

Dynamic Programming, Backtracking and Branch and Bound

Design & Analysis of Algorithms
ACSE0401

B.Tech 4th Sem

Ms. Parul Goel
Assistant Professor
CSE Department



Brief Introduction About Me

Ms. Parul Goel

Designation : Assistant Professor CSE Department
NIET Greater NOIDA

Qualification:

- B.Tech.(CSE)
- M. Tech.(CSE)

Teaching experience : 03



Evaluation Scheme

NOIDA INSTITUTE OF ENGINEERING & TECHNOLOGY, GREATER NOIDA (An Autonomous Institute)

B. TECH (CSE) EVALUATION SCHEME SEMESTER-IV

Sl. No.	Subject Codes	Subject Name	Periods			Evaluation Scheme				End Semester		Total	Credit
			L	T	P	CT	TA	TOTAL	PS	TE	PE		
1	AAS0402	Engineering Mathematics-IV	3	1	0	30	20	50		100		150	4
2	AASL0401	Technical Communication	2	1	0	30	20	50		100		150	3
3	ACSE0405	Microprocessor	3	0	0	30	20	50		100		150	3
4	ACSE0403A	Operating Systems	3	0	0	30	20	50		100		150	3
5	ACSE0404	Theory of Automata and Formal Languages	3	0	0	30	20	50		100		150	3
6	ACSE0401	Design and Analysis of Algorithm	3	1	0	30	20	50		100		150	4
7	ACSE0455	Microprocessor Lab	0	0	2				25		25	50	1
8	ACSE0453A	Operating Systems Lab	0	0	2				25		25	50	1
9	ACSE0451	Design and Analysis of Algorithm Lab	0	0	2				25		25	50	1
10	ACSE0459	Mini Project using Open Technology	0	0	2				50			50	1

Syllabus

B. TECH. SECOND YEAR			
Course Code	ACSE0401	L T P	Credits
Course Title	Design and Analysis of Algorithm	3 1 0	4
Course objective: Analyze asymptotic performance of algorithms designed using different computational model. Study advanced data structures like Red black Tree, binomial and Fibonacci heap and learn the concept of complexity classes.			
Pre-requisites: Basic knowledge of any programming language like C/C++/ Python/Java, Data Structures, Discrete Structures and Graph Theory			
Course Contents / Syllabus			
UNIT-I	Introduction	8 Hours	
Algorithms, Analyzing Algorithms, Complexity of Algorithms, Amortized Analysis, Growth of Functions, Methods of solving Recurrences, Performance Measurements, Sorting and Order Statistics –Insertion Sort, Shell Sort, Heap Sort, Priority queue, Comparison of Sorting Algorithms, Sorting in Linear Time, Counting Sort, Radix Sort.			
UNIT-II	Advanced Data Structures	8 Hours	
Red-Black Trees, B – Trees, Binomial Heaps, Fibonacci Heaps.			
UNIT-III	Divide and Conquer and Greedy Methods	8 Hours	
Divide and Conquer concepts with Examples Such as Quick sort, Merge sort, Strassen's Matrix Multiplication, Convex Hull, Searching. Greedy Methods with Examples Such as Activity Selection, Task scheduling, Knapsack, Minimum Spanning Trees – Prim's and Kruskal's Algorithms, Single Source Shortest Paths - Dijkstra's and Bellman Ford Algorithms, Huffman codes.			
UNIT-IV	Dynamic Programming, Backtracking, Branch and Bound	8 Hours	
Dynamic Programming concepts, Examples Such as All Pair Shortest Paths – Warshal's and Floyd's Algorithms, 0/1 Knapsack, Longest Common Sub Sequence, Matrix Chain Multiplication, Resource Allocation Problem. Graph searching (BFS, DFS),Backtracking, Branch and Bound with Examples Such as Travelling Salesman Problem, Graph Coloring, n-Queen Problem, Hamiltonian Cycles and Sum of Subsets.			
UNIT-V	Selected Topics	8 Hours	
String Matching Algorithms such as Rabin-karp Matcher, Finite Automaton Matcher, KMP Matcher, Boyer Moore Matcher. Theory of NP-Completeness, Approximation Algorithms and Randomized Algorithms			

Course Objective

- 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.

At the end of the semester, the student will be able:

	Description	Bloom's Taxonomy
CO 1	To have knowledge of basic principles of algorithm design and Analysis, asymptotic notations and growth of functions for time and space complexity analysis and applying the same in different sorting algorithms	Knowledge, analysis And design
CO 2	To apply different problem-solving approaches for advanced data structures	Knowledge, analysis And apply
CO 3	To apply divide and conquer method for solving merge sort, quick sort, matrix multiplication and Greedy Algorithm for solving different Graph Problem.	Knowledge, analysis and Apply
CO 4	To analyze and apply different optimization techniques like dynamic programming, backtracking and Branch & Bound to solve the complex problems	Knowledge, Analysis And Apply
CO 5	To understand the advanced concepts like NP Completeness and Fast Fourier Transform, to analyze and apply String Matching, Approximation and Randomized Algorithms to solve the complex problems	Knowledge, Analysis and Apply

At the end of the semester, the student will be able:

POs	Engineering Graduates will be able to
PO1	Engineering Knowledge
PO2	Problem Analysis
PO3	Design & Development of solutions
PO4	Conduct Investigation of complex problems
PO5	Modern Tool Usage
PO6	The Engineer and Society
PO7	Environment and sustainability
PO8	Ethics
PO9	Individual & Team work
PO10	Communication
PO11	Project management and Finance
PO12	Life Long Learning

CO-PO and PSO Mapping

Design and Analysis of Algorithm (kCS-502)

CO.K	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO1 ₁	PO12
ACSE0401.1	3	3	3	3	2	-	-	-	2	2	-	3
ACSE0401.2	3	3	3	3	2	2	-	1	1	1	-	3
ACSE0401.3	3	3	2	3	3	2	-	2	1	1	2	3
ACSE0401.4	3	3	3	3	2	2	-	2	2	1	3	3
ACSE0401.5	2	2	2	2	2	2	-	2	1	1	1	2
Average	2.8	2.8	2.6	2.8	2.2	1.6	-	1.8	1.4	1.2	1.2	2.8

Program Educational Objectives(PEOs)

- PEO1:** To have an excellent scientific and engineering breadth so as to comprehend, analyze, design and provide sustainable solutions for real-life problems using state-of-the-art technologies.
- PEO2:** To have a successful career in industries, to pursue higher studies or to support entrepreneurial endeavors and to face global challenges.
- PEO3:** To have an effective communication skills, professional attitude, ethical values and a desire to learn specific knowledge in emerging trends, technologies for research, innovation and product development and contribution to society.
- PEO4:** To have life-long learning for up-skilling and re-skilling for successful professional career as engineer, scientist, entrepreneur and bureaucrat for betterment of society

Subject Result: 96.63 %

Department Result: 96.63 %

Faculty-Wise Result:

Mr. Harsh Vardhan Mishra (B): 93.94%

Mr. Harsh Vardhan Mishra(C): 97.01%

Mr. Twinkle Tyagi(A): 96.978%

Mr. Twinkle Tyagi(D): 98.53%

End Semester Question Paper Template

B TECH

(SEM-IV) THEORY EXAMINATION 20__-20__

COMPILER DESIGN

Time: 3 Hours

Total

Marks: 100

Note: 1. Attempt all Sections. If require any missing data; then choose suitably.

SECTION A

Q.No.	Question	Marks	CO
1		2	
2		2	
.		.	
10		2	

End Semester Question Paper Templates

SECTION B

2. Attempt any three of the following:

3 x 10 = 30

Q.No.	Question	Marks	CO
1		10	
2		10	
.		.	
5		10	

3. Attempt any one part of the following:

1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

End Semester Question Paper Templates

4. Attempt any one part of the following:

1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

5. Attempt any one part of the following:

1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

6. Attempt any one part of the following:

1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

- Prerequisite
- Basic concept of c programming language.
- Concept of stack, queue and link list.
- **Recap**
- Flow Chart
- Algorithm

Course Objective

- 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.

- Dynamic programming
- Knapsack
- All pair shortest paths – Warshal's and Floyd's algorithms,
- Resource allocation problem
- Backtracking
- Graph Coloring
- n-Queen Problem
- Hamiltonian Cycles
- Sum of subsets
- Branch and Bound with examples such as Travelling Salesman Problem

Unit Objective

- This objective of this unit is to make students understand about
- Dynamic Programming
- Back tracking
- Branch and Bound ..

Topic Objective

- This objective of this topic is to make students understand about
- Concepts of dynamic programming.
- Knapsack Problem(0/1)
- Floyd Warshall Algorithm
- Resource Allocation problem

Dynamic Programming(CO4)

- In the divide-and-conquer strategy, you divide the problem to be solved into subproblems.
- The subproblems are further divided into smaller subproblems.
- That task will continue until you get subproblems that can be solved easily.
- The basic idea of dynamic programming is to use a table to store the solutions of solved subproblems.
- If you face a subproblem again, you just need to take the solution in the table without having to solve it again.
- Therefore, the algorithms designed by dynamic programming are very effective.

Dynamic Programming



To solve a problem by dynamic programming, you need to do the following tasks:

- Find solutions of the smallest subproblems.
- Find out the formula (or rule) to build a solution of subproblem through solutions of even smallest subproblems.
- Create a table that stores the solutions of subproblems. Then calculate the solution of subproblem according to the found formula and save to the table.
- From the solved subproblems, you find the solution of the original problem.

Knapsack problem(0/1)

- Consider a thief gets into a home to rob and he carries a knapsack. There are fixed number of items in the home — each with its own weight and value — Jewelry, with less weight and highest value vs tables, with less value but a lot heavy.
- To add fuel to the fire, the thief has an old knapsack which has limited capacity.
- Obviously, he can't split the table into half or jewelry into 3/4ths. He either takes it or leaves it.

Dynamic-Programming Approach

- Let i be the highest-numbered item in an optimal solution S for W dollars. Then $S' = S - \{i\}$ is an optimal solution for $W - w_i$ dollars and the value to the solution S is V_i plus the value of the sub-problem.
- We can express this fact in the following formula: define $c[i, w]$ to be the solution for items $1, 2, \dots, i$ and the maximum weight w .
- The algorithm takes the following inputs
- The maximum weight W
- The number of items n
- The two sequences $v = \langle v_1, v_2, \dots, v_n \rangle$ and $w = \langle w_1, w_2, \dots, w_n \rangle$

Dynamic-0-1-knapsack (v, w, n, W)

```
for w = 0 to W do
  c[0, w] = 0
for i = 1 to n do
  c[i, 0] = 0
  for w = 1 to W do
    if  $w_i \leq w$  then
      if  $v_i + c[i-1, w-w_i]$  then
         $c[i, w] = v_i + c[i-1, w-w_i]$ 
      else  $c[i, w] = c[i-1, w]$ 
    else
       $c[i, w] = c[i-1, w]$ 
```


- **Knapsack problem(0/1)**
- The set of items to take can be deduced from the table, starting at $c[n, w]$ and tracing backwards where the optimal values came from.
- If $c[i, w] = c[i-1, w]$, then item i is not part of the solution, and we continue tracing with $c[i-1, w]$. Otherwise, item i is part of the solution, and we continue tracing with $c[i-1, w-W]$.

Analysis

- This algorithm takes $\theta(n, w)$ times as table c has $(n + 1).(w + 1)$ entries, where each entry requires $\theta(1)$ time to compute.

Knapsack problem(0/1)

**A knapsack (kind of shoulder bag) with limited weight capacity.
Few items each having some weight and value.**

The problem states-

- Which items should be placed into the knapsack such that-
- The value or profit obtained by putting the items into the knapsack is maximum.
- And the weight limit of the knapsack does not exceed.

0/1 Knapsack Problem-

- In 0/1 Knapsack Problem,
- As the name suggests, items are indivisible here.
- We can not take the fraction of any item.
- We have to either take an item completely or leave it completely.
- It is solved using dynamic programming approach.

0/1 Knapsack Problem Using Dynamic Programming-

Consider-

- Knapsack weight capacity = w
- Number of items each having some weight and value = n
- 0/1 knapsack problem is solved using dynamic programming in the following steps-

Step-01:

Draw a table say 'c' with $(n+1)$ number of rows and $(w+1)$ number of columns.

Fill all the boxes of 0th row and 0th column with zeroes as shown-

Start filling the table row wise top to bottom from left to right.

Use the following formula-

- $c(i, w) = \max \{ c[i-1, w], v_i + c[i-1, w-w_i] \}$
- Here, $c(i, w)$ = maximum value of the selected items if we can take items 1 to i and have weight restrictions of w .
- This step leads to completely filling the table.
- Then, value of the last box represents the maximum possible value that can be put into the knapsack.

Step-03:

- To identify the items that must be put into the knapsack to obtain that maximum profit,
- Consider the last column of the table.
- Start scanning the entries from bottom to top.
- On encountering an entry whose value is not same as the value stored in the entry immediately above it, mark the row label of that entry.
- After all the entries are scanned, the marked labels represent the items that must be put into the knapsack.

Dynamic Programming(CO4)

Problem-

For the given set of items and knapsack capacity = 5 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

Item	Weight	Value
1	2	3
2	3	4
3	4	5
4	5	6

Dynamic Programming(CO4)

Step-01:

- Draw a table say 'T' with $(n+1) = 4 + 1 = 5$ number of rows and $(w+1) = 5 + 1 = 6$ number of columns.
- Fill all the boxes of 0^{th} row and 0^{th} column with 0.

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0					
2	0					
3	0					
4	0					

T-Table

Step-02:

- Start filling the table row wise top to bottom from left to right using the formula-
- $c(i, w) = \max \{ c[i-1, w], v_i + c[i-1, w-w_i] \}$

Finding T(1,1)-

We have,

$$i = 1$$

$$w = 1$$

$$(\text{value})_i = (\text{value})_1 = 3$$

$$(\text{weight})_i = (\text{weight})_1 = 2$$

Substituting the values, we get-

$$C(1,1) = \max \{ C(1-1, 1), 3 + C(1-1, 1-2) \}$$

$$C(1,1) = \max \{ C(0,1), 3 + C(0,-1) \}$$

$$C(1,1) = T(0,1) \quad \{ \text{Ignore } C(0,-1) \}$$

$$C(1,1) = 0$$

Finding C(1,2)-

$$i = 1$$

$$w = 2$$

$$(\text{value})_i = (\text{value})_1 = 3$$

$$(\text{weight})_i = (\text{weight})_1 = 2$$

Substituting the values, we get-

$$C(1,2) = \max \{ C(1-1, 2), 3 + C(1-1, 2-2) \}$$

$$C(1,2) = \max \{ C(0,2), 3 + C(0,0) \}$$

$$C(1,2) = \max \{ 0, 3+0 \}$$

$$C(1,2) = 3$$

Similar do for $(C(1,3), C(1,4), C(1,5))$

Finding C(2,1)-

$$i = 2$$

$$w = 1$$

$$(\text{value})_i = (\text{value})_2 = 4$$

$$(\text{weight})_i = (\text{weight})_2 = 3$$

Substituting the values, we get-

$$C(2,1) = \max \{ C(2-1, 1), 4 + C(2-1, 1-3) \}$$

$$C(2,1) = \max \{ C(1,1), 4 + C(1,-2) \}$$

$$C(2,1) = C(1,1) \quad \{ \text{Ignore } C(1,-2) \}$$

$$C(2,1) = 0$$

Finding C(2,2)-

$$i = 2$$

$$w = 2$$

$$(\text{value})_i = (\text{value})_2 = 4$$

$$(\text{weight})_i = (\text{weight})_2 = 3$$

Substituting the values, we get-


$$C(2,2) = \max \{ C(2-1, 2), 4 + C(2-1, 2-3) \}$$

$$C(2,2) = \max \{ C(1,2), 4 + C(1,-1) \}$$

$$C(2,2) = C(1,2) \quad \{ \text{Ignore } C(1,-1) \}$$

$$C(2,2) = 3$$

Dynamic Programming(CO4)



	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

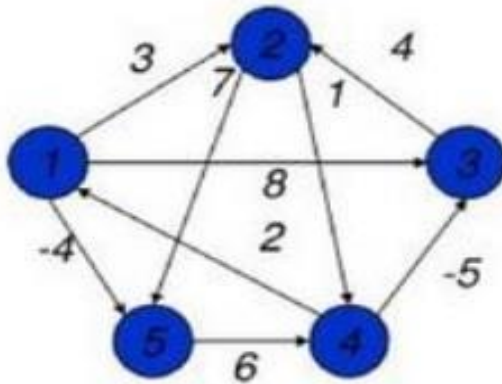
T-Table

All Pair Shortest Path

- The all pair shortest path algorithm is also known as Floyd-Warshall algorithm is used to find all pair shortest path problem from a given weighted graph.
- As a result of this algorithm, it will generate a matrix, which will represent the minimum distance from any node to all other nodes in the graph.
- At first the output matrix is same as given cost matrix of the graph. After that the output matrix will be updated with all vertices k as the intermediate vertex.
- The time complexity of this algorithm is $O(V^3)$, here V is the number of vertices in the graph.

All Pair Shortest Path

- Representation


$$\begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

All Pair Shortest Path

```
Begin
  for k := 0 to n, do
    for i := 0 to n, do
      for j := 0 to n, do
        if cost[i,k] + cost[k,j] < cost[i,j], then
          cost[i,j] := cost[i,k] + cost[k,j]
        done
      done
    done
    display the current cost matrix
  End
```

All Pair Shortest Path

Example: Input-The cost matrix of the graph.

0	3	6	∞	∞	∞	∞
3	0	2	1	∞	∞	∞
6	2	0	1	4	2	∞
∞	1	1	0	2	∞	4
∞	∞	4	2	0	2	1
∞	∞	2	∞	2	0	1
∞	∞	∞	4	1	1	0

- **All Pair Shortest Path**
- Output-Matrix of all pair shortest paths

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

For more examples refer this link

<https://www.youtube.com/watch?v=oNI0rf2P9gE>

Resource Allocation problem

- A resource allocation problem in which 'm' resources are to be allocated to 'n' projects. If 'j' resources are allocated to project I then profit is $P(i,j)$.
- The problem is to allocate the resource to the 'n' projects in such a way as to maximize total next profit.
- This problem can be formulated as an $n+1$ stage graph problem as follows . Stage i , $0 \leq i \leq n-1$, represents project I, there are $m+1$ vertices, associated with stage i , $1 \leq i \leq n-1$, stage 0 and n each one vertex S and T respectively. Vertex (i,j) represents I, resources allocated to projects $1,2,\dots,j$.

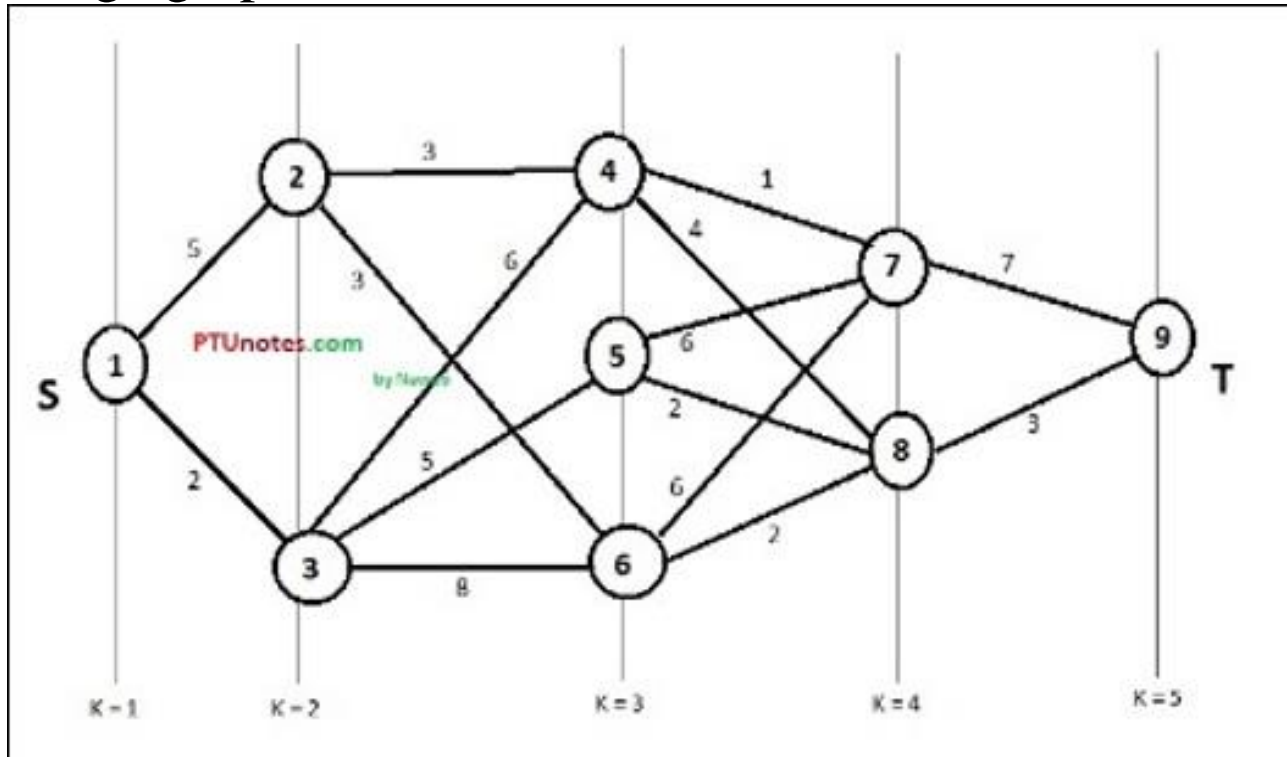
Resource Allocation problem

- An optimal allocation of resources is defined by a maximum cost of a path from S to T.
- A dynamic programming formulation for a k-stage is obtained by first noticing that every S to T path is result of a sequence k-2 decisions.
- Let $c(S,k)$ be a cost of path from node S to k and $d(k, T)$ be the cost of path from node k to T, via. some intermediate node, which can be calculated recursively.

$$d(S,T) = \max \{ c(S,k) + d(k,T) \}$$

Dynamic Programming(CO4)

- Consider the following example to understand the concept of multistage graph.



https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_multistage_graph.htm

- According to the formula, we have to calculate the cost (**i, j**) using the following steps
- **Step-1: Cost (K-2, j)**
- In this step, three nodes (node 4, 5, 6) are selected as **j**. Hence, we have three options to choose the minimum cost at this step.

$$Cost(3, 4) = \min \{c(4, 7) + Cost(7, 9), c(4, 8) + Cost(8, 9)\} = 7$$

$$Cost(3, 5) = \min \{c(5, 7) + Cost(7, 9), c(5, 8) + Cost(8, 9)\} = 5$$

$$Cost(3, 6) = \min \{c(6, 7) + Cost(7, 9), c(6, 8) + Cost(8, 9)\} = 5$$

- **Step-2: Cost (K-3, j)**
- Two nodes are selected as j because at stage $k - 3 = 2$ there are two nodes, 2 and 3. So, the value $i = 2$ and $j = 2$ and 3.

$$\begin{aligned} \text{Cost}(2, 2) &= \min \{c(2, 4) + \text{Cost}(4, 8) + \text{Cost}(8, 9), c(2, 6) + \\ &\quad \text{Cost}(6, 8) + \text{Cost}(8, 9)\} \\ &= 8 \end{aligned}$$

$$\begin{aligned} \text{Cost}(2, 3) &= \{c(3, 4) + \text{Cost}(4, 8) + \text{Cost}(8, 9), c(3, 5) + \text{Cost}(5, 8) + \\ &\quad \text{Cost}(8, 9), c(3, 6) + \text{Cost}(6, 8) + \text{Cost}(8, 9)\} \\ &= 10 \end{aligned}$$

- **Step-3: Cost (K-4, j)**

$$\text{Cost}(1, 1) = \{c(1, 2) + \text{Cost}(2, 6) + \text{Cost}(6, 8) + \text{Cost}(8, 9), c(1, 3) +$$

$$\text{Cost}(3, 5) + \text{Cost}(5, 8) + \text{Cost}(8, 9))\}$$
$$= 12$$

$$c(1, 3) + \text{Cost}(3, 6) + \text{Cost}(6, 8 + \text{Cost}(8, 9))\} = 13$$

Hence, the path having the minimum cost is **1 → 3 → 5 → 8 → 9**.

For more examples refer the video

<https://www.youtube.com/watch?v=9iE9Mj4m8jk>

Backtracking(CO4)- Objective

This objective of this topic is to make students understand about

- Concepts of Backtracking
- Graph Colouring
- N-Queens
- Sum of Subsets
- Hamiltonian Cycle

Prerequisite and Recap

- **Prerequisite**
 - Algorithms Concepts
 - C programming
 - Graph
- **Recap**
 - Dynamic Programming

Backtracking (CO4)

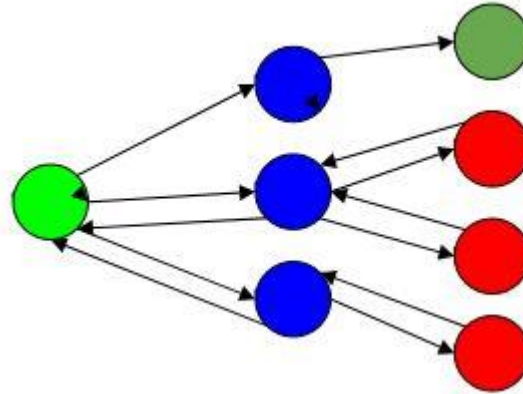
- Backtracking is an algorithmic-technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time.
- It is a technique based on algorithm to solve problem. It uses recursive calling to find the solution by building a solution step by step increasing values with time.
- It removes the solutions that doesn't give rise to the solution of the problem based on the constraints given to solve the problem.

Backtracking

- Backtracking algorithm is applied to some specific types of problems.
 - Decision problem used to find a feasible solution of the problem.
 - Optimisation problem used to find the best solution that can be applied.
 - Enumeration problem used to find the set of all feasible solutions of the problem
- In backtracking problem, the algorithm tries to find a sequence path to the solution which has some small checkpoints from where the problem can backtrack if no feasible solution is found for the problem.

Backtracking

For Example,



- Here, Green is the start point, blue is the intermediate point, red are points with no feasible solution, dark green is end solution.
- when the algorithm propagates to an end to check if it is a solution or not, if it is then returns the solution otherwise backtracks to the point one step behind it to find track to the next point to find solution.

ALGORITHM

Step 1 – if current_position is goal, return success

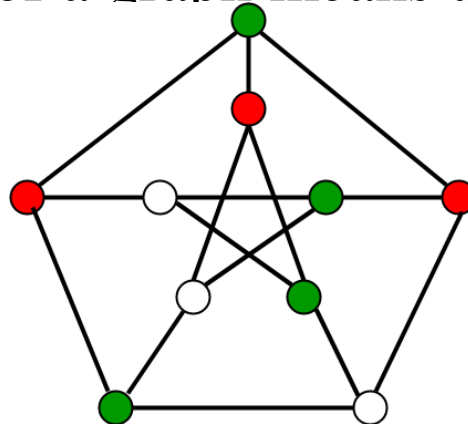
Step 2 – else,

Step 3– if current_position is an end point, return failed.

Step 4– else, if current_position is not end point, explore and repeat above steps.

Graph Colouring Problem

- Given an undirected graph and a number m , determine if the graph can be coloured with at most m colours such that no two adjacent vertices of the graph are colored with the same color.
- Here coloring of a graph means the assignment of colors to all vertices.



Graph Colouring Problem

Input:

- A 2D array $\text{graph}[V][V]$ where V is the number of vertices in graph and $\text{graph}[V][V]$ is adjacency matrix representation of the graph.
- A value $\text{graph}[i][j]$ is 1 if there is a direct edge from i to j , otherwise $\text{graph}[i][j]$ is 0.
- An integer m which is the maximum number of colors that can be used

Output:

An array $\text{color}[V]$ that should have numbers from 1 to m . $\text{color}[i]$ should represent the color assigned to the i th vertex.

N- Queens Problem

- N - Queens problem is to place n - queens in such a manner on an n x n chessboard that no queens attack each other by being in the same row, column or diagonal.
- It can be seen that for $n = 1$, the problem has a trivial solution, and no solution exists for $n = 2$ and $n = 3$.
- So first we will consider the 4 queens problem and then generate it to n - queens problem.
- Given a 4 x 4 chessboard and number the rows and column of the chessboard 1 through 4.

Backtracking

N- Queens Problem

	1	2	3	4
1				
2				
3				
4				

4x4 chessboard

Since, we have to place 4 queens such as q_1 q_2 q_3 and q_4 on the chessboard, such that no two queens attack each other. In such a conditional each queen must be placed on a different row, i.e., we put queen "i" on row "i."

N- Queens Problem

- Now, we place queen q_1 in the very first acceptable position (1, 1).
- Next, we put queen q_2 so that both these queens do not attack each other.
- We find that if we place q_2 in column 1 and 2, then the dead end is encountered.
- Thus the first acceptable position for q_2 in column 3, i.e. (2, 3) but then no position is left for placing queen ' q_3 ' safely.

N- Queens Problem

- So we backtrack one step and place the queen ' q_2 ' in (2, 4), the next best possible solution. Then we obtain the position for placing ' q_3 ' which is (3, 2).
- But later this position also leads to a dead end, and no place is found where ' q_4 ' can be placed safely.
- Then we have to backtrack till ' q_1 ' and place it to (1, 2) and then all other queens are placed safely by moving q_2 to (2, 4), q_3 to (3, 1) and q_4 to (4, 3).

N- Queens Problem

- That is, we get the solution (2, 4, 1, 3).
- This is one possible solution for the 4-queens problem.
- For another possible solution, the whole method is repeated for all partial solutions. The other solutions for 4 - queens problems is (3, 1, 4, 2) i.e.

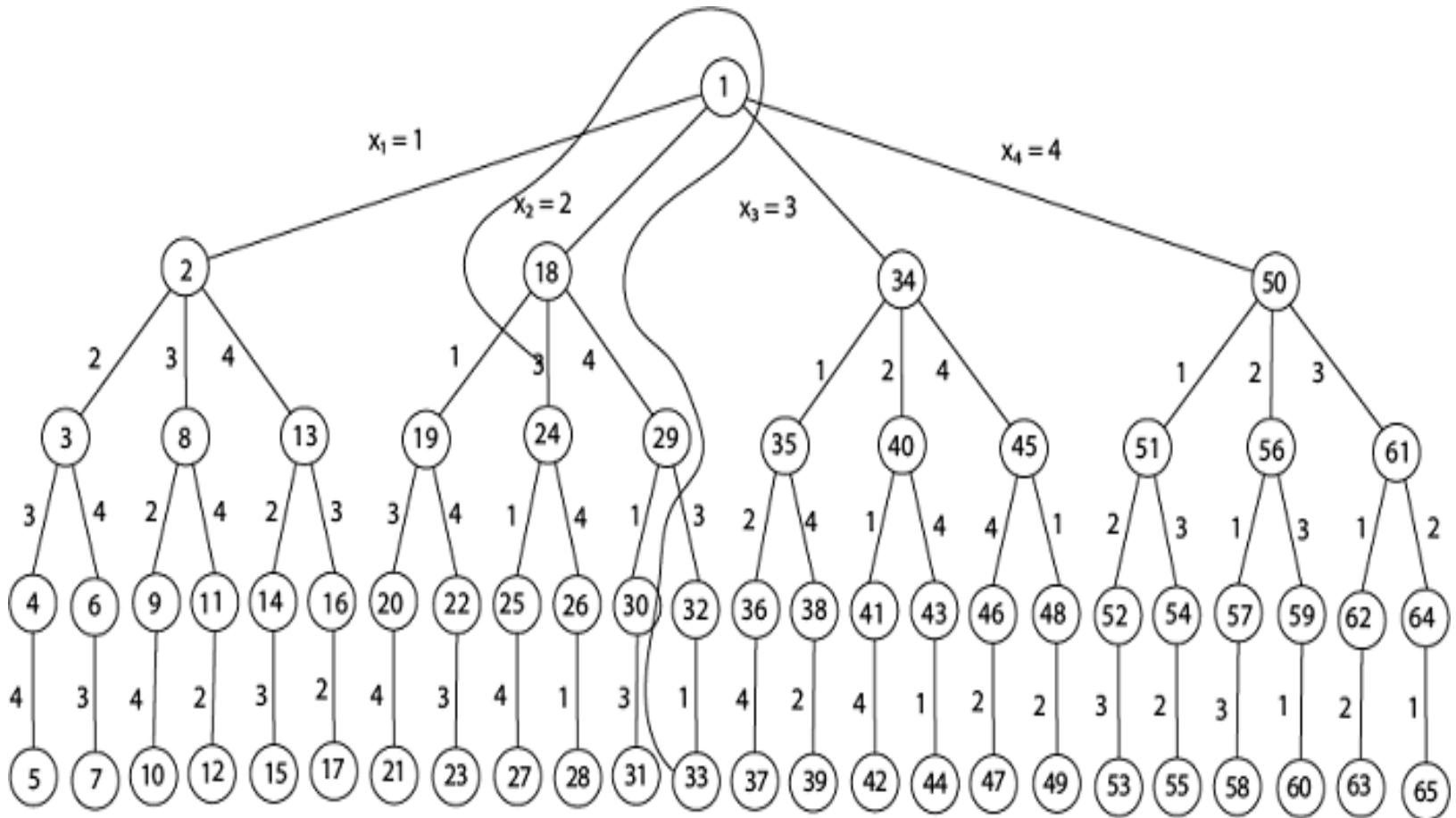
	1	2	3	4
1			q ₁	
2	q ₂			
3				q ₃
4		q ₄		

N- Queens Problem

- Fig shows the complete state space for 4 - queens problem. But we can use backtracking method to generate the necessary node and stop if the next node violates the rule, i.e., if two queens are attacking.
- It can be seen that all the solutions to the 4 queens problem can be represented as 4 - tuples (x_1, x_2, x_3, x_4) where x_i represents the column on which queen " q_i " is placed.

Backtracking

4- Queens Problem



N- Queens Problem

Place (k, i)

```
{  
  For j  $\leftarrow$  1 to k - 1  
    do if (x [j] = i)  
      or (Abs x [j]) - i) = (Abs (j - k))  
  then return false;  
  return true;  
}
```

- Place (k, i) return true if a queen can be placed in the kth row and ith column otherwise return is false.
- x [] is a global array whose final k - 1 values have been set. Abs (r) returns the absolute value of r.

N- Queens Problem

- $x []$ is a global array whose final $k - 1$ values have been set. $Abs (r)$ returns the absolute value of r .

```
N - Queens (k, n)
{
  For  $i \leftarrow 1$  to  $n$ 
    do if Place ( $k, i$ ) then
    {
       $x [k] \leftarrow i$ ;
      if ( $k == n$ ) then
        write ( $x [1....n]$ );
      else
        N - Queens ( $k + 1, n$ );
    }
}
```

Backtracking

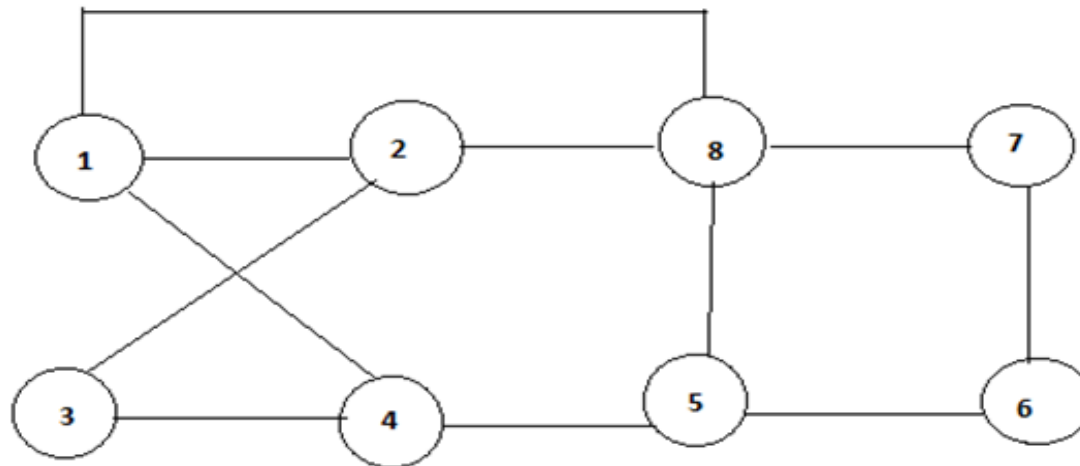
8- Queens Problem

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

1. Thus, the solution **for 8** - queen problem **for** (4, 6, 8, 2, 7, 1, 3, 5).
2. If two queens are placed at position (i, j) and (k, l).
3. Then they are on same diagonal only **if** $(i - j) = k - l$ or $i + j = k + l$.
4. The first equation implies that $j - l = i - k$.
5. The second equation implies that $j - l = k - i$.
6. Therefore, two queens lie on the duplicate diagonal **if** and only **if** $|j - l| = |i - k|$

Hamiltonian Cycles

- Hamiltonian Path in an undirected graph is a path that visits each vertex exactly once.
- A Hamiltonian cycle (or Hamiltonian circuit) is a Hamiltonian Path such that there is an edge (in the graph) from the last vertex to the first vertex of the Hamiltonian Path.



hamiltonian path is = 1 -- 2 -- 3 -- 4 -- 5 -- 6 -- 7 -- 8 -- 1

Sum of Subsets Problem

- Subset sum problem is to find subset of elements that are selected from a given set whose sum adds up to a given number K.
- We are considering the set contains non-negative values. It is assumed that the input set is unique (no duplicates are presented).
- Here backtracking approach is used for trying to select a valid subset when an item is not valid, we will backtrack to get the previous subset and add another element to get the solution.

Sum of Subsets Problem

- Subset sum problem is to find subset of elements that are selected from a given set whose sum adds up to a given number K.
- We are considering the set contains non-negative values. It is assumed that the input set is unique (no duplicates are presented).
- Here backtracking approach is used for trying to select a valid subset when an item is not valid, we will backtrack to get the previous subset and add another element to get the solution.

Sum of Subsets Problem

Backtracking Algorithm for Subset Sum

Using exhaustive search we consider all subsets irrespective of whether they satisfy given constraints or not. Backtracking can be used to make a systematic consideration of the elements to be selected.

Algorithm

subsetSum(set, subset, n, subSize, total, node, sum)

Input: The given set and subset, size of set and subset, a total of the subset, number of elements in the subset and the given sum.

Output: All possible subsets whose sum is the same as the given sum.

Sum of Subsets Algorithm

Begin

if total = sum, then

display the subset

//go for finding next subset

subsetSum(set, subset, , subSize-1, total-set[node], node+1, sum)

return

else

for all element i in the set, do

subset[subSize] := set[i]

subSetSum(set, subset, n, subSize+1, total+set[i], i+1, sum)

done

End

Sum of Subsets Example

Given a set $S = (3, 4, 5, 6)$ and $X = 9$. Obtain the subset sum using Backtracking approach.

Solution:

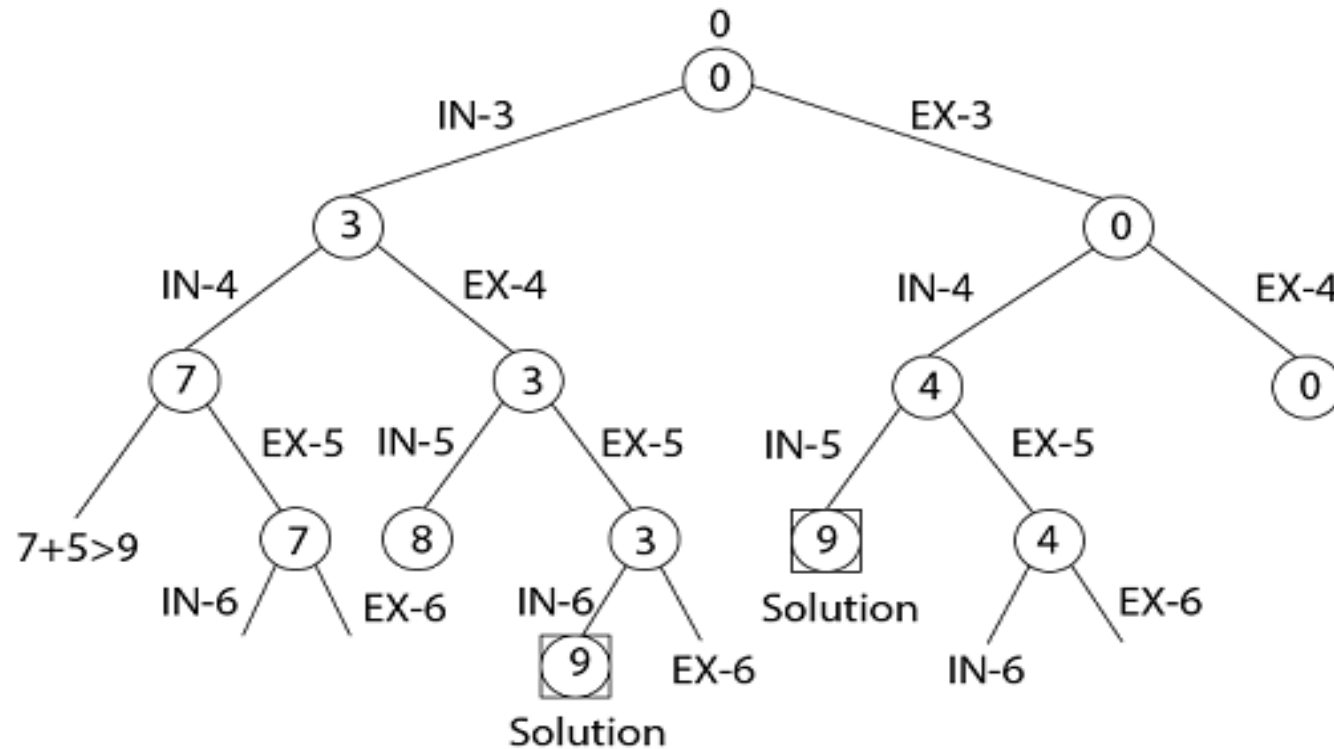
Initially $S = (3, 4, 5, 6)$ and $X = 9$.

$S' = (\emptyset)$

- The implicit binary tree for the subset sum problem is shown in Fig.
- The number inside a node is the sum of the partial solution elements at a particular level.
- Thus, if our partial solution elements sum is equal to the positive integer 'X' then at that time search will terminate, or it continues if all the possible solution needs to be obtained.

Backtracking

Sum of Subsets Example



Sum of Subsets Example

Input: The Set: {10, 7, 5, 18, 12, 20, 15}

The sum Value: 35

Output:

All possible subsets of the given set, where sum of each element for every subsets is same as the given sum value.

{10, 7, 18}

{10, 5, 20}

{5, 18, 12}

{20, 15}

Branch and Bound(CO4)- Objective

This objective of this topic is to make students understand about

- Concepts of Branch and Bound
- Travelling Salesman Problem

Prerequisite and Recap

- **Prerequisite**
 - Algorithms Concepts
 - C programming
 - Graph
-
- **Recap**
 - Backtracking

Branch and Bound (CO4)

- **Branch and bound** is an algorithm design paradigm which is generally used for solving combinatorial optimization problems.
- These problems are typically exponential in terms of time complexity and may require exploring all possible permutations in worst case.
- The Branch and Bound Algorithm technique solves these problems relatively quickly.

Travelling Salesman Problem

Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible tour that visits every city exactly once and returns to the starting point.

- In Branch and Bound method, for current node in tree, we compute a bound on best possible solution that we can get if we down this node.
- If the bound on best possible solution itself is worse than current best (best computed so far), then we ignore the subtree rooted with the node.

Travelling Salesman Problem

Note that the cost through a node includes two costs.

- 1) Cost of reaching the node from the root (When we reach a node, we have this cost computed)
- 2) Cost of reaching an answer from current node to a leaf (We compute a bound on this cost to decide whether to ignore subtree with this node or not).

Travelling Salesman Problem- Example

- In this method we expand the node which is most promising means the node which promises that expanding or choosing it will give us the optimal solution.
- So we prepare the tree starting from root then we expand it.
- The cost matrix is defined by
$$C(i,j) = W(i,j) \text{ , if there is a direct path from } C_i \text{ to } C_j.$$
$$= \infty \text{ , If there is no direct path from } C_i \text{ to } C_j.$$

Travelling Salesman Problem- Example

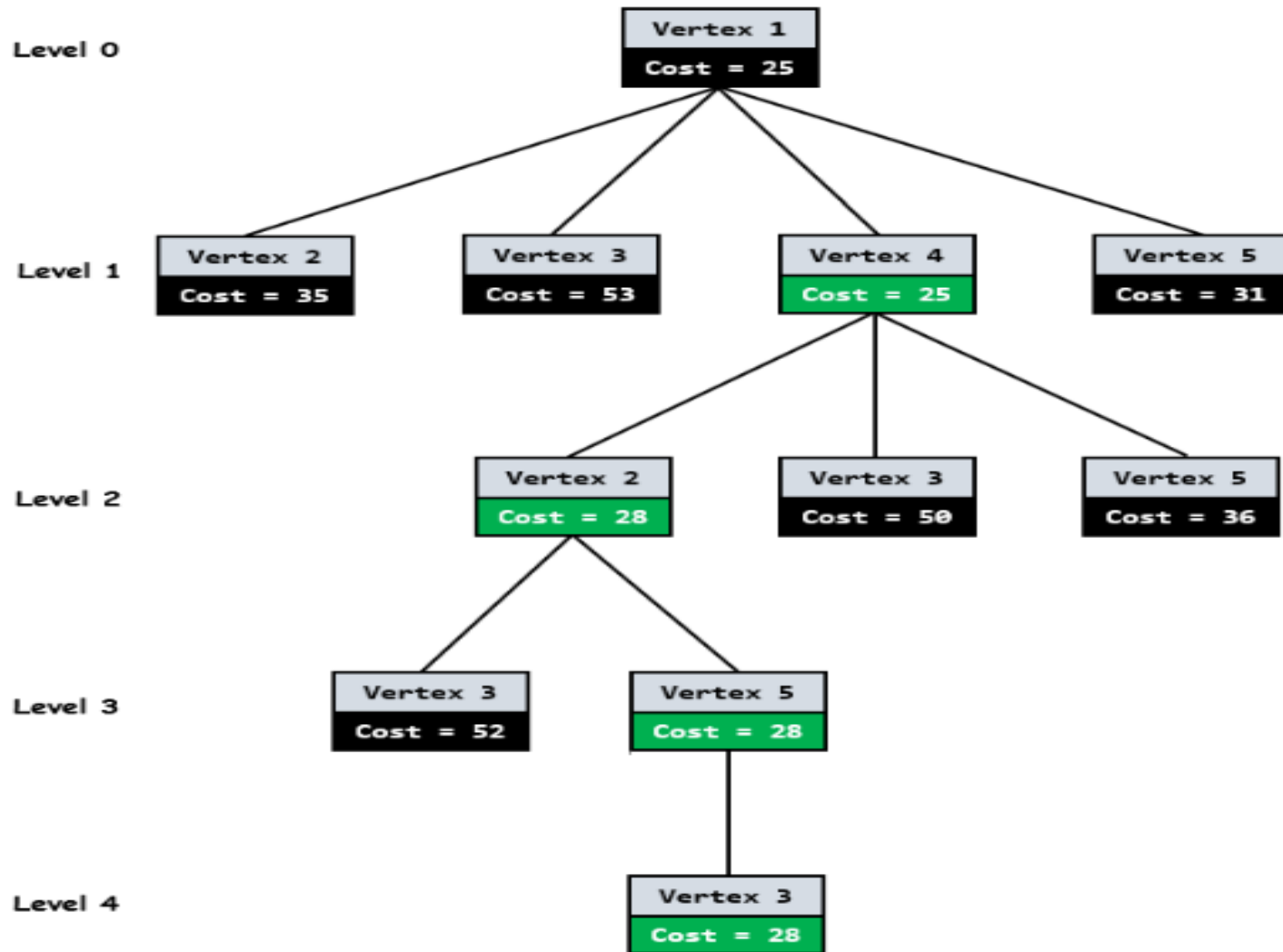
Consider the following cost matrix

	C0	C1	C2	C3	C4
C0	INF	20	30	10	11
C1	15	INF	16	4	2
C2	3	5	INF	2	4
C3	19	6	18	INF	3
C4	16	4	7	16	INF

Here $C(0,2)=30$, $C(4,0)=16$, $C(1,1)=\text{Infinity}(\infty)$ and so on. Below is the state space tree for the TSP, which shows optimal solution marked in green

<https://www.techiedelight.com/travelling-salesman-problem-using-branch-and-bound/>

Branch and Bound



State Space Diagram

Travelling Salesman Problem- Steps

- As we can see from above diagram , every node has a cost associated to it.
- When we go from city I to city j , cost of a node j will be sum of cost of parent node I , cost of the edge (I,j) and lower bound of the path starting at node j .
- As root node is the first node to be expanded , it doesn't have any parent. So cost will be only lower bound of the path starting at root.
- To get the lower bound of the path starting from the node , we reduce each row and column in such a way that there must be atleast one zero in each row and column.

Branch and Bound

- For doing this, we need to reduce the minimum value from each element in each row and column.

Let's start from root node

- We reduce the minimum value in each row from each element in that row.
- Minimum in each row of cost matrix M is marked by blue [10,2,2,3,4] below.

INF	20	30	10	11
15	INF	16	4	2
3	5	INF	2	4
19	6	18	INF	3
16	4	7	16	INF

Branch and Bound

- After reducing the row, we get the below matrix. We then reduce the minimum value in each column from each element in that column.

INF	10	20	0	1
13	INF	14	2	0
1	3	INF	0	2
16	3	15	INF	0
12	0	3	12	INF

- Minimum in each column is marked by blue[1 0 3 0 0]. After reducing the column, we get below reduced matrix. This matrix will be further processed by child nodes of root node to calculate their lower bound.

Branch and Bound

.

INF	10	17	0	1
12	INF	11	2	0
0	3	INF	0	2
15	3	12	INF	0
11	0	0	12	INF

- The total expected cost at the root node is sum of all reductions.
- $\text{Cost} = [10 \ 2 \ 2 \ 3 \ 4] + [1 \ 0 \ 3 \ 0 \ 0] = 25$
- Since we have discovered the root node C0, the next node to be expanded can be any node from C1, C2, C3, C4. Whichever node has minimum cost, that node will be expanded further. So we have to find out the expanding cost of each node.

Branch and Bound

- The parent node (C0) has below reduced matrix-

INF	10	17	0	1
12	INF	11	2	0
0	3	INF	0	2
15	3	12	INF	0
11	0	0	12	INF

- Let us consider edge from 0 → 1**
 - As we are adding edge (0,1) to our search space, we get outgoing edges for city 0 to infinity and all incoming edges to city 1 to infinity. We also set (1,0) to infinity.
 - So in reduced matrix of parent node we change all the elements in row 0 and column 1 and at index (1,0) to infinity (marked in red).

Branch and Bound

.

INF	10	17	0	1
12	INF	11	2	0
0	3	INF	0	2
15	3	12	INF	0
11	0	0	12	INF

The resulting cost matrix is

INF	INF	INF	INF	INF
INF	INF	11	2	0
0	INF	INF	0	2
15	INF	12	INF	0
11	INF	0	12	INF

- .
- 3. We try to calculate lower bound of the path starting at node 1 using above resulting cost matrix.
- 4. The lower bound is 0 as matrix is already in reduced form. i.e. all rows and all columns have zero value.

Therefore for node 1, cost will be

$$\begin{aligned}\text{Cost} &= \text{cost of node 0} + \text{cost of the edge (0,1)} + \text{lower bound of the} \\ &\quad \text{path starting at node 1} \\ &= 25 + 10 + 0 = 35\end{aligned}$$

Branch and Bound

Lets consider edge from 0 \rightarrow 2

1. Change all the elements in row 0 and column 2 and at index(2,0) to infinity(marked in red).

INF	10	17	0	1
12	INF	11	2	0
0	3	INF	0	2
15	3	12	INF	0
11	0	0	12	INF

The resulting cost matrix is:

INF	INF	INF	INF	INF
12	INF	INF	2	0
INF	3	INF	0	2
15	3	INF	INF	0
11	0	INF	12	INF

Branch and Bound

- Now calculate lower bound of the path starting at node 2 using the Approach discussed earlier. Resulting matrix will be-

INF	INF	INF	INF	INF
1	INF	INF	2	0
INF	3	INF	0	2
4	3	INF	INF	0
0	0	INF	12	INF

Therefore the cost of node 2, cost will be

$$\begin{aligned}\text{Cost} &= \text{Cost of node 0} + \text{cost of the edge (0,2)} + \text{Lower bound of the} \\ &\quad \text{path starting at node 2} \\ &= 25 + 17 + 11 = 53\end{aligned}$$

Branch and Bound

Let us consider edge from 0 → 3

1. Change the elements in row 0 and column 3 and at index(3,0) to Infinity(marked in red).

INF	10	17	0	1
12	INF	11	2	0
0	3	INF	0	2
15	3	12	INF	0
11	0	0	12	INF

The Resulting cost matrix is :

INF	INF	INF	INF	INF
12	INF	11	INF	0
0	3	INF	INF	2
INF	3	12	INF	0
11	0	0	INF	INF

Branch and Bound

- Now calculate lower bound of the path starting at node 3 using the Approach discussed earlier.
- Lower bound of the path starting at node 3 is 0 as it is already in reduced form, i.e. all rows and all columns have zero value.

Therefore for node 3, cost will be

$$\begin{aligned}\text{Cost} &= \text{Cost of node 0} + \text{cost of the edge (0,3)} + \text{Lower bound of the} \\ &\quad \text{path starting at node 3} \\ &= 25 + 0 + 0 = 25\end{aligned}$$

Similarly we calculate for $0 \rightarrow 4$. Its cost will be 31.

Branch and Bound

- Now we find a live node with estimated cost. Live nodes 1,2,3 and 4 has costs 35, 53, 25 and 31 respectively.
- The minimum among them is node 3 having cost 25. So node 3 will be expanded further as shown in state space tree diagram.
- After adding its children to list of live nodes, we again find a live node with least cost and expand it.
- We continue the search till a leaf is encountered in space search tree. If a leaf is encountered, then the tour is completed and we will return back to the root node.

Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details

- Youtube/other Video Links
- <https://www.youtube.com/watch?v=oNI0rf2P9gE>
- <https://www.youtube.com/watch?v=9iE9Mj4m8jk>
- <https://www.youtube.com/watch?v=kyLxTdsT8ws>
- <https://nptel.ac.in/courses/106106133/>

Quiz

1. The problem of placing n queens in a chessboard such that no two queens attack each other is called as_____.
2. _____ enumerates a list of promising nodes that could be computed to give the possible solutions of a given problem.
3. When dynamic programming is applied to a problem, it takes far less time as compared to other methods that don't take advantage of overlapping subproblems.(True/False).
4. Branch and bound is a _____
5. Both LIFO branch and bound strategy and backtracking leads to depth first search.(True/False)

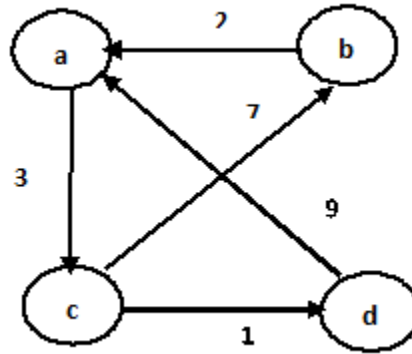
Quiz

6. _____ can traverse the state space tree only in DFS manner?
7. A node is said to be _____ if it has a possibility of reaching a complete solution.
8. A state-space tree for a backtracking algorithm is constructed using _____.
9. The leaves in a state-space tree represent only complete solutions.(True / False)
10. Backtracking algorithm is faster than the brute force technique(True/ False)

Weekly Assignment

1. Explain Floyd algorithm and also analyze its complexity. [CO4]
2. What is nQueens problem? Write the steps for solving the nQueens problem. [CO4]
3. Explain Graph colouring Problem with the help of suitable example. [CO4]
4. Find the optimal solution to the knapsack 0/1 instances $n=7, m=15$.
 $(V_1, V_2, V_3, V_4, V_5, V_6, V_7) = (10, 5, 15, 7, 6, 18, 3)$
 $(W_1, W_2, W_3, W_4, W_5, W_6, W_7) = (2, 3, 5, 7, 1, 4, 1)$ [CO4]
5. Let $S = \{4, 6, 7, 8\}$ and $m=18$. Find all possible subsets of S that sums to m . Draw state space tree that is generated. [CO4]
6. Use Floyd Warshall algorithm to find shortest path for all of the vertices of the given graph. Give the complexity of the algorithm. [CO4]

Weekly Assignment



6. Consider following instance for simple Knapsack Problem. Find the solution using greedy method. $N:8$, Value of knapsack is 110.

$V : \{ 30, 20, 100, 90, 160, 200, 180, 220 \}$

$W : \{ 5, 10, 20, 22, 33, 43, 45, 55 \}$

[CO4]

7. Write short note on Branch and Bound Technique.

[CO4]

1. Which of the following is/are property/properties of a dynamic programming problem?
 - a) Optimal substructure
 - b) Overlapping subproblems
 - c) Greedy approach
 - d) Both optimal substructure and overlapping subproblems

2. If a problem can be broken into subproblems which are reused several times, the problem possesses _____ property.
 - a) Overlapping subproblems
 - b) Optimal substructure
 - c) Memoization
 - d) Greedy

3. When dynamic programming is applied to a problem, it takes far less time as compared to other methods that don't take advantage of overlapping subproblems.
- a) True
 - b) False
4. Which of the following problems is NOT solved using dynamic programming?
- a) 0/1 knapsack problem
 - b) Matrix chain multiplication problem
 - c) Edit distance problem
 - d) Fractional knapsack problem

5. Which of the following problems should be solved using dynamic programming?
- a) Mergesort
 - b) Binary search
 - c) Longest common subsequence
 - d) Quicksort
6. If a problem can be solved by combining optimal solutions to non-overlapping problems, the strategy is called _____
- a) Dynamic programming
 - b) Greedy
 - c) Divide and conquer
 - d) Recursion

7. Which of the problems cannot be solved by backtracking method?
- a) n-queen problem
 - b) subset sum problem
 - c) hamiltonian circuit problem
 - d) travelling salesman problem
8. Backtracking algorithm is implemented by constructing a tree of choices called as?
- a) State-space tree
 - b) State-chart tree
 - c) Node tree
 - d) Backtracking tree

9. In what manner is a state-space tree for a backtracking algorithm constructed?
- a) Depth-first search
 - b) Breadth-first search
 - c) Twice around the tree
 - d) Nearest neighbour first
10. The problem of finding a subset of positive integers whose sum is equal to a given positive integer is called as?
- a) n- queen problem
 - b) subset sum problem
 - c) knapsack problem
 - d) hamiltonian circuit problem

Glossary Question

Q.1 The 0/1 Knapsack problem is an example of _____

- a) Greedy algorithm
- b) 2D dynamic programming**
- c) 1D dynamic programming
- d) Divide and conquer

Q.2 _____ method can be used to solve the Knapsack problem?

- a) Brute force algorithm
- b) Recursion
- c) Dynamic programming
- d) Brute force, Recursion and Dynamic Programming**

Q.3 _____ is used to solve the matrix chain multiplication problem?

- a) Dynamic programming
- b) Brute force
- c) Recursion
- d) Dynamic Programming, Brute force, Recursion**

Glossary Question

Q.4 _____ problems can be solved using the longest subsequence problem?

- a) Longest increasing subsequence
- b) Longest palindromic subsequence
- c) Longest bitonic subsequence
- d) Longest decreasing subsequence

Q.5 _____ is the worst case time complexity of dynamic programming solution of the subset sum problem(sum=given subset sum)?

- a) $O(n)$
- b) $O(\text{sum})$
- c) $O(n^2)$
- d) $O(\text{sum} * n)$

Q.6 In _____ directions queens attack each other?

- a) 1
- b) 2
- c) 3
- d) 4

Glossary Question

Q.7 Placing n-queens so that no two queens attack each other is called_____

- a) n-queen's problem
- b) 8-queen's problem
- c) Hamiltonian circuit problem
- d) subset sum problem

Q.8 The n-queens problem implemented in _____.

- a) carom
- b) chess
- c) ludo
- d) cards

Q.9 _____methods can be used to solve n-queen's problem.

- a) greedy algorithm
- b) divide and conquer
- c) iterative improvement
- d) backtracking

Old Question Papers(2019-2020)

Printed Page 1 of 2

Sub Code: RCS502

Paper Id: **110502**

Roll No:

--	--	--	--	--	--	--	--	--	--	--	--	--	--

**B. TECH.
(SEM V) THEORY EXAMINATION 2019-20
DESIGN AND ANALYSIS OF ALGORITHM**

Time: 3 Hours

Total Marks: 70

Note: 1. Attempt all Sections. If require any missing data; then choose suitably.

SECTION A

1. Attempt *all* questions in brief.

2 x 7 = 14

- a. How do you compare the performance of various algorithms?
- b. Take the following list of functions and arrange them in ascending order of growth rate. That is, if function $g(n)$ immediately follows function $f(n)$ in your

Old Question Papers(2019-2020)

list, then it should be the case that $f(n)$ is $O(g(n))$.

$f_1(n) = n^{2.5}$, $f_2(n) = \sqrt{2^n}$, $f_3(n) = n + 10$, $f_4(n) = 10n$, $f_5(n) = 100n$, and
 $f_6(n) = n^2 \log n$

- c. What is advantage of binary search over linear search? Also, state limitations of binary search.
- d. What are greedy algorithms? Explain their characteristics?
- e. Explain applications of FFT.
- f. Define feasible and optimal solution.
- g. What do you mean by polynomial time reduction?

SECTION B

2. Attempt any *three* of the following:

7 x 3 = 21

- a. (i) Solve the recurrence $T(n) = 2T(n/2) + n^2 + 2n + 1$
(ii) Prove that worst case running time of any comparison sort is $\Omega(n \log n)$

Old Question Papers(2019-2020)

- b. Insert the following element in an initially empty RB-Tree.
12, 9, 81, 76, 23, 43, 65, 88, 76, 32, 54. Now Delete 23 and 81.
- c. Define spanning tree. Write Kruskal's algorithm for finding minimum cost spanning tree. Describe how Kruskal's algorithm is different from Prim's algorithm for finding minimum cost spanning tree.
- d. What is dynamic programming? How is this approach different from recursion? Explain with example.
- e. Define NP-Hard and NP- complete problems. What are the steps involved in proving a problem NP-complete? Specify the problems already proved to be NP-complete.

SECTION C

3. Attempt any *one* part of the following:

7 x 1 = 7

- (a) Among Merge sort, Insertion sort and quick sort which sorting technique is the best in worst case. Apply the best one among these algorithms to Sort the list E, X, A, M, P, L, E in alphabetic order.
- (b) Solve the recurrence using recursion tree method:
$$T(n) = T(n/2) + T(n/4) + T(n/8) + n$$

Old Question Papers(2019-2020)

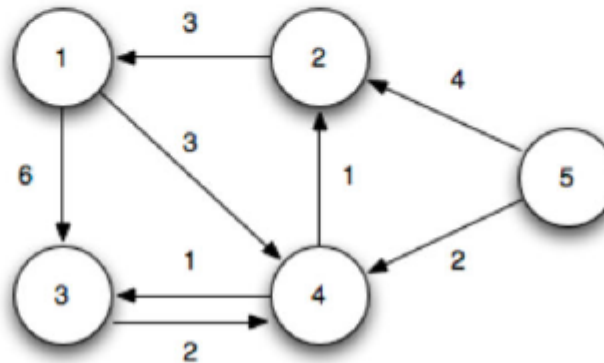
4. **Attempt any *one* part of the following:** **7 x 1 = 7**
- (a) Using minimum degree 't' as 3, insert following sequence of integers 10, 25, 20, 35, 30, 55, 40, 45, 50, 55, 60, 75, 70, 65, 80, 85 and 90 in an initially empty B-Tree. Give the number of nodes splitting operations that take place.
 - (b) Explain the algorithm to delete a given element in a binomial Heap. Give an example for the same.
5. **Attempt any *one* part of the following:** **7 x 1 = 7**
- (a) Compare the various programming paradigms such as divide-and-conquer, dynamic programming and greedy approach.
 - (b) What do you mean by convex hull? Describe an algorithm that solves the convex hull problem. Find the time complexity of the algorithm.

Old Question Papers(2019-2020)

6. Attempt any *one* part of the following:

7 x 1 = 7

- (a) Solve the following 0/1 knapsack problem using dynamic programming.
 $P=[11,21,31,33]$ $w=[2,11,22,15]$ $c=40$, $n=4$.
- (b) Define Floyd Warshall Algorithm for all pair shortest path and apply the same on following graph:



7. Attempt any *one* part of the following:

7 x 1 = 7

- (a) Describe in detail Knuth-Morris-Pratt string matching algorithm. Compute the prefix function π for the pattern ababbabbabbababbabb when the alphabet is $\Sigma = \{a,b\}$.
- (b) What is an approximation algorithm? What is meant by P (n) approximation algorithms? Discuss approximation algorithm for Travelling Salesman Problem.

Old Question Papers(2018-2019)

Printed Pages: 02

Subject Code: RCS502

Paper Id: **110502**

Roll No:

--	--	--	--	--	--	--	--	--	--

B TECH
(SEM V) THEORY EXAMINATION 2018-19
DESIGN & ANALYSIS OF ALGORITHMS

Time: 3 Hours

Total Marks: 70

Note: 1. Attempt all Sections. If require any missing data; then choose suitably.
2. Any special paper specific instruction.

SECTION A

1. Attempt *all* questions in brief. **2 x 7 = 14**
- a. Rank the following by growth rate:
 n , $2^{\lg \sqrt{n}}$, $\log n$, $\log(\log n)$, $\log^2 n$, $(\lg n)^{\lg n}$, 4 , $(3/2)^n$, $n!$
 - b. Prove that if $n \geq 1$, then for any n -key B-Tree of height h and minimum degree $t \geq 2$, $h \leq \log_t((n+1)/2)$.
 - c. Define principal of optimality. When and how dynamic programming is applicable.
 - d. Explain application of graph coloring problem.
 - e. Compare adjacency matrix and linked Adjacency lists representation of a Graph with suitable example/diagram.
 - f. What are approximation algorithms? What is meant by $P(n)$ approximation algorithms?
 - g. What do you mean by stability of a sorting algorithm? Explain its application.

Old Question Papers(2018-2019)

SECTION B

2. Attempt any *three* of the following:

7 x 3 = 21

- a. Use a recursion tree to give an asymptotically tight solution to the recurrence $T(n) = T(\alpha n) + T((1 - \alpha)n) + cn$, where α is a constant in the range $0 < \alpha < 1$ and $c > 0$ is also a constant.
- b. Define BNP, NP hard and NP Complete Problems. Prove that Travelling Salesman Problem is NP-Complete.
- c. Consider the weights and values of items listed below. Note that there is only one unit of each item. The task is to pick a subset of these items such that their total weight is no more than 11 Kgs and their total value is maximized. Moreover, no item may be split. The total value of items picked by an optimal algorithm is denoted by V_{opt} . A greedy algorithm sorts the items by their value-to-weight ratios in descending order and packs them greedily, starting from the first item in the ordered list. The total value of items picked by the greedy algorithm is denoted by V_{greedy} . Find the value of $V_{\text{opt}} - V_{\text{greedy}}$

Old Question Papers(2018-2019)

Item	I ₁	I ₂	I ₃	I ₄
W	10	7	4	2
V	60	28	20	24

- d. Insert the following keys in a 2-3-4 B Tree:
40, 35, 22, 90, 12, 45, 58, 78, 67, 60 and then delete key 35 and 22 one after other.
- e. Prove that if the weights on the edge of the connected undirected graph are distinct then there is a unique Minimum Spanning Tree. Give an example in this regard. Also discuss Prim's Minimum Spanning Tree Algorithm in detail.

SECTION C

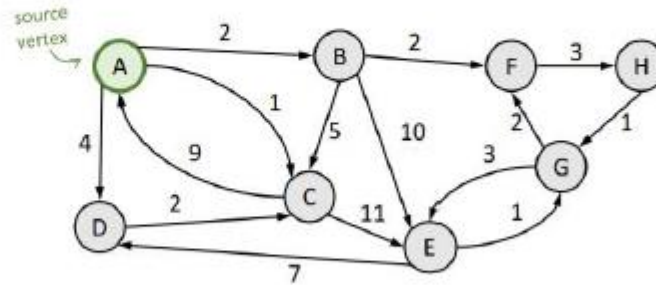
3. Attempt any *one* part of the following: 7 x 1 = 7
- (a) The recurrence $T(n) = 7T(n/3) + n^2$ describes the running time of an algorithm A. Another competing algorithm B has a running time of $S(n) = aS(n/9) + n^2$. What is the smallest value of 'a' such that A is asymptotically faster than B.?
- (b) How will you sort following array A of elements using heap sort:
 $A = (23, 9, 18, 45, 5, 9, 1, 17, 6)$.
4. Attempt any *one* part of the following: 7 x 1 = 7
- (a) Explain the different conditions of getting union of two existing binomial Heaps. Also write algorithm for union of two Binomial Heaps. What is its complexity?
- (b) Insert the elements 8, 20, 11, 14, 9, 4, 12 in a Red-Black Tree and delete 12, 4, 9, 14 respectively.

Old Question Papers(2018-2019)

5. Attempt any *one* part of the following:

7 x 1 = 7

- (a) When do Dijkstra and the Bellman-Ford algorithm both fail to find a shortest path? Can Bellman ford detect all negative weight cycles in a graph? Apply Bellman Ford Algorithm on the following graph:



- (b) Given an integer x and a positive number n , use divide & conquer approach to write a function that computes x^n with time complexity $O(\log n)$.

6. Attempt any *one* part of the following:

7 x 1 = 7

- (a) Solve the Subset sum problem using Backtracking, where $n=4$, $m=18$, $w[4] = \{5, 10, 8, 13\}$
- (b) Give Floyd War shall algorithm to find the shortest path for all pairs of vertices in a graph. Give the complexity of the algorithm. Explain with example.

Old Question Papers(2018-2019)

7. Attempt any *one* part of the following:

7 x 1 = 7

- (a) What is the application of Fast Fourier Transform (FFT)? Also write the recursive algorithm for FFT.
- (b) Give a linear time algorithm to determine if a text T is a cycle rotation of another string T'.
For example, 'RAJA' and 'JARA' are cyclic rotations of each other.

Old Question Papers(2017-2018)

Printed pages: 01

Sub Code: ECS502

Paper Id:

1	0	3	6
---	---	---	---

Roll No:

--	--	--	--	--	--	--	--	--	--

B TECH
(SEM V) THEORY EXAMINATION 2017-18
DESIGN AND ANALYSIS OF ALGORITHMS

Time: 3 Hours

Total Marks: 100

Notes: Attempt all Sections. Assume any missing data.

SECTION-A

1. Define/Explain the following:

(2*10=20)

- (a) Difference between Complete Binary Tree and Binary Tree?
- (b) Difference between Greedy Technique and Dynamic programming.
- (c) Solve the following recurrence using Master method:
$$T(n) = 4T(n/3) + n^2$$
- (d) Name the sorting algorithm that is most practically used and also write its Time Complexity.
- (e) Find the time complexity of the recurrence relation
$$T(n) = n + T(n/10) + T(7n/5)$$
- (f) Explain Single source shortest path.
- (g) Define Graph Coloring.
- (h) Compare Time Complexity with Space Complexity.
- (i) What are the characteristics of the algorithm?
- (j) Differentiate between Backtracking and Branch and Bound Techniques.

Old Question Papers(2017-2018)

SECTION-B

2. Attempt any three of the following:

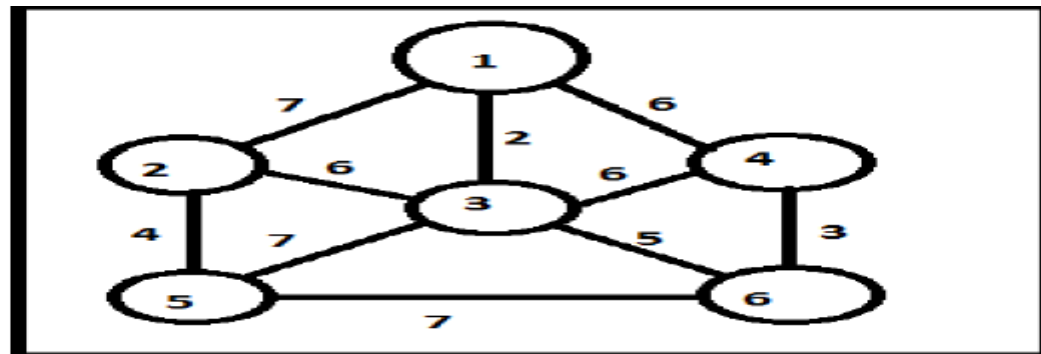
(10×3=30)

(a) Solve the following By Recursion Tree Method

$$T(n) = n + T(n/5) + T(4n/5)$$

(b) Insert the following information F, S, Q, K, C, L, H, T, V, W, M, R, N, P, A, B, X, Y, D, Z, E, G, I. Into an empty B-tree with degree $t=3$.

(c) What is Minimum Cost Spanning Tree? Explain Kruskal's Algorithm and Find MST of the Graph. Also write its Time-Complexity



(d) What is Red-Black tree? Write an algorithm to insert a node in an empty red-black tree explain with suitable example.

(e) Explain HEAP-SORT on the array. Illustrate the operation of HEAP-SORT on the array

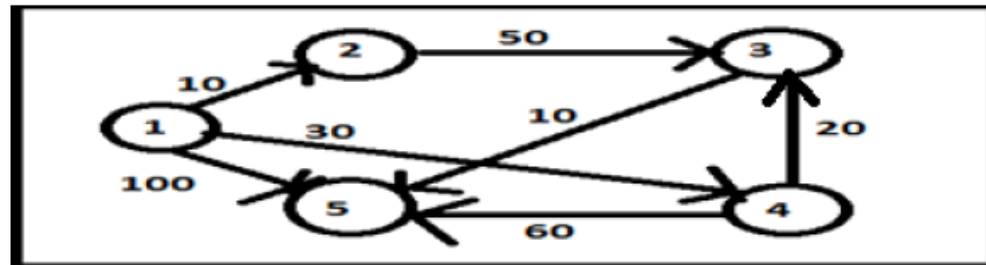
$A = \{6, 14, 3, 25, 2, 10, 20, 7, 6\}$

Old Question Papers(2017-2018)

SECTION C

3. Attempt any one part of the following: (10 x 1=10)

- (a) Explain Convex –Hull problem.
- (b) Find the shortest path in the below graph from the source vertex 1 to all other vertices by using Dijkstra's algorithm.



4. Attempt any one part of the following: (10 x 1=10)

- (a) What is backtracking? Discuss sum of subset problem with the help of an example.
- (b) Write down an algorithm to compute Longest Common Subsequence (LCS) of two given strings and analyze its time complexity.

Old Question Papers(2017-2018)

5. Attempt any one part of the following:

(10 x 1= 10)

- (a) The recurrence $T(n) = 7T(n/2) + n^2$ describe the running time of an algorithm A. A competing algorithm A' has a running time of $T'(n) = aT'(n/4) + n^2$. What is the largest integer value for a A' is asymptotically faster than A?
- (b) Discuss the problem classes P, NP and NP –complete .with class relationship.

6. Attempt any one part of the following:

(10 x 1=10)

- (a) Explain properties of Binomial Heap in .Write an algorithm to perform uniting two Binomial Heaps. And also to find Minimum Key.
- (b) Given the six items in the table below and a Knapsack with Weight 100, what is the solution to the Knapsack problem in all concepts. I.e. explain greedy all approaches and find the optimal solution

Old Question Papers(2017-2018)

ITEM ID	WEIGHT	VALUE	VALUE/WEIGHT
A	100	40	.4
B	50	35	.7
C	40	20	.5
D	20	4	.2
E	10	10	1
F	10	6	.6

7. Attempt any one part of the following: (10 x 1=10)

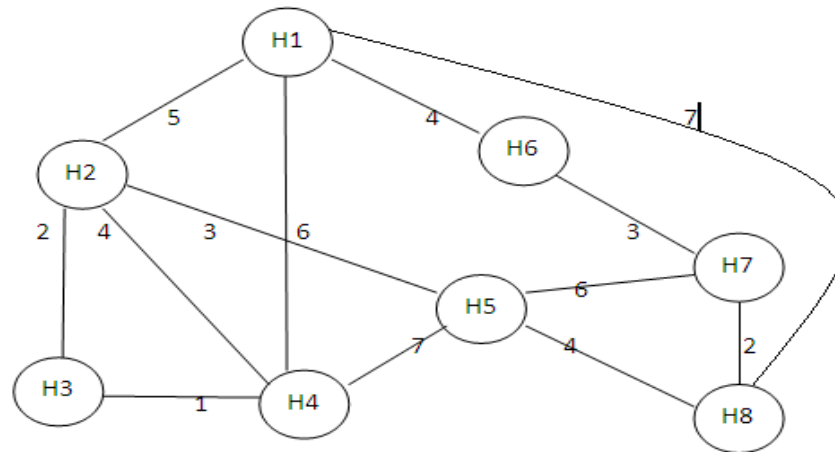
(a) Compute the prefix function π for the pattern $P = a b a c a b$ using KNUTH-MORRIS-PRATT Algorithm. Also explain Naïve String Matching algorithm.

(b) Explain Approximation and Randomized algorithms.

Expected Questions for University Exam

1. Solve the problem of travelling salesman problem on following graph and perform operation for each vertex.

[CO4]



2. Write Short notes on the following.

[CO4]

- i) Graph Coloring
- ii) Sub Set Sum Problem
- iii) Hamiltonian Cycle

Expected Questions for University Exam

3. Find out one optimal solution for following using back tracking.

4*4 chessboard,4 Queen Problem

8*8 chessboard,8 Queen Problem

[CO4]

4. Write the difference between the Greedy method and Dynamic programming.

[CO4]

5. Explain the term state space search tree with suitable example.

[CO4]

6. Explain Floyd algorithm and also analyze its complexity

[CO4]

Recap of Unit

This unit gives the insights of different techniques like Dynamic Programming, Back tracking and Branch and Bound . Different real world problem have been solved with these approaches.

The problems discussed here are Travelling salesman Problem, n Queens problem , Sum of subsets problem, Floyd Warshall algorithm, Knapsack Problem, Graph Colouring and Hamiltonian Cycle.

Unit 4

Thank You

