CO2	PO1, 2
	b)	Define i) Time Complexity ii) Space Complexity	L5	CO2	PO2
12	a)	Discuss binary search algorithm and analyze its time complexity	L6	CO1	PO2, 3
	b)	Explain the algorithm of quick sort with example and find the time complexity	L3	CO2	PO3
13	a)	Explain binary search algorithm	L1	CO3	PO3, 4
	b)	Explain the algorithm of Merge sort with example and find the time complexity.	L2	CO2	PO4
14	a)	Give the algorithm for Stassen’s matrix multiplication and find the time complexity.	L3	CO3	PO5
	b)	Explain the properties / characteristics of an algorithm with an example.	L2	CO1	PO3
15	a)	Write a java program to implement Quick sort algorithm for	L3	CO2	PO2

		sorting a list of integers in ascending order.			
	b)	Sort the list of numbers using merge sort:78,32,42, 62, 98, 12, 34, 83,10	L4	CO1	PO4
16	a)	Discuss binary search algorithm and analyze its time complexity	L1	CO3	PO5
	b)	Discuss various the asymptotic notations used for best case average case and worst case analysis of algorithms	L2	CO4	PO4

UNIT-II

MULTIPLE CHOICE QUESTIONS

1. Time taken to perform (n-1) UNIONS is **[C]**

A) $O(n^2)$ B)$O(n^3)$ C) O(n) D)O(1)

2. How many spanning trees does the following graph have? **[C]**

A) 1 B) 2 C) 3 D) 4

3. Each path through the tree represents a_____in the binary search **[A]**

A) Sequence of comparisons B) Element comparisons

C) Sequence of elements D) Comparisons

4. The edges which are present in the graph but not present in spanning tree are called

A)tree edges B)graph edges C)back edges D)weighted edges [B]

5. Which of the following are disjoint set operations **[B]**

A) Make-set B) Union C)Find-Set D)All

6. The problem of job sequencing with deadlines comes in the class of **[A]**

A) Greedy method B) Divide and Conquer

C)Dynamic programming D) Branch and Bound

7. Which of the following search is not efficient for large sets **[B]**

A)Binary B)Sequential C)Both A&B D)none

8. Average and worst case complexity of binary search **[A]**

A)O(log n) B)O(n) C)$O(2^n)$ D)$O(n^2)$

FILL IN THE BLANKS

1. _____rule is used in FIND algorithm.

[Association]

2. _____Search is used for testing connected components.

[DFS]

3. A graph G is said to be connected iff contains _____

[single connected component]

4. The complexity of binary search algorithm_____

[O(log n)]

5. Binary search only works on_____list

[sorted list]

6. Determining connected components of an undirected graph is an application___set

[disjoint set]

7. Operation Find(i) determines the_____of the tree containing I

[root]

8. Any connected graph with n vertices must have at least_____edges

[n-1]

9. The intersection of two bi-connected components consists of at most one vertex called an____point [Articulation point]

10. Maximal sub graphs of G which are ____

[Clique]

S. No		Questions	BT	CO	PO
Part –A(Short Answer Questions)					
1		Describe union operation on sets	L3	CO1	PO2
2		Describe find operation on sets	L1	CO2	PO3
3		Define a spanning tree and minimal spanning tree	L2	CO3	PO5
4		Define Graph in DAA ?	L3	CO1	PO7
5		Define Tree in DAA ?	L4	CO2	PO1
6		Differentiate Graph and Tree	L5	CO2	PO5
7		What is set? Write different types of set operation?	L3	CO3	PO3
8		Explain different types UNION and FIND algorithm with example?	L1	CO1	PO7
9		What is Disjoint set ? Give an example.	L3	CO1	PO9
10		Define a connected and bi-connected component	L2	CO2	PO4
Part– B(Long Answer Questions)					
11	a)	What is a Backtracking and give the 4 – Queens's solution. Draw the portion of thestate space tree for n = 4 queens using backtracking algorithm	L3	CO1	PO2
	b)	What is a Hamiltonian Cycle? Explain how to find Hamiltonian path and cycle usingbacktracking algorithm.	L2	CO2	PO4
12	a)	Give the statement of sum –of subsets problem. Find all sum of subsets for n=4, (w1, w2, w3, w4) = (11, 13, 24, 7) and M=31.Draw the portion of the state space treeusing fixed – tuple sized approach.	L4	CO1	PO6
	b)	Define: i) State Space tree ii) E – Node iii) Dead Nod	L3	CO2	PO7
13	a)	Define Chromatic number & Give the state space tree for 4 – Coloring problem.	L1	CO1	PO5

	b)	Explain the Graph – coloring problem. And draw the state space tree for m= 3colors n=4 vertices graph. Discuss the time and space complexity.	L2	CO3	PO7
14	a)	Differentiate divide and conquer and greedy method	L2	CO2	PO5, 6
	b)	Write an algorithm for N – queen's problem. Give time and space complexity for8 – queen's problem.	L3	CO1	PO6
15	a)	Distinguish between Dynamic Programming and Greedy method.	L4	CO2	PO7
	b)	What is Graph in DAA? Give an example	L1	CO3	PO8
16	a)	Explain waiting rule for finding UNION of sets and collapsing Rule	L2	CO2	PO4
	b)	Explain with examples find() and Union() algorithms	L3	CO2	PO4

UNIT-III

MULTIPLE CHOICE QUESTIONS

1. Dijkstra's algorithm: **[C]**

A) Has greedy approach to find all shortest paths

B) Has both greedy and dynamic approach to find all shortest paths

C) Has greedy approach to compute single source shortest paths to all other vertices

D) Has both greedy and dynamic approach to compute single source shortest paths to allother vertices

2. Which of the following statements is true about Kruskal's algorithm [B]

A) It is inefficient algorithm, and it never gives the minimum spanning tree.

B) It is an efficient algorithm, and it always gives the minimum spanning tree.

C) It is an efficient algorithm, and it does not always gives the minimum spanning tree.

D) It is inefficient algorithm, and it always gives the minimum spanning tree.

3. If there are n nodes then possible binary search trees are [C]

A) N B) N+1 C)(1/N+1)2NCN D)NC2

4. Which of the following search technique is used in backtracking [D]

A) Linear B)DFS C)BFS D)Binary

5. The time required by prims algorithm is [D]

A)O(n) B) Θ (n) C) Θ (n2) D) O (n2)

6. Kruskals algorithm is used to construct _____ [C]

A) binary tree B)BST C)Minimum spanning tree D)none

7. If there are n nodes the possible binary search trees are_____[B]

A)(n+1)2ncn B)1/(n+1) 2ncn C)(n-1) 2ncn D)1/(n-1) 2ncn

8. Travelling sales person problem is to find a_____[A]

A)tour of minimum cost B)tour of maximum cost C)cost D)none

9. Which method can be used when the solution to a problem can be viewed as the result of a sequence of decisions [A]

A) Greedy method B)Divide & conquer C)Dynamic programming D)none

FILL IN THE BLANKS

1) The number of possible binary trees with 4 nodes_____

[14]

2) In all pairs shortest path problem, the shortest path between____in a graph

[every nodeto every other node in the graph]

3) An optimal code for a file is always represented by a_____in which every non leaf nodehas two children

[full binary tree]

4). A tree which includes all vertices in a graph G is called

[spanning tree]

5) _____Algorithm technique is used in the implementation of Kruskal solution for theMST.

[Greedy technique]

6) _____Property satisfies the optional substructure property then a locally optimalsolution is globally optimal.

[greedy choice property]

7.) The Knapsack problem belongs to the domain_____problem

[optimization]

8.) Dynamic programming belongs to_____class

[recursion]

9) _____ is the time complexity for travelling sales man problem solved by dynamic programming

[O(n2 2n)]

10) _____is a BFS like state space tree search in which the list of live nodes is a queue

[FIFO search]

S. No		Questions	BT	CO	PO
Part –A(Short Answer Questions)					
1		Define greedy method	L2	CO1	PO2
2		Define job sequencing with deadlines problem	L3	CO2	PO4
3		Define minimum cost spanning tree	L2	CO3	PO5
4		Define Knapsack problem?	L3	CO3	PO6, 7
5		Define Prim's algorithm	L2	CO1	PO8
6		Define Kruskal's algorithm	L1	CO2	PO9
7		Define single source shortest path problem	L3	CO4	PO11
8		Define dynamic programming	L1	CO5	PO10
9		List the features of dynamic programming	L2	CO3	PO8
10		Distinguish greedy method and dynamic programming	L1	CO2	PO9
Part– B(Long Answer Questions)					
11	a)	What is a principle of optimality? Explain how travelling sales person problem uses the dynamic programming technique with example and also find space and time complexity.	L3	CO3	PO10
	b)	Explain single source shortest path problem with example	L1	CO1	PO1
12	a)	Give the statement of Reliability design problem and explain with suitable example.	L2	CO2	PO1
	b)	Explain prims algorithm with example	L3	CO3	PO9
13	a)	Explain Kruskal's algorithm with example	L1	CO1	PO8
	b)	What is Reliability design with example	L2	CO3	PO5
14	a)	Explain optimal binary search tree algorithm with example	L3	CO4	PO9
	b)	Explain 0/1 knapsack problem with example	L1	CO3	PO3
15	a)	What is All – Pair Shortest Path problem (APSP)? Discuss the Floyd's APSP algorithm and discuss the analysis of this algorithm.	L2	CO1	PO6
	b)	Describe the travelling sales man problem and discuss how to solve it using dynamic programming?	L4	CO2	PO9
16	a)	Explain Kruskal's algorithm with example	L1	CO3	PO5
	b)	Describe the Dynamic 0/1 Knapsack Problem. Find an optimal solution for the dynamic programming 0/1 knapsack instance for n=3, m=6, profits are (p1, p2, p3) = (1,2,5), weights are (w1,w2,w3)=(2,3,4).	L1	CO1	PO3

UNIT-IV

MULTIPLE CHOICE QUESTIONS

1. Which of the following can search for an answer node by using an intelligent ranking function [C]

A) LIFO B)FIFO C) LC D)Binary

2. What is the time complexity of 0/1 Knapsack problem [A]

A) O(n) B)$O(n^2)$ C) $O(2^{n/2})$ D) $O(n^3)$

3. Which of the following technique is used to solve graph coloring problem [C]

A) Divide and Conquer B) Dynamic programming

C) Backtracking D) Branch and Bound

4. Which of the following is a generated node that cannot be expanded further? [B]

A) E-node B) Dead node C) Live Node D) Null node

5. Which of the following divides a set S of candidates into two or more sets such that their union covers S [C]

A) Branching B) Bounding C) Backtracking D) Approximation

6. If the lower bound for some tree node qi is greater than the upper bound for some other node qk, then qi and all its decedents may be discarded is referred as [C]

A) Pruning for minimization B) pruning for maximization

C) Pruning for generalization D)pruning for conceptualization

7. Branch and bound technique is applicable for only [A]

A) Minimization problem B) Maximization problem

C) Reliability Design problem D) BFS Algorithm

8. If there is m live node then addition and deletion of a node to a list can be carried out in the time of [A]

A)O(log m) B)O(1) C) Θ (1) D)O(m)

9. We start at a particular node in a graph visiting all nodes exactly once and come back to initial node with minimum cost. This is known as ____ [D]

A)0/1 knapsack problem B) Optimal storage on tapes

C) Minimum cost spanning trees D) Travelling sales person problem

FILL IN THE BLANKS

1)_____rules are used to solve 0/1 knapsack

[branch and bound]

2). The travelling sales person problem which asked for a tour that has minimum cost.

Then this tour is_____

[Hamiltonian cycle]

3). For N queens problem the time complexity_____

[O(log n)]

4). The solution space tree of 8 queens contains _____

[8th to the power of 8 tuples]

5). The_____time complexity of 8-queens problems

[$O(n^2)$]

6). In backtracking the function which needs to be maximized or minimized for a given problem is known as _____

[criterion function]

7) Sum of subsets is an example of_____problem

[backtracking]

8). Using backtracking for 4-queens problem tee organization contains_____no of nodes

[31]

9). _____is the time complexity of sum of subsets problem

[$O(n2^n)$]

10). In_____technique all the children of the E-node are generated before any other live node can become the E-node

[branch and bound]

11). State generation methods in which in the E-node remains the E-node until it is dead is called_____

[branch and bound]

S. No		Questions	BT	CO	PO
Part –A(Short Answer Questions)					
1		Define i) Feasible solution ii) Optimal solution.	L1	CO2	PO2
2		Define Greedy Method?	L2	CO3	PO4
3		What is spanning tree ? give example	L3	CO2	PO7
4		What is job sequence with dead line?	L1	CO1	PO5
5		What is minimum spanning tree?	L4	CO2	PO5
6		What is single source shortest path ?	L4	CO1	PO3
7		What is time complexity of job sequence with dead line?	L4	CO3	PO8
8		What is time complexity of spanning tree?	L4	CO1	PO6
9		What is time complexity o single source shortest path ?	L1	CO2	PO3
10		Distinguish between Prim's and Kruskal's spanning tree algorithm.	L1	CO3	PO9
Part– B(Long Answer Questions)					
11	a)	Find an optimal solution to the knapsack instance n=7 objects and the capacityof knapsack m=15. The profits and weights of the objects are (P1,P2,P3, P4, P5, P6, P7)= (10, 5,15,7,6,18,3) (W1,W2,W3,W4,W5,W6,W7)= (2,3,5,7,1,4,1).	L1	CO1	PO3
	b)	State the Job – Sequencing Deadline Problem	L1	CO2	PO5
12	a)	Discuss the single – source shortest paths (i.e. Dijkstra's) algorithm with suitableexample and also find the time complexity.	L2	CO2	PO7
	b)	What is a Spanning tree? Explain Prim's Minimum cost spanning tree algorithm withsuitable example and also find the time complexity.	L3	CO1	PO9
13	a)	Find an optimal sequence to the n=5 Jobs where profits (P1,P2,P3,P4,P5) =(20,15,10,5,1) and deadlines (d1,d2,d3,d4,d5) =(2,2,1,3,3).	L1	CO2	PO11

	b)	What is a Minimum Cost Spanning tree? Explain Kruskal's Minimum costspanning tree algorithm with suitable example and also find the time complexity	L4	CO3	PO5, 7
14	a)	State the Greedy Knapsack? Write the algorithm for Greedy knapsack and alsocompute the time complexity	L1	CO1	PO4
	b)	Write an algorithm for job sequence with dead lines.	L1	CO2	PO7
15	a)	Write an algorithm for Kruskal's algorithm.	L1	CO2	PO1
	b)	Write an algorithm for Prim's algorithm.	L1	CO1	PO3
16	a)	Write an algorithm for Dijkstra's algorithm.	L3	CO2	PO5
	b)	Write Application of Greedy Method.	L3	CO3	PO3

UNIT-V

MULTIPLE CHOICE QUESTIONS

1. Let S be an NP complete problem and Q and R be two other not known to be NP. Q is polynomial time reducible to S and S is polynomial time reducible to R. [B]

Which one of thefollowing statements is true?

A) R is NP-complete B)R is NP-hard C)Q is NP-complete D)Q is NP-hard

2. Language accepted by a deterministic Turing machine in polynomial time belongs to [B]

A)P B) NP C) NC D) NH

3. Problem that is_____has property that it can be solved in polynomial time if and only if all other NP- Hard problems are solved in polynomial time [B]

A) NP-Hard B)NP-complete C)P D)NP

4. Any problem for which answer is either 0 or 1 is called_____[decision]

4. If an NP complete problem is solvable in polynomial time___ [A]

A) P=NP B)P!=NP C)P D)NP

5. The halting problem is to determine for an arbitrary_____ [D]

A) NP-complete B)Non deterministic C)P complete D)deterministic

6. Two problems L1 and L2 are said to be polynomial equivalent iff___ [A]

A) L1=L2 & L2=L1 B)L1!=L2 & L2!=L1 C)both C)none

7. The computing times for choice,failure and success are taken to be [D]

A)O(1) B)O(n) C) Θ (1) D) Θ (n)

8. Any problem for which answer is either zero or one is called [A]

A) Decision D) optimization C) Clique D) none

9. Assuming P != NP, which of the following is true ? [B]

(A) NP-complete = NP (B) NP-complete P

(C) NP-hard = NP (D) P = NP-complete

FILL IN THE BLANKS

1) **The problem 3-SAT and 2-SAT are**

[NP-complete and in P respectively]

2) **The_____states that whatever the initial state and decision are, the remaining decision must constitute an optimal decision sequence.**

[Principal of optimality**]**

3) **If there are m live nodes then to test whether a list is empty can be carried out is_____time**

[Θ (1)]

4) **If an NP hard problem can be solved in polynomial time then all_____problems can be solved in polynomial time.**

[NP-complete]

5) **The use of function reduce does not decrease the computing time as much expected by the reduction in the number of _____**

[objects]

6) **_____problems require satisfying explicit and implicit constraints**

[backtracking]

7) **_____is a set of all decision problems solvable by non-deterministic algorithms in polynomial time**

[NP]

8) **RSA code breaking by factoring belongs to_____class**

[NP-complete]

9) **NP stands for _____**

[non deterministic polynomial time]

10) **A problem L is NP complete iff _____**

[Any given solution for NP-complete problem scan be verified quickly, but there is no efficient known solution]

S. No		Questions	BT	CO	PO
Part –A(Short Answer Questions)					
1		Define class P?	L4	CO1	PO4
2		Compare NP-hard and NP-completeness	L4	CO2	PO6
3		Define NP-hard problem	L4	CO3	PO7
4		Define NP-complete problem	L4	CO1	PO7
5		Define deterministic problem?	L4	CO2	PO2
6		Define non-deterministic problem	L4	CO1	PO1
7		Define i) LC – Search ii) Branch and Bound (BB) iii) FIFO – BB.	L4	CO3	PO5
8		Explain optimization problem	L1	CO2	PO9
9		Define Bounding Function?	L1	CO3	PO2
10		Define Cook's theorem?	L1	CO1	PO3
Part– B(Long Answer Questions)					
11	a)	Draw the portion of state space tree generated by FIFOBB for the job sequencing with deadlines instance n=5 , (p1,p2,..,p5) =(6,3,4,8,5), (t1,t2,..t5) = (2,1,2,1,1) and (d1,d2,..,d5)=(3,1,4,2,4).What is the penalty corresponding to an optimal solution	L2	CO2	PO8
	b)	Explain deterministic and non-deterministic algorithms	L1	CO3	PO9
12	a)	Write non deterministic algorithm for sorting and searching	L4	CO1	PO6
	b)	Write anon-deterministic knapsack algorithm	L1	CO3	PO10
13	a)	Explain P and NP problems are related	L1	CO3	PO4
	b)	Distinguish NP- hard and NP-complete problems	L3	CO4	PO2
14	a)	Define Bounding Function? Give the statement of 0/1 Knapsack FIFO BB and explain the procedure with the knapsack instance for n=4.m=15,(p1,p2,p3,p4)=(10,10,12,18) (w1,w2,w3,w4) =(2, 4, 6, 9).	L1	CO2	PO9
	b)	Distinguish between backtracking and branch – and bound techniques.	L1	CO3	PO1
15	a)	Explain the strategy to prove that a problem is NP-hard	L1	CO1	PO2
	b)	Explain travelling sales person person problem LCBB procedure with the followinginstance and draw the portion of the state space tree and find an optimal tour. $\begin{pmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{pmatrix}$	L1	CO3	PO9
16	a)	State and prove cook's theorem	L2	CO1	PO4
	b)	Draw the portion of state space tree generated by LCBB for the 0/1 Knapsack instance: n = 5, (p1,p2,…,p5) = (10,15,6,8,4), (w1,w2,..,w5) = (4,6,3,4,2) and m=12. Find an optimal solution using fixed – tuple sized approach.	L4	CO2	PO6

UNIT I:

Introduction: Algorithm, Psuedo code for expressing algorithms, Performance Analysis-Space complexity, Time complexity, Asymptotic Notation- Big oh notation, Omega notation,Theta notation and Little oh notation, Probabilistic analysis, Amortized analysis.

Divide and conquer: General method, applications-Binary search, Quick sort, Merge sort,Strassen's matrix multiplication.

INTRODUCTION TO ALGORITHM

History of Algorithm

- The word algorithm comes from the name of a Persian author, Abu Ja'far Mohammed ibn Musa al Khowarizmi (c. 825 A.D.), who wrote a textbook on mathematics.
- He is credited with providing the step-by-step rules for adding, subtracting, multiplying,and dividing ordinary decimal numbers.
- When written in Latin, the name became Algorismus, from which algorithm is but a smallstep
- This word has taken on a special significance in computer science, where "algorithm" hascome to refer to a method that can be used by a computer for the solution of a problem
- Between 400 and 300 B.C., the great Greek mathematician Euclid invented an algorithm
- Finding the greatest common divisor (gcd) of two positive integers.
- The gcd of X and Y is the largest integer that exactly divides both X and Y .
- Eg.,the gcd of 80 and 32 is 16.
- The Euclidian algorithm, as it is called, is considered to be the first non-trivial algorithmever devised.

What is an Algorithm?

Algorithm is a set of steps to complete a task.

For example,

Task: to make a cup of tea.

Algorithm:

· add water and milk to the kettle,

· boil it, add tea leaves,

· Add sugar, and then serve it in cup.

''a set of steps to accomplish or complete a task that is described precisely enough thata computer can run it''.

Described precisely: very difficult for a machine to know how much water, milk to beadded etc. in the above tea making algorithm.

These algorithms run on computers or computational devices..For example, GPS in oursmartphones, Google hangouts.

GPS uses shortest path algorithm.. **Online shopping** uses cryptography which uses RSAalgorithm.

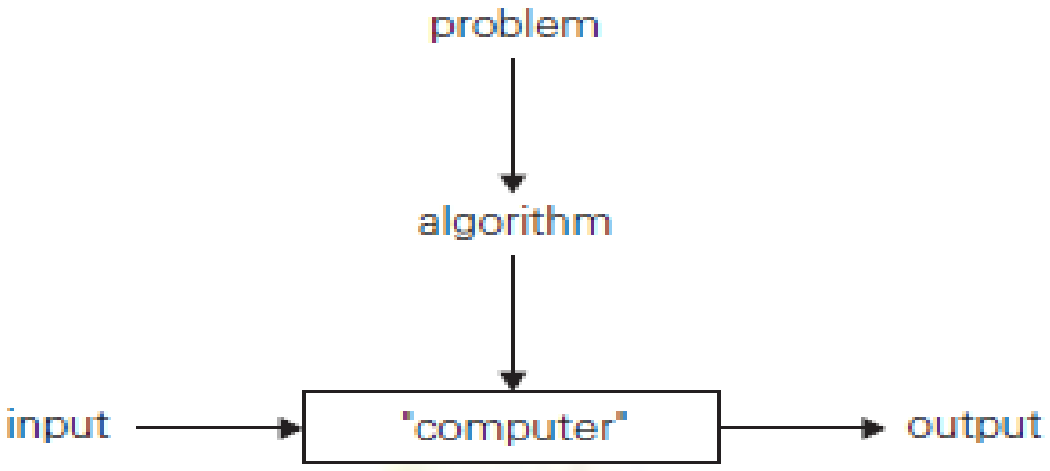
- Algorithm Definition1:

- An algorithm is a finite set of instructions that, if followed, accomplishes a particular task.In addition, all algorithms must satisfy the following criteria:
- Input. Zero or more quantities are externally supplied.
- Output. At least one quantity is produced.
- Definiteness. Each instruction is clear and unambiguous.
- Finiteness. The algorithm terminates after a finite number of steps.
- Effectiveness. Every instruction must be very basic enough and must befeasible.

- Algorithm Definition2:

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

- Algorithms that are definite and effective are also called *computational procedures*.
- A program is the expression of an algorithm in a programming language



- **Algorithms for Problem Solving**

The main steps for Problem Solving are:

1. Problem definition
2. Algorithm design / Algorithm specification
3. Algorithm analysis
4. Implementation
5. Testing

- [Maintenance]Step1. Problem Definition

What is the task to be accomplished?

Ex: Calculate the average of the grades for a given student

- Step2.Algorithm Design / Specifications:

Describe: in natural language / pseudo-code / diagrams / etc

- Step3. Algorithm analysis

Space complexity - How much space is required

Time complexity - How much time does it take to run the algorithmComputer Algorithm

An algorithm is a procedure (a finite set of well-defined instructions) for accomplishingsome tasks which, given an initial state terminate in a defined end-state

The computational complexity and efficient implementation of the algorithm are importantin computing, and this depends on suitable data structures.

- Steps 4,5,6: Implementation, Testing, Maintainance
- Implementation:

Decide on the programming language to use C, C++, Lisp, Java, Perl, Prolog, assembly, etc.

, etc.

Write clean, well documented code

- Test, test, test

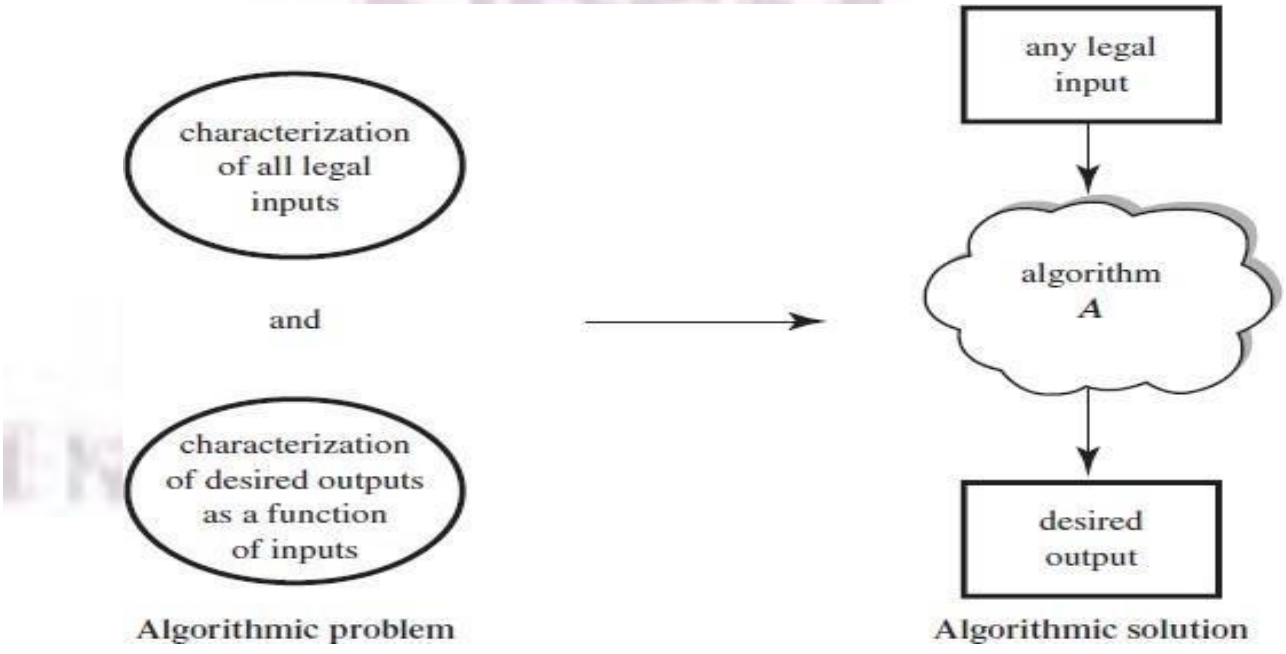
Integrate feedback from users, fix bugs, ensure compatibility across different versions

- Mai

ntenance. Release

Updates,fix bugs

Keeping illegal inputs separate is the responsibility of the algorithmic problem, while treating special classes of unusual or undesirable inputs is the responsibility of the algorithmitself.



- **4 Distinct areas of study of algorithms:**

- How to devise algorithms. Techniques – Divide & Conquer, Branch and Bound , Dynamic Programming
- How to validate algorithms.
- Check for Algorithm that it computes the correct answer for all possible legal inputs. algorithm validation. First Phase
- Second phase Algorithm to Program Program Proving or Program Verification Solution be stated in two forms:
- First Form: Program which is annotated by a set of assertions about the input and output variables of the program predicate calculus
- Second form: is called a specification
- 4 Distinct areas of study of algorithms (..Contd)
- How to analyze algorithms.
- Analysis of Algorithms or performance analysis refer to the task of determining how much computing time & storage an algorithm requires
- How to test a program 2 phases
- Debugging - Debugging is the process of executing programs on sample data sets to determine whether faulty results occur and, if so, to correct them.
- Profiling or performance measurement is the process of executing a correct program on data sets and measuring the time and space it takes to compute the results

PSEUDOCODE:

- Algorithm can be represented in Text mode and Graphic mode
- Graphical representation is called Flowchart
- Text mode most often represented in close to any High level language such as C, Pascal Pseudocode

- **Pseudocode: High-level description of an algorithm.**

- More structured than plain English.
- Less detailed than a program.
- Preferred notation for describing algorithms.
- Hides program design issues.

- **Example of Pseudocode**:

- To find the max element of an array

Algorithm *arrayMax*(*A*, *n*) Input
array *A* of *n* integers Output
maximum element of *AcurrentMax*
← *A*[0]

for *i* ← 1 to *n* – 1 do
if *A*[*i*] > *currentMax* then

currentMax ← *A*[*i*]

return *currentMax*

- Control flow
- if … then … [else …]
- while … do …
- repeat … until …
- for … do …
- Indentation replaces braces
- Method declaration
- Algorithm *method* (*arg* [, *arg*…])
- Input …
- Output …
- Method call
- *var.method* (*arg* [, *arg*…])
- Return value
- return *expression*
- Expressions
- Assignment (equivalent to □)
- Equality testing (equivalent to □□)
- n^2 Superscripts and other mathematical formatting allowed

PERFORMANCE ANALYSIS:

- What are the Criteria for judging algorithms that have a more direct relationship to performance?
- computing time and storage requirements.
- **Performance evaluation** can be loosely divided into two major phases:
- a priori estimates and
- a posteriori testing.
- □refer as *performance analysis* and *performance measurement* respectively

- The space complexity of an algorithm is the amount of memory it needs to run to

completion.

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

Space Complexity:

- Space Complexity Example:
- Algorithm abc(a,b,c)

{

return a+b++*c+(a+b-c)/(a+b) +4.0;

}

- The Space needed by each of these algorithms is seen to be the sum of the following component.

1.A fixed part that is independent of the characteristics (eg:number,size)of the inputs and outputs.

The part typically includes the instruction space (ie. Space for the code), space for simple variable and fixed-size component variables (also called aggregate) space for constants, andso on.

2. A variable part that consists of the space needed by component variables whose size is dependent on the particular problem instance being solved, the space needed by referenced variables (to the extent that is depends on instance characteristics), and the recursion stack space.

The space requirement s(p) of any algorithm p may therefore be written as,S(P)
= c+ Sp(Instance characteristics)
Where ‘c’ is a constant.

Example 2:

Algorithm sum(a,n)

{ s=0.0;

for I=1 to n do s=

```
s+a[I]; return s;
}
```

- The problem instances for this algorithm are characterized by n,the number of elements to be summed. The space needed d by 'n' is one word, since it is of type integer.
- The space needed by 'a'a is the space needed by variables of tyepe array of floating point numbers.
- This is atleast 'n' words, since 'a' must be large enough to hold the 'n' elements to be summed.
- So,we obtain Ssum(n)>=(n+s)
- [n for a[],one each for n,I a& s]

Time Complexity:

- The time T(p) taken by a program P is the sum of the compile time and the run time(execution time)

- The compile time does not depend on the instance characteristics. Also we may assume that a compiled program will be run several times without recompilation .This rum time is denoted by tp(instance characteristics).

- The number of steps any problem statement is assigned depends on the kind of statement.

- For example, comments à 0 steps.

Assignment statements is 1 steps.

[Which does not involve any calls to other algorithms]

Interactive statement such as for, while & repeat-until à Control part of the statement.

We introduce a variable, count into the program statement to increment count with initial value 0.Statement to increment count by the appropriate amount are introduced into the program.

This is done so that each time a statement in the original program is executescount is incremented by the step count of that statement.

Algorithm:

```
Algorithm sum(a,n)
{
s= 0.0;
count = count+1; for
I=1 to n do
{
count        =count+1;
s=s+a[I];
count=count+1;
}
count=count+1;
count=count+1;
return s;
}
```

- If the count is zero to start with, then it will be 2n+3 on termination. So each invocation of sum execute a total of 2n+3 steps.

2. The second method to determine the step count of an algorithm is to build atable in which we list the total number of steps contributes by each statement.

- First determine the number of steps per execution (s/e) of the statement and the total number of times (ie., frequency) each statement is executed.

- By combining these two quantities, the total contribution of all statements, thestep count for the entire algorithm is obtained.

Statement	*Steps per execution*	*Frequency*	*Total*
1. Algorithm Sum(a,n)2.{ 3. S=0.0; 4. for I=1 to n do5. s=s+a[I]; 6. return s;7. }	0 0 1 1 1 1 0	- -1 n+1n 1 -	0 0 1 n+1 n 1 0
Total			2n+3

How to analyse an Algorithm?

Let us form an algorithm for Insertion sort (which sort a sequence of numbers).The pseudocode for the algorithm is give below.

Pseudo code for insertion Algorithm:

Identify each line of the pseudo code with symbols such as C1, C2 ..

Pseudo code for Insertion Algorithm	**Line Identification**
for j=2 to A length	C1
key=A[j]	C2
//Insert A[j] into sorted Array A[1...j-1]	C3
i=j-1	C4
while i>0 & A[j]>key	C5
A[i+1]=A[i]	C6
i=i-1	C7
A[i+1]=key	C8

Let Ci be the cost of ith line. Since comment lines will not incur any cost C3=0.

Cost	**No. Of times Executed**
C1	N
C2	n-1
C3=0	n-1
C4	n-1
C5	$n-1$ $\sum t_j$ j=2
C6	n $\sum t_j - 1$ j=2
C7	n $\sum t_j - 1$ j=2
C8	n-1

Running time of the algorithm is:

T(n)=C1n+C2(n-1)+0(n-1)+C4(n-1)+C5($\sum_{j=2}^{n-1} t_j$)+C6($\sum_{j=2}^{n} t_j - 1$)+C7($\sum_{j=2}^{n} t_j - 1$)+C8(n-1)

Best case:

It occurs when Array is sorted.All t_j values are 1.

$T(n)=C1n+C2(n-1)+0\ (n-1)+C4(n-1)+C5(\sum_{j=2}^{n-1} 1)+C6(\sum_{j=2}^{n} 1-1)+C7(\sum_{}^{n} 1-1)+$
$C8(n-1)$

$=C1n+C2\ (n-1)+0\ (n-1)+C4\ (n-1)+C5+C8\ (n-1)$

$=(C1+C2+C4+C5+C8)\ n-(C2+C4+C5+C8)$

· Which is of the form an+b.

· Linear function of n.

So, linear growth.

Worst case:

It occurs when Array is reverse sorted, and tj =j

$T(n)=C1n+C2(n-1)+0\ (n-1)+C4(n-1)+C5(\sum_{j=2}^{n-1} j)+C6(\sum_{j=2}^{n} j-1)+C7(\sum_{j=2}^{n} j-1)+$
$C8(n-1)$

$=C1n+C2(n-1)+C4(n-1)+C5(\frac{n(n-1)}{2}-1)+C6(\sum_{j=2}^{n} (n-1))+C7(\sum_{j=2}^{n} (n-1))+C8(n-1)$

which is of the form an^2+bn+c

Quadratic function. So in worst case insertion set grows in n2.Why $j=2$ we concentrate on worst-case running time?

· The worst-case running time gives a guaranteed upper bound on the running time for any input.

· For some algorithms, the worst case occurs often. For example, when searching, the worst case often occurs when the item being searched for is not present, and searches for absent items may be frequent.

· Why not analyze the average case? Because it's often about as bad as the worst case.

Order of growth:

It is described by the highest degree term of the formula for running time. (Drop lower-order terms. Ignore the constant coefficient in the leading term.)

Example: We found out that for insertion sort the worst-case running time is of the forman^2 + bn + c.

Drop lower-order terms. What remains is an2.Ignore constant coefficient. It results in n^2.But we cannot say that the worst-case running time T(n) equals n^2 .Rather It grows like n^2 . But itdoesn't equal n^2.We say that the running time is Θ (n^2) to capture the notion that the order ofgrowth is n^2. We usually consider one algorithm to be more efficient than another if its worst-case running time has a smaller order of growth.

Complexity of Algorithms

The complexity of an algorithm M is the function f(n) which gives the running time and/or storage space requirement of the algorithm in terms of the size 'n' of the input data. Mostly, the storage space required by an algorithm is simply a multiple of the data size 'n'.

Complexity shall refer to the running time of the algorithm.

The function f(n), gives the running time of an algorithm, depends not only on the size 'n' of the

input data but also on the particular data. The complexity function f(n) for certain cases are:

1. **Best Case** : The minimum possible value of f(n) is called the best case.

2. **Average Case** : The expected value of f(n).

3. **Worst Case** : The maximum value of f(n) for any key possible input.

ASYMPTOTIC NOTATION

 Formal way notation to speak about functions and classify them

The following notations are commonly use notations in performance analysis and used to characterize the complexity of an algorithm:

1. Big–OH (O) ,
2. Big–OMEGA (Ω),
3. Big–THETA (Θ) and
4. Little–OH (o)

Asymptotic Analysis of Algorithms:

Our approach is based on the *asymptotic complexity* measure. This means that we don't try to count the exact number of steps of a program, but how that number grows with the size of theinput to the program. That gives us a measure that will work for different operating systems, compilers and CPUs. The asymptotic complexity is written using big-O notation.

- It is a way to describe the characteristics of a function in the limit.
- It describes the rate of growth of functions.
- Focus on what's important by abstracting away low-order terms and constant factors.

Time complexity	Name	Example
O(1)	Constant	Adding an element to the front of a linked list
O(logn)	Logarithmic	Finding an element in a sorted array
O (n)	Linear	Finding an element in an unsorted array
O(nlog n)	Linear	Logarithmic Sorting n items by 'divide-and-conquer'-Mergesort
$O(n^2)$	Quadratic	Shortest path between two nodes in a graph
$O(n^3)$	Cubic	Matrix Multiplication
$O(2^n)$	Exponential	The Towers of Hanoi problem

Big 'oh': the function f(n)=O(g(n)) iff there exist positive constants c and no such that f(n)<=c*g(n) for all n, n>= no.

Omega: the function f(n)=(g(n)) iff there exist positive constants c and no such thatf(n) >= c*g(n) for all n, n >= no.

Theta: the function f(n)=(g(n)) iff there exist positive constants c1,c2 and no such that c1g(n) <= f(n) <= c2 g(n) for all n, n >= no

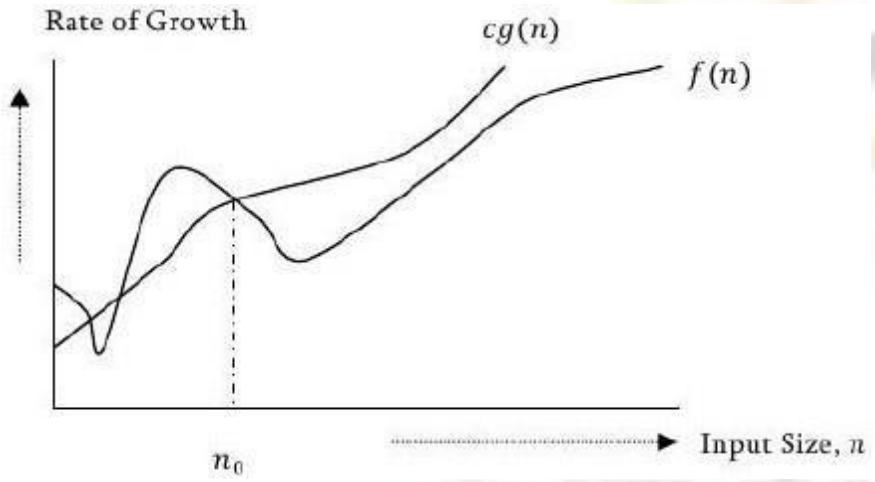
Big-O Notation

This notation gives the tight upper bound of the given function. Generally we represent it as f(n) = O(g (11)). That means, at larger values of n, the upper bound off(n) is g(n). For example, if f(n) = $n^4 + 100n^2 + 10n + 50$ is the given algorithm, then n^4 is g(n). That means g(n) gives the maximum rate of growth for f(n) at larger values of n.

O —notation

defined as O(g(n)) = {f(n): there exist positive constants c and n_o such that0 <= f(n) <= cg(n) for all n >= n_o}. g(n) is an asymptotic tight upper bound for f(n). Our objective is to give some rate of growth g(n) which is greater than given algorithms rate of growth f(n).

In general, we do not consider lower values of n. That means the rate of growth at lower values of n is not important. In the below figure, n_o is the point from which we consider the rate of growths for a given algorithm. Below n_o the rate of growths may be different.



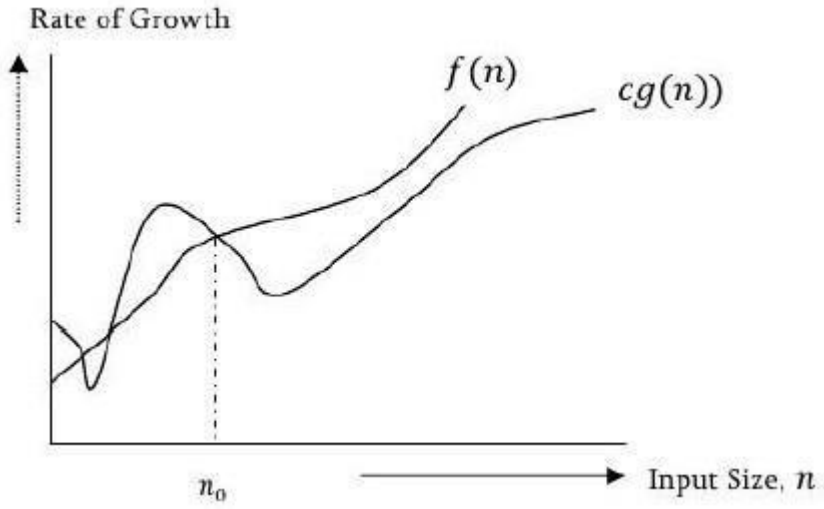
Note Analyze the algorithms at larger values of n only What this means is, below no we donot care for rates of growth.

Omega— Ω notation

Similar to above discussion, this notation gives the tighter lower bound of the given algorithm and we represent it as f(n) = Ω (g(n)). That means, at larger values of n, the tighter lower bound of f(n) is g

For example, if f(n) = $100n^2 + 10n + 50$, g(n) is Ω (n^2).

The . Ω. notation as be defined as Ω (g (n)) = {f(n): there exist positive constants c and n_o such that 0 <= cg (n) <= f(n) for all n >= n_o}. g(n) is an asymptotic lower bound for f(n). Ω



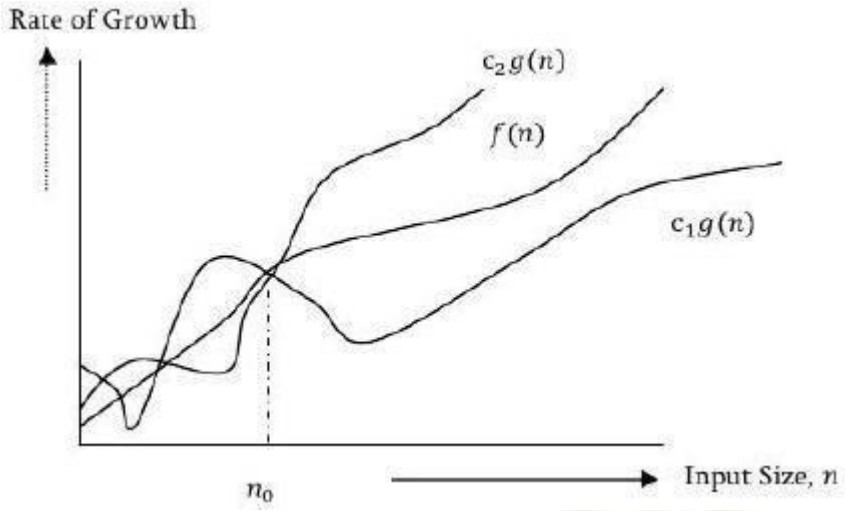
(g (n)) is the set of functions with smaller or same order of growth as f(n).

Theta- Θ notation

This notation decides whether the upper and lower bounds of a given function are same ornot. The average running time of algorithm is always between lower bound and upper bound.

If the upper bound (O) and lower bound (Ω) gives the same result then Θ notation will also have the same rate of growth. As an example, let us assume that f(n) = 10n + n is the expression. Then, its tight upper bound g(n) is O(n). The rate of growth in best case is g (n) = 0(n). In this case, rate of growths in best case and worst are same. As a result, the average case will also be same.

None: For a given function (algorithm), if the rate of growths (bounds) for O and Ω are notsame then the rate of growth Θ case may not be same.



Now consider the definition of Θ notation It is defined as Θ (g(n)) = {f(71): there exist positive constants C1, C2 and no such that O<=5 c_1g(n) <= f(n) <= c_2g(n) for all n >= n_o}. g(n) is an asymptotic tight bound for f(n). Θ (g(n)) is the set of functions with the same order of growth as g(n).

Important Notes

For analysis (best case, worst case and average) we try to give upper bound (O) and lower bound (Ω) and average running time (Θ). From the above examples, it should also be clearthat, for a given function (algorithm) getting upper bound (O) and lower bound (Ω) and average running time (Θ) may not be possible always.

For example, if we are discussing the best case of an algorithm, then we try to give upper bound (O) and lower bound (Ω) and average running time (Θ).

In the remaining chapters we generally concentrate on upper bound (O) because knowing lower bound (Ω) of an algorithm is of no practical importance and we use 9 notation if upperbound (O) and lower bound (Ω) are same.

Little Oh Notation

The little Oh is denoted as o. It is defined as : Let, f(n} and g(n} be the non negative functions then

Program for binary search (recursive)	Algorithm for binary search (recursive)
int binary_search(int A[], int key, int imin, int imax)	**Algorithm binary_search(A, key, imin, imax)**
{ **if (imax < imin)** **return array is empty; if(key<imin || K>imax)** **return element not in array listelse** **{** **int imid = (imin +imax)/2; if (A[imid] > key)** **return binary_search(A, key, imin, imid-1);else if (A[imid] < key)** **return binary_search(A, key, imid+1, imax);else** **return imid;** **}** **}**	**{** **if (imax < imin)** **then return "array is empty";** **if(key<imin || K>imax)** **then return "element not in array list"else** **{** **imid = (imin +imax)/2; if (A[imid] > key)** **then** **return binary_search(A, key, imin, imid-1);else if (A[imid] < key) then** **return binary_search(A, key, imid+1, imax);else** **return imid;** **}** **}**

lim

$$\frac{f(n)}{g(n)} = 0$$

such that f(n}= o(g{n)} i.e f of n is little Oh of g of n.

f(n) = o(g(n)) if and only if f'(n) = o(g(n)) and f(n) != Θ {g(n))

Applications of Divide and conquer rule or algorithm:

- Binary search,
- Quick sort,
- Merge sort,
- Strassen's matrix multiplication.

Binary search or Half-interval search algorithm:

1. This algorithm finds the position of a specified input value (the search "key") withinan array sorted by key value.
2. In each step, the algorithm compares the search key value with the key value of the middle element of the array.
3. If the keys match, then a matching element has been found and its index, or position, is returned.
4. Otherwise, if the search key is less than the middle element's key, then the algorithm repeats its action on the sub-array to the **left** of the middle element or, if the search key is greater, then the algorithm repeats on sub array to the **right** of the middle element.
5. If the search element is

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 alreadybeen 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 subproblems 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 treeis 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 Vtogether 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 f rom ‘ 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 intern al 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 lev el L. Hence the expected cost contribution from the internal node for ai is p (i) * level (ai). 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 binarysearch tree.

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

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 capacit y remaining in knapsack is M and no profit has accrued or the capacity re maining is M- Wn and a profit of Pn 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 □ 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□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 $(1\text{-}ri)^{mi}$. 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-nodeare 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 livenode.

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 havecost 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 $x_i \geq 0$

S_i ={all non negative real numbers}

x_i =0 or 1 or S_i ={0,1}

$l_i \leq x_i \leq u_i$ or S_i = {a: $l_i \leq a \leq u_i$ }

The explicit constraints depend on the particular instance I of the problem being solved. All tuples thatsatisfy 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

$(X_1, \ldots\ldots, X8)$. The explicit constraints using this formulation s are $S_i=\{1,2,3,4,5,6,7,8\}$, $1 \leq i \leq n$. 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).

			Q				
					Q		
							Q
	Q						
	2	3	4	5	6	Q 7	8
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

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

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 arezero.

The second representation calls for storing only each non-zero coefficient and its correspondingexponent; 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.

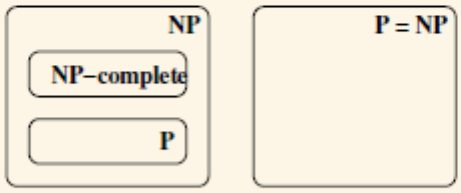
33.2.2. NP-Complete Problems

- 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

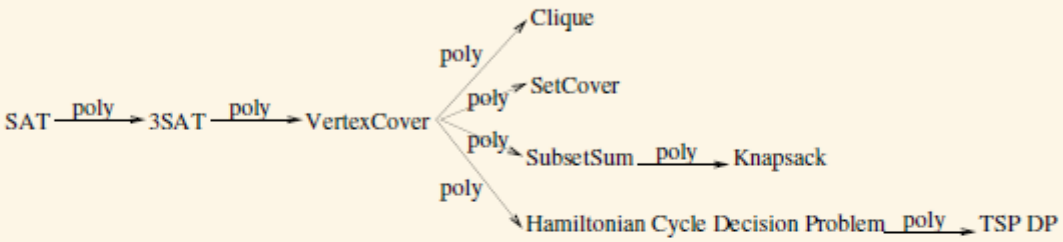
  and so **P** = **NP**.
- If anyone ever shows that an **NP**-complete problem is intractable, then
  - every **NP**-complete problem is also intractable

  and, of course, **P** ≠ **NP**.
- So there are two possibilities:

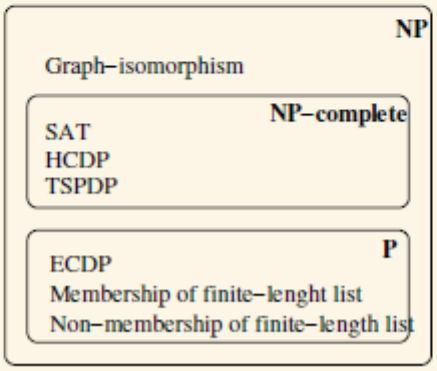


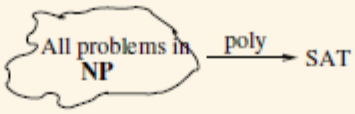
  We don't know which of these is the case.
- But this gives Computer Scientists a clear line of attack. It makes sense to focus efforts on the **NP**-complete problems: they all stand or fall together.
- So these sound like very significant problems in our theory. But how would you show that a decision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

LINK TO WEB CONTENT

https://www.seas.gwu.edu/~ayoussef/cs6212/**greedy**.html

www.cs.umsl.edu/~sanjiv/classes/cs5130/lectures/gm.pdf

www.geeksforgeeks.org/**8-queen-problem**

www.bridges.canterbury.ac.nz/features/eight.html

https://www.hackerearth.com/practice/.../p-**np-complete-np-and-np-hard**

www.cs.princeton.edu/courses/archive/spr11/.../**NP-complete**ness.pdf

UNIT WISE ASSIGNMENT QUESTIONS

UNIT -1

1) What is An Algorithm? Explain Characteristics of an Algorithm?

2) what is Complexity? Discuss with examples Complexity types.

3)With neat diagram discuss Non-recursive Binary Tree Traversals?

4) Explain bi-connected components with examples?

UNIT-2

1) Explain minimum cost spanning tree with example?

2)Discuss and explain Strassen's matrix multiplication?

3) With example explain quicksort sorting method? And find its complexity?

4) Explain dis-joint set operations (find (),union ())?

UNIT-3

1)Explain Multi stage Graphs algorithm with example

2). Write about Reliability design?

3) Explain sum of subset problem using backtracking

4) Describe graph coloring problem and write an algorithm for m-coloring

.

UNIT-4

1) Write an algorithm for Hamiltonian cycle with an example

2) Explain all pairs shortest path problem with example.

3) Explain travelling sales person problem using branch and bound

4)Explain principle of LIFO branch and bound

UNIT-5

1) State and prove cook's theorem

2) Explain deterministic and non-deterministic algorithms

3) Explain Hamiltonian cycles with neat sketch?

4). Explain chromatic number decision problem and clique decision problem

I AND II MID EXAM QUESTION BANK

I MID

1. (a). What is An Algorithm? Explain Characteristics of an Algorithm?

(b). what is Complexity? Discuss with examples Complexity types.

2. (a) With neat diagram discuss Non-recursive Binary Tree Traversals?

(b). Explain bi-connected components with examples?

3. (a). Explain minimum cost spanning tree with example?

(b).Discuss and explain Strassen's matrix multiplication?

4. (a) With example explain quicksort sorting method? And find its complexity?

(b) Explain dis-joint set operations (find (),union ())?

5. (a) Write a procedure to Single Source Shortest Path problem with example?

(b). Discuss binary search algorithm and analyze its time complexity?

6. (a). Discuss Job sequencing with deadlines with example?

(b). Explain about Amortized complexity?

7. (a) Explain in detail job sequencing with deadlines problem with example.

(b)Explain single source shortest path problem with example

8. (a) Explain 0/1 knapsack problem with example

(b) Explain prim's algorithm with example

9. (a) Explain kruskal's algorithm with example

(b)Explain the concept multistage graphs with example

10. (a) Explain breadth first search algorithm with example.

(b)Explain depth first search algorithm with example

II MID

1. (a) Explain Multi stage Graphs algorithm with example

(b) Write about Reliability design?

2.(a). Explainsum of subset problem using backtracking

(b) Describe graph coloring problem and write an algorithm for m-coloring

3.(a) Write an algorithm for Hamiltonian cycle with an example

(b) Explain all pairs shortest path problem with example.

4. (a). Explain travelling sales person problem using branch and bound

(b) Explain principle of LIFO branch and bound

5. (a) State and prove cook's theorem

(b)Explain deterministic and non-deterministic algorithms

6. (a.) Explain Hamiltonian cycles with neat sketch?

(b) Explain chromatic number decision problem and clique decision problem

7. (a.) Explain all pairs shortest path problem with example

(b). Describe the travelling salesman problem and discuss how to solve it using dynamic programming?

8. (a). Explain the concept multistage graphs with example

(b). Explain optimal binary search tree algorithm with example

9. (a). Write an algorithm for N-queens problem using backtracking

(b). Explain subset-sum problem and discuss the possible solution strategies using backtracking

10 (a) Describe graph coloring problem and write an algorithm for m-coloring

(b) Write an algorithm for Hamiltonian cycle with an example