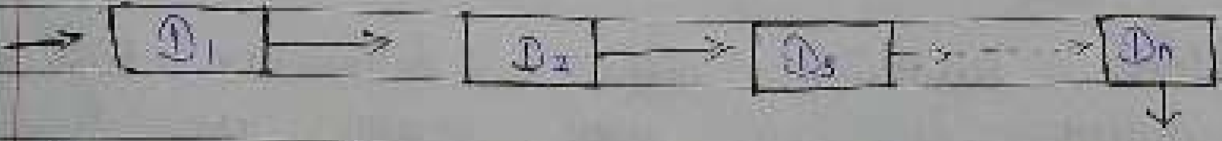
ASSIGNMENT - 02

Que 1 How reliability design can be obtained using dynamic programming :

In reliability design, the problem is to design a system that is composed of several devices connected in series.



If we imagine r_i as the reliability of a device.

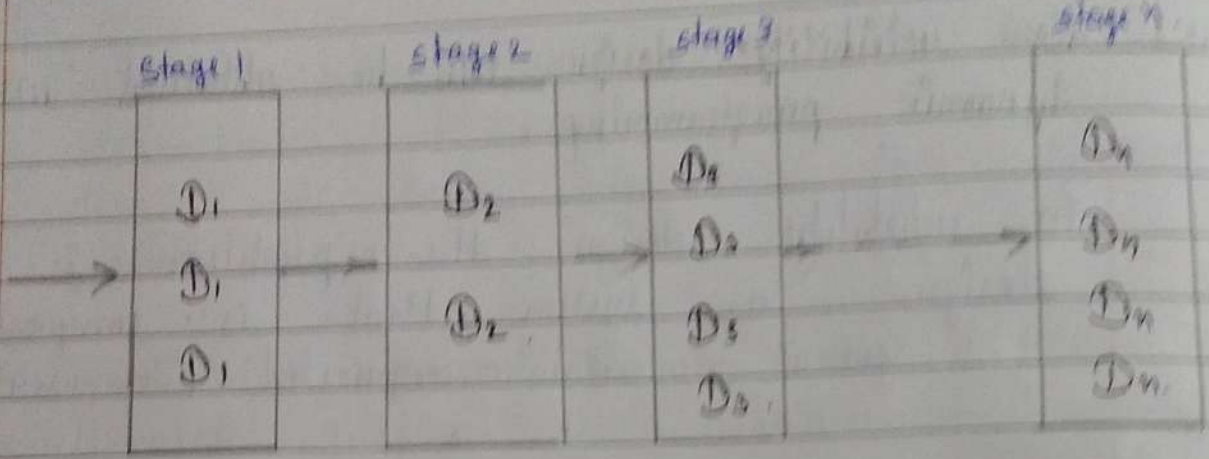
Then the reliability of a device of the function can be given as πr_i

If $r_i = 0.99$ and $n = 10$ that n device are set in a series $1 \leq i \leq 10$, then reliability of whole system πr_i can be given as

$$\pi r_i = 0.904$$

So if we duplicate the device at each stage then reliability of the systems can be increased.

Say, multiple copies of same device type are connected in parallel through the use of switching circuits.



Here, switching circuit determines which devices in any given group are functioning properly.

They make use of such devices at each stage, that result is increase in reliability at each stage.

If, at each stage, there are m similar type of device D_i, then the probability that all M_i have malfunction is $(1-r_i)^{m_i}$, which is very very less

Then the maxinization problem can be given can be given as follows -

Maximize $\Pi \, 1 \le i \le n \ \phi_i(m_i)$

Subject to $\sum_{1 \le i \le n} (c_i m_i \le c)$

$m_i \ge 1$ & integer $1 \le i \le n$.

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

The reliability of system can be given

$\prod_{1 \le i \le n} \phi_i(m_i)$.

If we increase no. of devices at any stage beyond the certain limit, then also only cost will increase but reliability could not be increase.

Que 2. Explain the concept of Dynamic programming. Write the diff. between dynamic programming and greedy approach.

Ans CONCEPT OF DYNAMIC PROGRAMMING.

(i) Dynamic programming is a name coined by Richard Bellman in 1966.

(ii) Dynamic programming is a powerful algorithm design technique that can be used when the solution to the problem may be viewed as the result of the sequence of decision.

(iii) Dynamic programming is also used in optimization problem.

Definition - A Dynamic programming algorithm solves each sub subproblem once and then saves the result in a table.

Working of Dynamic Programming -

- Identify Subproblems :- Divide the main problem into smaller, independent subproblems.
- Store Solution :- Solve each subproblem and store the solution in a table or array.
- Built up Solution :- Use the stored solutions to built up to the main problem.

- Avoid Redundancy :- By storing solutions, dynamic programming ensure that each subproblem is solved only once, reducing computation time

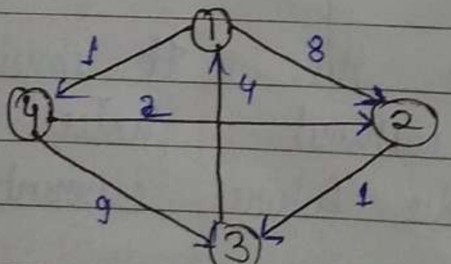
Difference Between Greedy Approach & Dynamic Programming

Greedy Approach	Dynamic Programming
* We decide based on the local solution.	We decide try to select individual in every step however the selection may rely on sol^n to subproblems.
* It doesn't guarantee optimal solution	It guarantees an optimal solution.
* It only considers the current choice without looking about the future	It considers the future choice while selecting the current choice.
* They select a choice i.e locally optimum.	It selects a choice i.e. globally optimum.
* There is no concept of result storing No extra memory is required	There is a concept of result storing so memory is required for storing the result.
* e.g fractional knapsack	e.g. 0/1 knapsack.

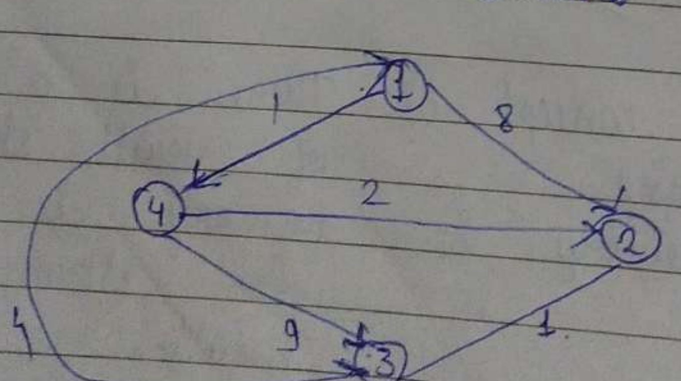
Date:
P. No:

Greedy Approach	Dynamic Programming.
* Non- overlapping subproblems	Overlapping subproblems
* It is faster than dynamic programming	Dynamic programming is comparatively slower.
* Give fast result	Gives slow result comparatively
* Every problem can't be solved by Greedy algorithm.	Every problem can be solved by dynamic programming.

Que 3 Use the Floyd - Warshall algorithm & find shortest path b/w all pair of vertices for the following graph.



Consider the follow. directed weighed graph.



Using Floyd Warshall Algorithm, finding the shortest path distance b/w every pair of vertices

Step 1:- Remove all the self loops & parallel edges (keeping the lowest weight edge) from graph.

- In the given graph, there are neither self edges nor parallel edges.

Step 2:- Write the initial distance matrix.

- It represents the distance b/w every pair of vertices in the form of given weight.
- For diagonal element (self loop), distance value = 0.
- For vertices having a direct edge b/w them, distance value = weight of the edge.
- For vertices having no direct edge b/w them, distance value = ∞.

Initial distance matrix for the given graph is :-

$$D_0 = \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & \infty & 0 & \infty \\ \infty & 2 & 9 & 0 \end{bmatrix}$$

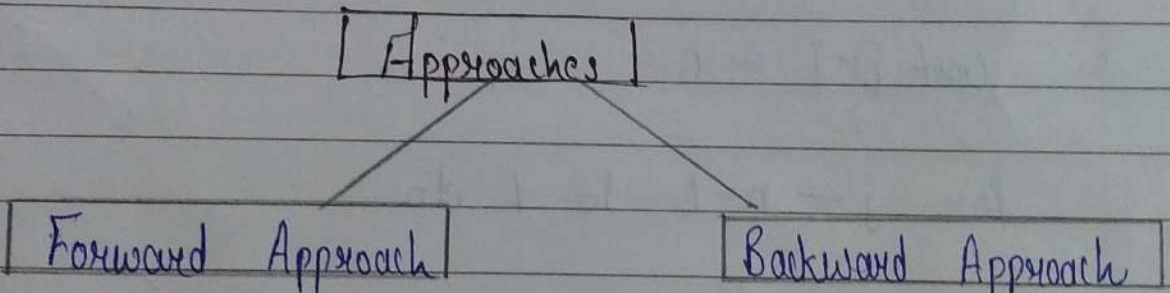
Step 3:- Using floyd Warshall Algo. write follow. 4 matrice

$$D_1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 8 & \infty & 1 \\ 2 & \infty & 0 & 1 & \infty \\ 3 & 4 & 12 & 0 & 5 \\ 4 & \infty & 2 & 9 & 0 \end{array}$$

A multistage graph $G = (V, E)$ is a directed graph $G = (V, E)$ is a directed graph in which the vertices are partitioned into $K \geq 2$ disjoint sets V_i, $1 \leq i \leq k$.

In addition, if (u, v) is an edge in E, then $u \in V_i$ and $v \in V_{i+1}$, for some i, $1 \leq i \leq k$.
The sets V_1 & V_K are such that $|V_1| = |V_K| = 1$.

There are two approaches to find the shortest path from the source node to the sink node in a multistage graph.


Approaches
Forward Approach
Backward Approach

Forward Approach :-

→ Here, we assume that there are k stages in the graph.

→ We start from last stage & find out the cost of each & every node to the first stage

→ We then find out the minimum cost path from the source to destination. (i.e. from 1 to k stage)

Algorithm :-

```
MULTI - STAGE (G, k, n, p)

// Description : solving multi-stage problem using Dynami
                                                  programming
// Input :

k : Number of stages in Graph G = (V, E)

c[i, j] : Cost of edge (i, j)

// Output : p[1 : k] : Minimum cost path

  cost [n] ← 0

  for j ← n-1 to 1 do

  // Let r be a vertex such that (j, r) E & c[j, r]
                 + cos[r] is minimum.

  cost [j] ← c[j, r] + cost [r].

    π [j] ← r.

end
// Find minimum cost path.
    p [1] ← 1.

    p [k] ← n.
```

```
for j ← 2 to k-1 do.
    p[j] ← π[p[j-1]]
end.
```

⊛ Backward Approach :-

If there are 'K' stages in a graph using backward approach.

→ We will find out the cost of each & every vertex starting from 1st stage to kth stage.

→ We will find out the minimum cost path from destination to source (i.e. from k to 1 stage)

Algorithm :-

```
Algorithm BGraph (G, k, n, p)
// Same Function as F Graph
{
 Bcost[1] = 0.0;

 For j = 2 to n do
 {
 // compute b cost[j]
 Let r be such that is an edge of
         G and b cost[r] + c[r, j];

   D[j] = r;
 }
// find minimum cost path.
    p[1] = 1;          p[k] = n;
    For j = k-1 to 2 do
        p[j] = d[p[j+1]];
    }.
```

Time Complexity of Multistage Graph

$= O(n^2)$

Ques 5. Find the optimal solⁿ for 0/1 knapsack problem $(W_1, W_2, W_3, W_4) = \{10, 15, 6, 9\}$, $(P_1, P_2, P_3, P_4) = \{2, 5, 8, 1\}$ + $M = 30$

i	P_i	W_i
1	2	10
2	5	15
3	8	6
4	1	9

$n = 4$

$M = 30$

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

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

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

$S^2_1 = \{(8,6)\ (10,16)\ (13,21)\ (15,31)\}$

$S^3 = \{(0,0)\ (2,10)\ (5,15)\ (7,25)\ (8,6)\ (10,16)$
$(13,21)\ (15,31)\}$

Using domain ance rule, we have:

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

$S^3_1 = \{(1,9)\ (9,15)\ (11,25)\ (14,30)\ (16,40)\}$

$S^4 = \{(0,0)\ (8,6)\ .(1,9)\ .(9,15)\ (11,25)\ (14,30)\ (16,40)\}$

Purging $(P_i\ w_i)$ with $w_i > M$

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

19/4/24