NAME Sidra kousar

STD CS-C DIV ROLL NO 08299221197

SUBJECT

INDEX

SR. NO.	DATE	TITLE	PAGE NO.	SIGN / REMARKS

1) What is ADA ? what is the need of to study Algorithms ? Explain in detail

Ans ADA stands for Analysis and Design of Algorithms.
The Analysis is a process of Estimating the Efficiency of an algorithms. There are two fundamental parameters based on which we can analysis the algorithm.

→ Time Space Complexity : It is a function of input size n that refers to the amount of time needed by an algorithm to run the completion.

→ Space complexity :- It can be understand as the amount of space required by an algorithm to run to completion.

Generally we make three types of analysis which is as follows

→ Wasest-case time complexity
→ Average-case time complexity
→ Best-case time complexity

Need of Algorithm

i) To understand the basic idea of the problem
ii) To find an approach to solve the problem
iii) To improve the Efficiency of existing techniques
iv) To understand the basic principles of designing the algorithm
v) To understand the flow of problem
vi) To measure the behaviours of the methods in all cases.
vii) To understand the principle of designing
viii) with the help of algorithm we can convert art into a science
ix) we can measure and analyse the complexity of problems concerning input size without implementing and running it will reduce the cost of design

2) Write all the three cases of master theorem for the equation.

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

Ans $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$f(n) = \Theta(n^k \log^p n)$

$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^k \log^p n)$

$a \geq 1$, $b > 1$, $k \geq 0$

p is a real number

case I :- if $a > b^k$ then $T(n) = \Theta(n^{\log_b a})$.

case II :- if $a = b^k$

a) if $p > -1$ then $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$

b) if $p = -1$ then $T(n) = \Theta(n^{\log_b a} \log \log n)$

c) if $p < -1$ then $T(n) = \Theta(n^{\log_b a})$

case III :- if $a < b^k$

a) if $p \geq 0$ then $T(n) = \Theta(n^k \log^p n)$

b) if $p < 0$ then $T(n) = O(n^k)$

8) What is an asymptotic notations? Give the different notations to used for represent the complexity of algorithms

Ans Asymptotic notations are the mathematics notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value

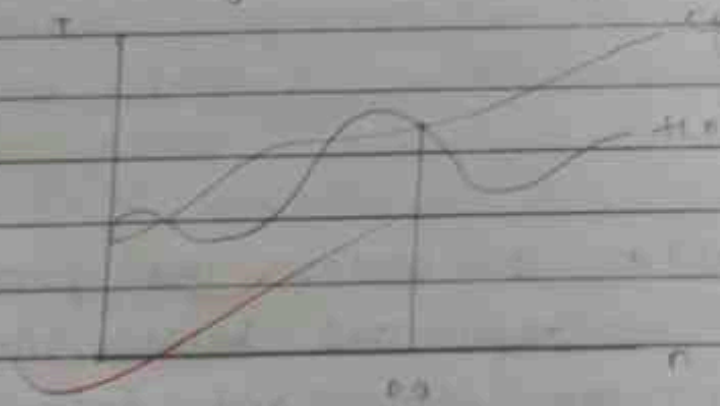
There are mainly three asymptotic notations

1) Big-Oh (O) Notation
2) Omega (Ω) Notation
3) Theta (Θ) Notation

Big-oh (O) Notation → It represents the upper bound of the running

time of an algorithm, therefore it gives the worst case complexity of an algorithm

→ It is most widely used notation for Asymptotic notation
→ It specifies the upper bound of the function
→ The maximum time required by an algorithm are worst case time complexity
→ It returns the highest possible output value (big-O) for a given input



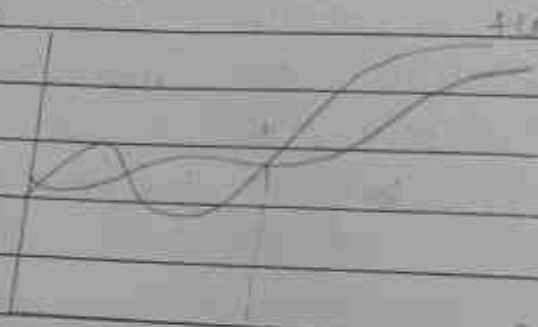
f(n) = O(g(n))

ii) Omega (Ω) :- It represent the lower bound of the running time of an algorithm, thus, it provides the best case complexity of an algorithm

→ The Execution bound serves as a lower bound on the algorithms time complexity.

→ The function $f(n) = \Omega(g(n))$ iff there Exists positive constant c and n_0.

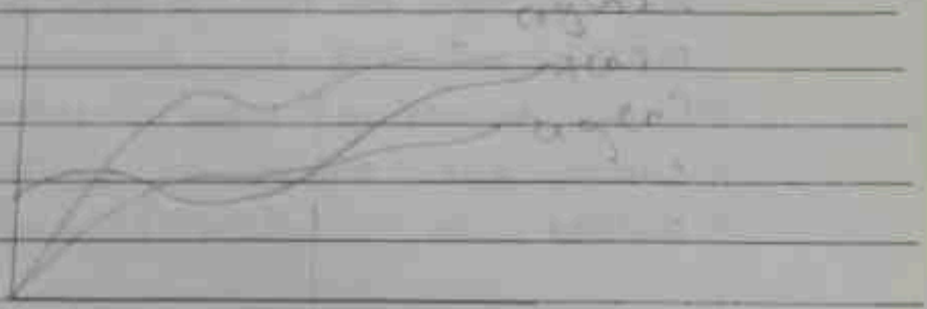
$$f(n) \geq c * g(n), \forall n, n \geq n_0$$



iii) Theta (Θ) : It Encloses the function from above and below since it represents the upper and lower bound of the running time of an algorithm. it is used for analyzing the average case complexity of an algorithm.

The function $f(n) = \Theta(g(n))$ iff there Exists positive constant c_1, c_2 and n_0.

$c_1 g(n) \leq f(n) \leq c_2 g(n)$



$\Rightarrow f(n) = \Theta(g(n))$

4) How can we prove that strassen's matrix representation is advantageous over ordinary matrix multiplication?

Ans Let A and B be two $n \times n$ matrices. The product matrix $C = AB$ is also an $n \times n$ matrix whose i^{th}, j^{th} elements is formed by taking the elements in the i^{th} row of A and j^{th} column of B and then multiplying them to get

$$C(i,j) = \sum_{1 \leq k \leq n} A(i,k)\, B(k,j) \quad \text{--- (1)}$$

$\forall$ i and j b/w 1 and n

Now to compute C11 we multiply two elements and need 8 multiplications since the matrix C has 4 elements. Thus for the resulting matrix multiplication algorithm which we refer to as the conventional method is $O(n^3)$

Suppose A and B are such partitioned into 4 square submatrices. Each sub matrix having dimensions $n/2 \times n/2$

Then product of AB can be computed by using the above formula for the product of 2×2 matrix of AB i.e

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \quad \text{--- (i)}$$

Where $C_{11} = A_{11}B_{11} + A_{12}B_{21}$

$C_{12} = A_{11}B_{12} + A_{12}B_{22}$

$C_{21} = A_{21}B_{11} + B_{22}B_{21}$

$C_{22} = A_{21}B_{12} + A_{22}B_{21}$ --- (ii)

To compute AB using Eqn (iii), we need to perform eight multiplication of $\frac{n}{2} \times \frac{n}{2}$ matrices and four additions of $n/2 \times n/2$ matrices. Since two $n/2 \times n/2$ matrices can be added in cn^2 for some constant c, the overall computing time $T(n)$ of the resulting divide and conquer algorithm is given by the recurrence

$$T(n) = \begin{cases} b & n \leq 2 \\ 8T(n/2) + cn^2 & n > 2 \end{cases}$$

where b and c are constants.

This recurrence relation can be solved as to obtain $T(n) = O(n^3)$. Hence no improvement over the conventional method has been made.

Volker Strassen discovered a way to compute Cij's of Eqn (iii)) using 7 multiplication and 8 addition or subtraction.

$$P = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$Q = (A_{21} + A_{22})B_{11}$$
$$R = A_{11}(B_{12} - B_{22})$$
$$S = A_{22}(B_{21} - B_{11})$$
$$T = (A_{11} + A_{22})B_{22}$$
$$U = (A_{21} - A_{11})(B_{11} + B_{12})$$
$$V = (A_{12} - A_{22})(B_{21} + B_{22}) \quad \text{--- (iv)}$$

After this C_{ij} can be computed using Eq^n (v)

$$C_{11} = P + S - T + V$$
$$C_{12} = R + T$$
$$C_{21} = Q + S$$
$$C_{22} = P + R - Q + U \quad \text{--- (v)}$$

The resulting recurrence relation for $T(n)$ is

$$T(n) \begin{cases} b & n \leq 2 \\ 7T(n/2) + an^2 & n > 2 \end{cases}$$

Where a and b are constants.

Now

$$T(n) = an^2\left[1 + \frac{7}{4} + \left(\frac{7}{4}\right)^2 + \cdots + \left(\frac{7}{4}\right)^{\log_2 n - 1}\right] + n^{\log_2 7} T(1)$$

$\leq cn^2 \left(\frac{7}{4}\right)^{\log_2 n} + 7^{\log_2 n}$, c a constant

$= cn^{\log_2 7} + n^{\log_2 7}$ $= O(n^{\log_2 7})$ as $c < 1$

So strassen's matrix multiplication has complexity $O(n^{2.8})$

5) Give the divide and conquer solution for quick sort and analyze its complexity.

Ans The divide and conquer approach can be used to arrive at an efficient sorting method quicksort. In quicksort, the division into two subarrays is made so that the sorted subarrays do not need to be merged later. This is accomplished by rearranging the elements in $a[1:n]$; $a[i] \leq a[j]$ for all i b/w 1 and n and all j b/w $m+1$ and n for some m, $1 \leq m \leq n$. Thus, the elements in $a[1:m]$ and $a[m+1:n]$ can be independently sorted. No merge is required. The rearrangement of the

Elements is accompolished polished by picking some Element of a[i] say t = a[s] and then reordering the other Elements so that all Elements appearing after t are greater than or Equal to t. This rearranging is known as partitioning

Time complexity of quick sort algorithm

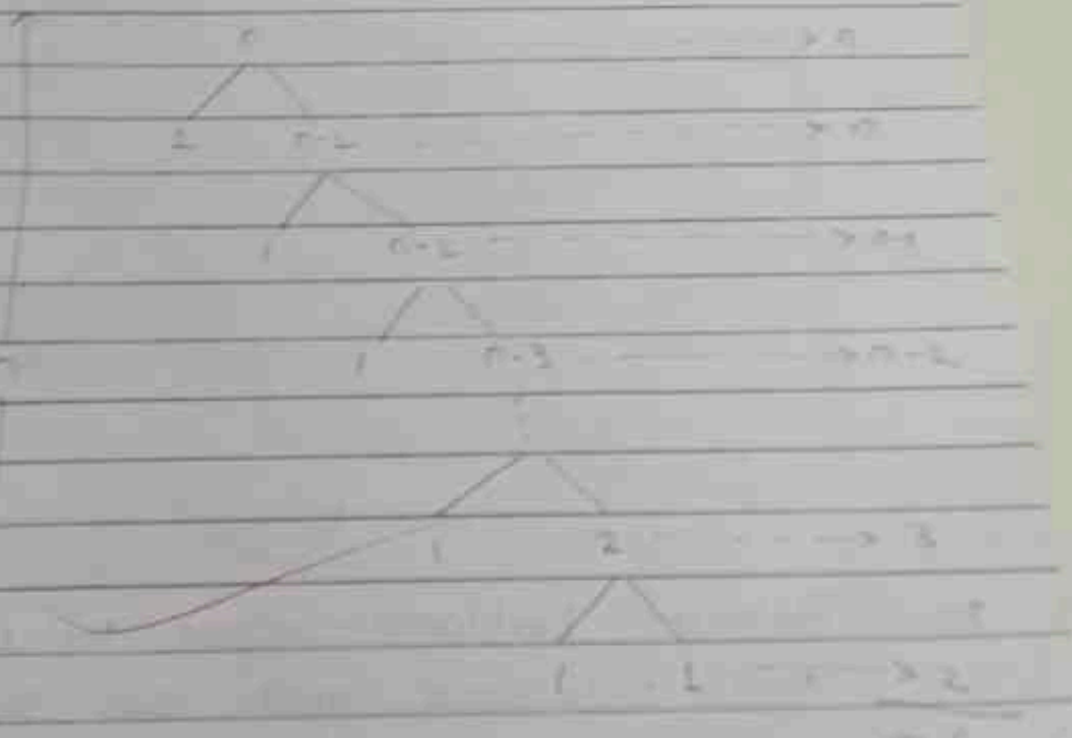
⇒ worst-case partitioning -> the worst case behaviour for quick-sort occurs when the partitioning routine produces one region with n-1 Elements and one with only 1 Element. let us assume that this unbalanced partitioning arises at every step of the algorithm

∴ partitioning cost $\Theta(n)$ time and $T(1) = \Theta(1)$, the recurrence for the running time is $T(n) = T(n-1) + \Theta(n)$
To Evaluate this recurrence, we observe that $T(1) = \Theta(1)$ and then iterate

$= \Theta(n) + \Theta(n)$

$= \Theta(n) + \sum_{k=1}^{n} \Theta(k) + \Theta(n^2)$

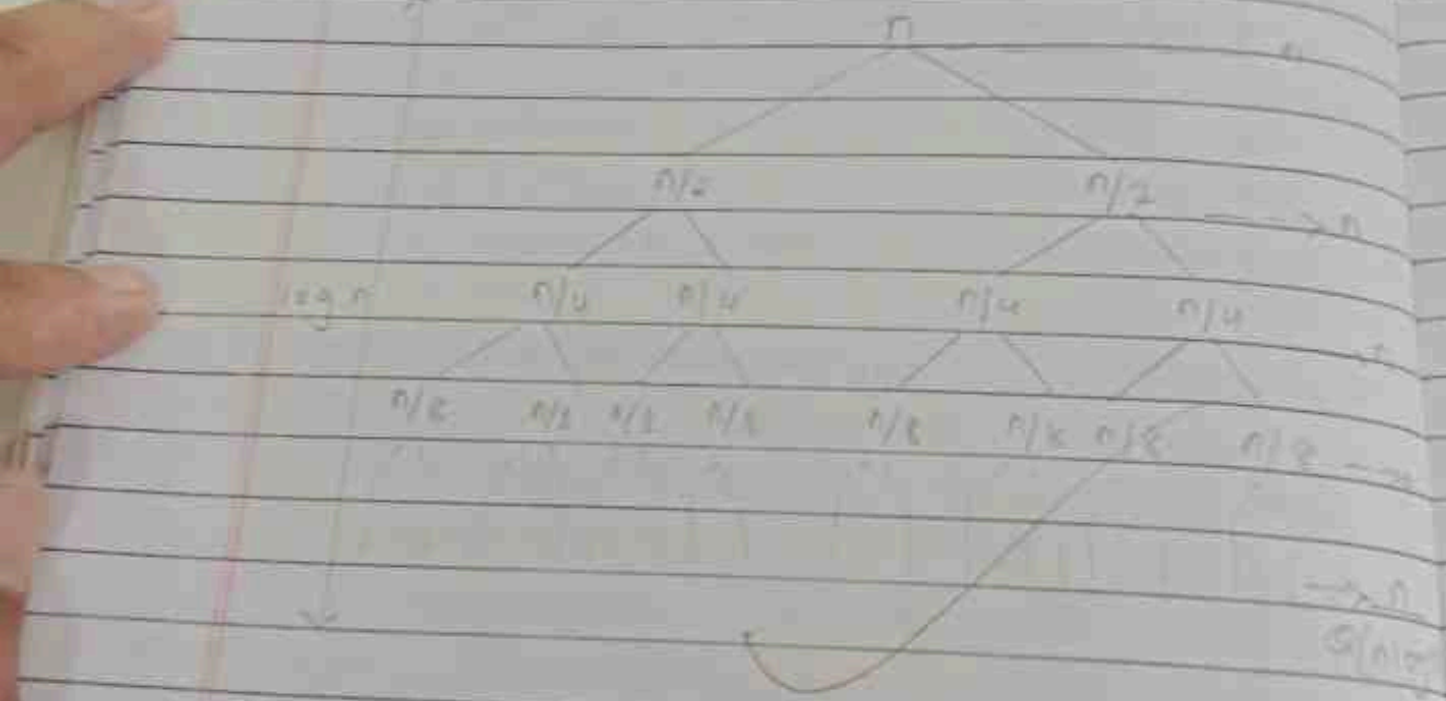
Below fig shows a recurrence tree for the worst-case execution of quick sort



Ans Best-case partitioning :- If the partitioning produces of size n/2 quick sort runs much faster the recurrence is then

$T(n) = 2T(\frac{n}{2}) + \Theta(n)$

which has solution $T(n) = \Theta(n \log n)$

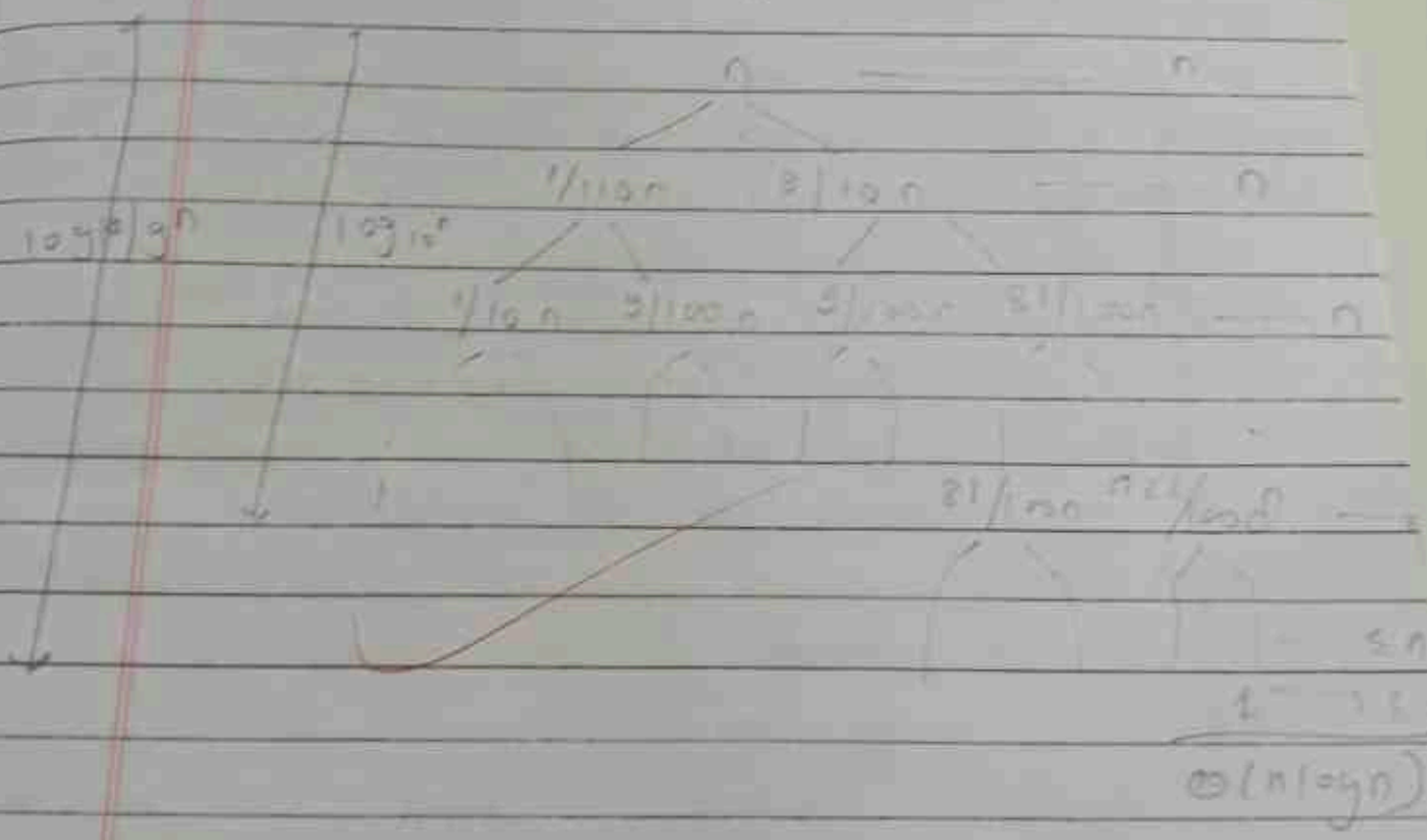


iii) Balanced partitioned - suppose the partitioning algo. propotional split then the recurrence relation is

$$T(n) = T(9n/10) + T(n/10) + n.$$

of the running time of quick sort where $\Theta(n)$ is replaced by n "c" constant

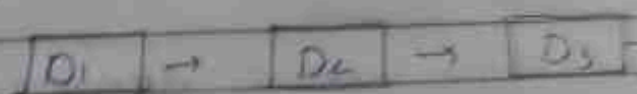
The total cost of quick sort is $\therefore$
$O(n \log n)$
Thus, with a 9 to 1 proportional split at every level of recursion, quick sort is in $O(n \log n)$ time.



Assignment - 03

1. How reliability design can be obtained using dynamic programming?

Ans The reliability design problem is to design a system that is composed of several devices connected in series



If we assign reliability as the reliability of a device

Then the reliability of a device of the function can be given as R_i

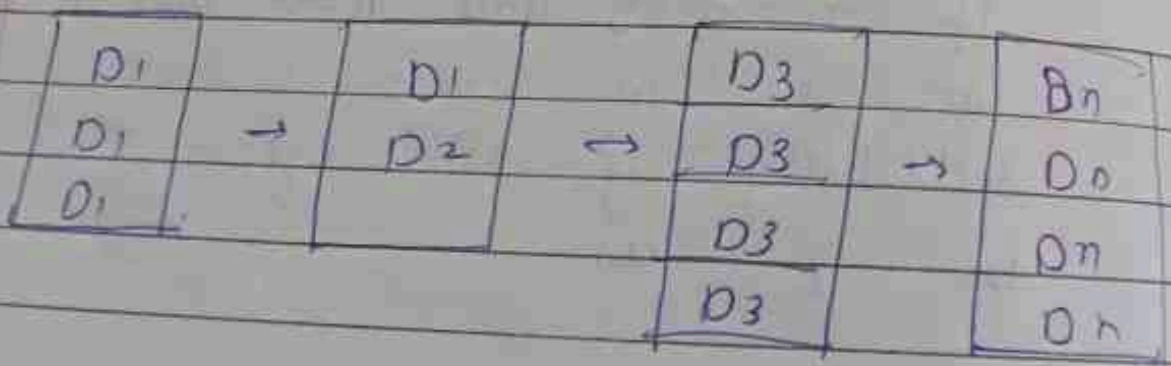
If $r_1 = 0.99$ and $n = 10$ that 10 devices are set in series $1 \le n \le 10$ then reliability of whole system π_i can be given as

$R_n = 0.90$

So, if we duplicate the device of each stage - then reliability of the system can be increased

Say, multiple couples of same device type are connected in parallel through the use of switching circuit stage

Hence switching circuit determine which device in any given group are functioning properly they are make use of such devices of each stage that result is increase in reliability at each stage



then the maximization problem can be given as

maximize $\pi_{1 \le i \le n} \phi_i(m_i)$

subject to $\sum_i (c_i m_i \le c \; (1 \le i \le n)$

$m_i \ge 1$ and integer $1 \le i \le n$

Here $\phi_i(m_i)$ denotes the reliability stage.

The reliability of system can be given $\prod_{1 \le i \le n} \phi_i(m_i)$.

If x_i increases, number of devices's at any stage beyond the cutain limit then also only cost will increases but reliability could not increase.

2 Find the optimal solution for 0/1 knapsack problem $(w_1, w_2, w_3, w_4) = (10, 15, 6, 5)$ $P_1, P_2, P_3, P_4 = \{2, 5, 8, 1\}$ and $m = 30$?

Sol Given that $n = 4$

$m = 30$

$(P_1, P_2, P_3, P_4) = (2, 5, 8, 1)$

$(w_1, w_2, w_3, w_4) = (10, 15, 6, 5)$

$S^0 = \{(0,0)\}$; $S_1^0 = \{(2,10)\}$

$S_2^1 = \{(0,0)\}$; $(2,10)$; $S_1^0 = \{(5,15), (7,25)\}$

$S_6^2 = \{(0,0)\ (2,10)\ (5,15)\ (7,25)\}$

$S_4^2 = \{(8,6)\ (10,16)\ (13,21)\ (15,31)\}$

$S_3 = \{(0,0)\ (2,10)\ (5,15)\ (7,25)\ (8,6)\ (10,6)\ (13,21)\ (15,31)\}$

Using domain once's rules we have

$S^3 = \{(0,0)\ (8,6)\ (10,16)\ (13,21)\ (15,31)\}$

$S_1^3 = \{(1,9)\ (3,15)\ (11,25)\ (14,30)\ (16,40)\}$

$S^4 = \{(0,0)\ (8,6)\ (1,9)\ (9,15)\ (11,25)\ (14,30)\ (14,40)\}$

pwaging (P, wj) with wj = m

S^0 = {(0,0) (8,6) (1,2) (9,5) (11,23) (14,20)}

Q-3 What is multistage graph problem discuss its solution based on dynamic approach also give as suitable algorithm and find its computing for

Ans The multiple stage graph problem is to find minimum cost form source to sink i.e. : a target

A multistage is a directed graph (G=(V,E)) in which the vertices are partitioned into k>2 disjoint sets Vi 1 ≤ i ≤ k

So that if there is an edge <u,v> from u to v in the u∈Vi and v∈Vi+1 for some 1≤ i≤k and sets Vi and Vk are such that |V1|=|Vk|=1

Solution based on dynamic programming

A dynamic programming for multistage graph problem is obtained by first noting that every source (s) to target (t) path is the result of a sequence of consideration which vertex v_i, i≤i≤k-2 is to be on the path

∴ c(i,i) = min

multistage graph vertex/node corresponding forward approach :

1. Algorithm FGraph (G, k, n, P)
2. // the input is a k stage graph G=(V,E)
3. // the indexed in order of stage E is a set of edges
4. // c(i,j) is the cost of <i,j> //
   [i,k] is a minimum
5. {

cost [n] := 0.0;
for j i = n-1 to 1 step -1 do
3 // compute cost [j]

9 Let r be a vertex such that <j,r>
is an edge
10 of G and c[j,d] + cost[r] is minimum
11 cost[j] := c[j,r] + cost[r];
12 d[j] := r
13 3
14 // find the minimum cost path
15 p[1] := 1; p[k] := n;
16 for i := 2 to k-1 do p[j] := d[p[j-1]]
17 3

Q-4 Explain the concept of dynamic programming with the difference dynamic programming & greedy approach

Ans. Dynamic programing like the divide and conquer method solves problems by a containing the solution to sub problems

As divide and conquer algorithm partition the problem into independent

sub problems recursively.

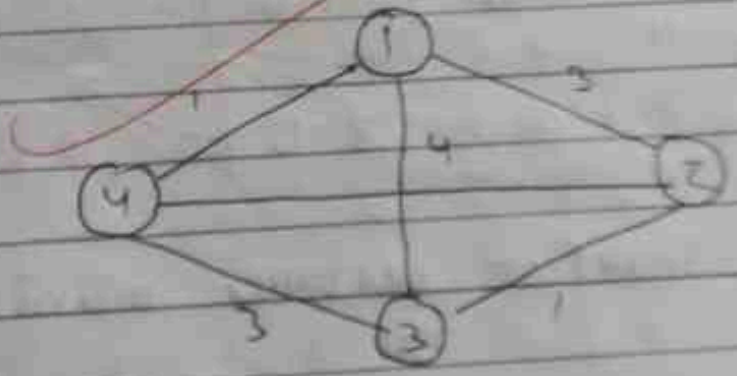
and then combine their solution to solve the original problem while in contrast dynamic programming is applicable when the sub problems are not independent that is when sub problems share sub-problem in this contrast a divide and conquer algorithm does more work than necessary repetedly & saving on common sub problems.

Difference B/W dynamic programming and greedy approach.

dynamic programming	greedy approach
i) in dynamic programming we make decision at each step considering current problem and solution to previously solved sub problem to	in greedy approach we make whatever choice seems best at the moment in the hope that it will lead to

	calculate optimum sol^n	global optimum solution.
11.	dynamic programming follows an algorithm technique	A greedy method finds the feasible solving
	0/1 knapsack problem	For a fractional knapsack

Q5 Use the floyd warshall algorithm and find shortest path b/w the all pair of vertices for the following graph



Sol

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4	∞	0	∞
4	∞	0	9	0

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4	12	0	5
4	∞	2	9	0

initial distance matrix D1

	1	2	3	4
1	0	8	9	1
2	∞	0	1	∞
3	4	12	0	5
4	∞	2	3	0

	1	2	3	4
1	0	8	5	1
2	5	0	1	6
3	4	12	0	5
4	7	2	3	0

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

Shortest distance matrix

OR

Solution:

Step 1) Remove all the self loop and parallel edges from the graph

	1	2	3	4
1	∞	1		
2	1	∞		
3	0	5		
4	9	0		

4
1
6
5
0

In the given graph, there are certain self edges as pseudo Edge.

Step 2) With the initial distance matrix D^0 represents the distances between every pair of vertices in the form of given weights. For diagonal elements distance values are

four vertices having a direct edge but then distance value = weight of that edge.

four vertices or direct edges but then distance value $= \infty$

initial distance matrix for the given

$$D^0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 8 & \infty & 1 \\ 2 & \infty & 0 & 1 & \infty \\ 3 & 4 & \infty & 0 & \infty \\ 4 & \infty & 2 & 9 & 0 \end{array}$$

Step-3 > using floyd warshall algorithm write the following matrices

$D_1 =$

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4	12	0	5
4	∞	2	9	0

$D_2 =$

	1	2	3	4
1	0	8	9	1
2	∞	0	1	∞
3	4	12	0	5
4	∞	2	3	0

$D_3 =$

	1	2	3	4
1	0	8	8	1
2	5	0	1	6
3	4	12	0	5
4	7	2	3	0

D_4 =	1	2	3	4
1	0	3	4	1
2	5	0	1	6
3	4	7	0	5
4	8	7	2	3	0

The last matrix D_4 represents shortest path distance between pair of vertices

20/6/24.

Assignment -05

Q-1 Explain Branch and bound technique for solving travelling sales problem

Ans The sales man known problem on the TSP is well, known algorithm problem in computer science the goal is to find as shortest tenture to minimize

The Branch & bound method :-

The Branch of bound method involved feeling the problem into a series of sub problem

possible solution of sub problem

→ The step involved in solving TSP asking the branch and bound technique following

→ choose a start node of them set Bound to a high value say infinity the cheapest Eaten between the current node and then add the distance to current distance

→ Repeat the procen until the current distance is less than the bounded

→ then add up to the distance so. that the bound Equals the distance

→ Repeat this procen until ~~this~~ all this are covered.

→ The node at the top of the tree is called root all Edge cistems down word

(ii) DFS and BFS graph algorithm.

	BFS	DFS
1	stand for breadth first search	stand for Depth first search
2	BFS uses queue Data structures	DFS uses stack data structure
3	BFS build tree level by level	DFS build tree subtree by subtree
4	working of FIFO	working on LIFO
5	Application is shortest path algorithm	application of a cycle graph
6	BFS traversed approach in first walk through all node	DFS also traverse approach begin at root node to nil node.

Q-3 Note on Tree Traversal

Ans Visiting Each node in a Tree are in a specific order common technique.

There are following 3 common traversal

1) In order
2) pre order
3) post order

1) In order :-

Left Root Right

2) pre order :-

Root Left Right

3) post order :-

Left Right Root

B - Tree

- Self Balance BST
- Store data in sorted manner
- In B tree there are n no. of key
- used in database and file system
- B tree no duplicate key present

Hamiltonian Cycle

- closed path in a graph
- Each vertex visited once
- finding hamiltonian cycle is NP complete

Application → Route planning, circuit design Scheduling

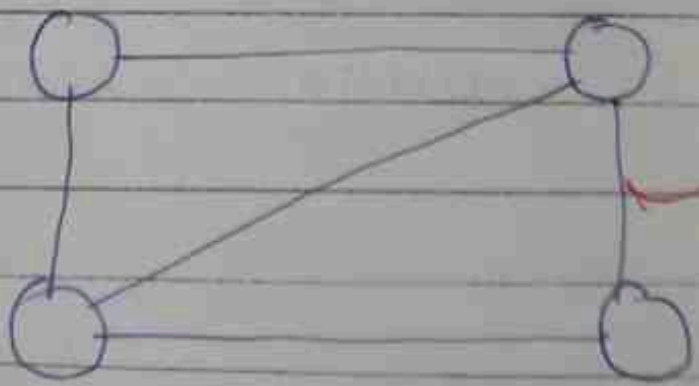
Q4 What is graph colouring problem

Ans The graph colouring problem is a graph theory problem that involving assigning.

→ colour each node one by one for first node.
→ for the Seard node you can choose
→ combine to close color for each node

lets take a example for better understanding with red, blue and green

RGB color.



n = 4, m = 3

adjacent matrix

n	0	1	2	3
0	1	1	0	1
1	1	1	1	1
2	0	1	1	1
3	1	1	1	1

C_1 = red

C_2 = green

$x(i)$ = $\in$ (green)

Returning the optimal solution of red green blue or $\{1, 0, 1, 3\}$