ADA4

UNIT - IV

DYNAMIC PROGRAMMING

- The general method
- Warshall's Algorithm
- Floyd's Algorithm for all pair shortest path
- Single source shortest paths : general weights
- 0/1 Knapsack
- Travelling Salesperson problem

What is Dynamic Programming ? Explain

Dynamic Programming is one of the most efficient programming technique. It was invented by Richard Bellman.

Dynamic Programming is used to mainly solve such problems which have overlapping sub-problems. It makes use of a recurrence relation.

Dynamic programming suggests solving each of the smaller subproblems only once and records the results in a table from which we can then obtain a solution to the original problem.

One variation of the dynamic programming approach seeks to avoid solving un-necessary subproblems by using the concept of memory functions.

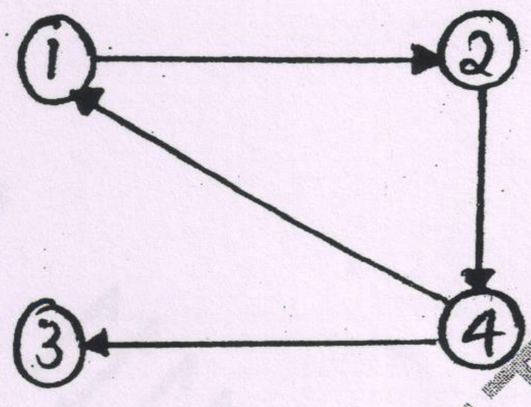
Some of the problems that can be solved using Dynamic Programming technique is -

- Warshall's Algorithm
- Floyd's Algorithm
- knapsack Problem etc.

ADA4

Write a note on Warshall's Algorithm.

Warshalls Algorithm makes use of the concept of Dynamic Programming. It is used to compute the transitive closure of the given graph. It accepts adjacency matrix as its input and generates path matrix as its output. Consider the graph



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

adjacency matrix

⇒ Warshall's Algorithm ⇒

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

Path matrix

The formula for computing transitive closure is as given below

$$r_{ij}^{k} = r_{ij}^{k-1} \text{ or } \left(r_{ik}^{k-1} \text{ and } r_{kj}^{k-1} \right)$$

The algorithm is as given below-

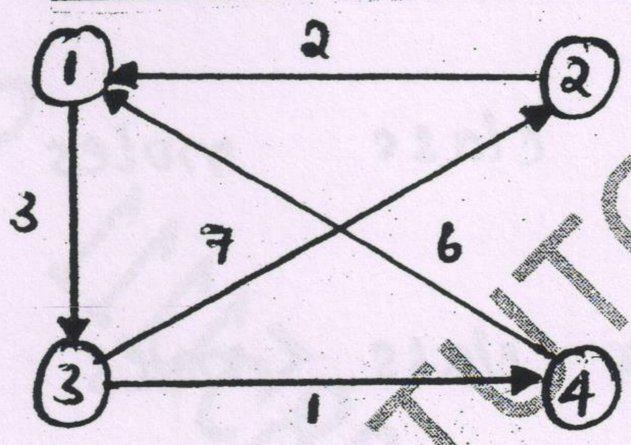
TRACE - Refer class notes

ANALYSIS - Refer class notes

ADA4

Write a note on Floyd's Algorithm.

Floyds Algorithm makes use of the concept of Dynamic Programming. It is used to compute the shortest distance between every a pair of nodes in a given graph and hence it is also called as the "all-pair-shortest-path" algorithm. Consider the graph



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix} \end{array} \Rightarrow \boxed{\text{FLOYD'S ALGORITHM}} \rightarrow \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 7 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix} \end{array}$$

The formula for Floyd's algorithm is -

$$d_{ij} = w_{ij} \quad \text{if } k = 0$$

$$d_{ij}^{k} = \min\left(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\right) \quad \text{if } k \geq 1$$

Floyd's Algorithm is as shown below -

TRACE - Refer class notes

ANALYSIS - Refer class notes

ADAB

Write a note on the Knapsack Problem.

Knapsack Problem can be solved by making use of the Dynamic programming approach.

Knapsak problem is such a problem in which we have "n" items and the weights of each item is "w_1 w_2 .... w_n" and the values associated with each item is "v_1 v_2 .... v_n". We must find the most valuable set of items that fit into the knapsack.

The recurrence relation related to the knapsack problem is -

$$V_{ij} = \begin{cases} V_{i-1,j} & \text{if } j - w_i < 0 \\ \max\left[(V_{i-1,j}), (p_i + V_{i-1,j-w_i})\right] & \text{if } j - w_i \geq 0 \end{cases}$$

P.T.O

The Algorithm for knapsack is as shown below -

TRACE - Refer class notes

ANALYSIS - Refer class notes