9/2/2020 UNIT-III Dynamic Programming →

→Dynamic programming is typically applied to optimization problem. In the world Dynamic programming, the word programming stands for planning.

→Dynamic prog is a technique for solving the problems with overlapping subproblems.

→In this method each sub problem is solved only once. The result of each sub problem is recorded in a table form from which a solution to the original problem is obtained.

General Method :- (2M or 3M)

• PRINCIPALE OF OPTIMALITY :-

The dynamic programming algorithm obtains the solution using principle of optimality.

The principle of optimality states that in an optimal sequence of decisions (or) choices each subsequence must also be optimal.

Diff b/w divide & conquer & Dynamic prog :-

• STEPS OF DYNAMIC PROGRAMING :-

→ Dynamic prog design involves 4 major Steps:

1) Characterize the structure of optimal solution.

2) Recursively define the value of an optimal solution.
3) By using bottom up technique compute the value of optimal solution.
4) Compute an optimal soln from computed information.

→ APPLICATIONS OF DYNAMIC PROGRAMMING →

1) Matrix chain Multiplication.
2) Optimal Binary search trees.
3) 0/1 Knapsack problem
4) all pairs shortest path problem.
5) Travelling sales person problem.
6) Reliability design.

① Matrix chain Multiplication

Ex: $A_1 = 5 \times 4$ (P_1), $A_2 = 4 \times 6$ (P_2), $A_3 = 6 \times 2$ (P_2), $A_4 = 2 \times 7$ (P_4)

$m(1,2)$
$5 \times 4 \times 4 \times 6$
$5 \times 4 \times 6 = 120$

$m(2,3)$
$4 \times 6 \times 6 \times 2 = 4 \times 6 \times 2 = 48$

$m(3,4) = 6 \times 2 \times 2 \times 7$
$= 6 \times 2 \times 7 = 84$

cost table

	1	2	3	4
1	0	120	88	158
2		0	48	104
3			0	84
4				0

sequence table

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

$A_1 | A_1 \times A_2 | A_1 \times (A_2 \cdot A_3)$

$m(1,3) \rightarrow A_1, A_2, A_3$
$A_1 \times (A_2 \cdot A_3)$
$m(1,1) + m(2,3) + 5 \times 4 \times 4 \times 2$
$= 0 + 48 + 40 = 88$ ✓ least

$(A_1 \times A_2) \cdot A_3$
$= 120 + 0 + 5 \times 6 \times 2$
$= 120 + 60 = 180$

m(2,4) A_2, A_3, A_4

$A_2 \times (A_3 \cdot A_4)$

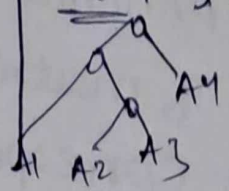
m(2,2) + 84 + 4x6x7

= 0 + 84 + 168

= 252

$(A_2 \cdot A_3) \times A_4$

104 → least.



m(1,4) $A_1, A_2 A_3 A_4$

$(A_1 \times (A_2 \times A_3)) \times A_4$

m(1,3) + m(4,4) + 5x2x2x7

88 + 0 + 5x2x7

88 + 70 = 158 (least)

$(A_1 \cdot A_2) \times (A_3 \cdot A_4)$

m(1,2) + m(3,4) + 5x6x6x7

= 120 + 84 + 5x6x7

= 204 + 210

= 414

$A_1 \times ((A_2 \times A_3) \times A_4)$

m(1,1) + m(2,4) + 5x4x7

= 0 + 104 + 140

= 244

Input :- n matrices $A_1, A_2 \ldots A_n$ n dimensions $P_1 \times P_2, P_2 \times P_3 \ldots P_n \times P_{n+1}$ respectively.

Goal :- To compute the matrix product $A_1, A_2 \cdots A_n$

Problem :- In what order should $A_1, A_2 \cdots A_n$ be multiplied so that it would take the minimum number of computations to derive the product.

To solve this problem using dynamic programming the following steps are performed

Step 1 : Let Mij denote the cost of multiplying $A_i \cdots A_j$, where the cost is measured in number of scalar multiplications.

$M(i,i) = 0, \forall i$ and $M(1,n)$ is required Solution.

Step-2 : The seq of decisions can be build using the principle of optimality.

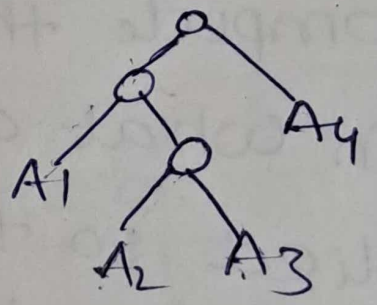
* Consider process of matrix chain multiplication

Let 'T' be the tree corresponding to optimal way of multiplying $A_i \cdots A_j$. T has a left subtree 'L' & right subtree 'R', L-corresponds to multiplying $A_1 \cdots A_k$ & R corresponds to multiplying $A_{k+1} \cdots A_j$.

Step 3:- Apply the foll for computing each sequence

$$M_{ij} = \text{Min}\{M_{i,k} + M_{k+1,j} + P_i P_{k+1} P_j \mid i <= k <= j-1\}$$

Tree : $(A_1 \times (A_2 \cdot A_3)) A_4$



→ Optimal Binary search tree :-

Problem Description :- Let $A_1, A_2, \ldots A_n \Rightarrow \{a_1, a_2 \ldots a_n\}$ be a set of identifiers such that $a_1 < a_2 < a_3 \ldots$ Let $P(i)$ be the probability with which can search for a_i & $q(i)$ be the probability of searching element x such that $a_i < x < a_{i+1}$ and $0 \le i \le n$ i.e $p(i)$ is the probability of success search and $q(i)$ is the probability of unsuccessful search. Also $\sum_{1 \le i \le n} P(i) + \sum_{1 \le i \le n} q(i)$, then obtain a tree with minimum cost. such a tree with optimum cost is called "optimal binary search tree".

→ TO solve this problem using dynamic programming the foll steps will be performed

STEP-1 :- Notations used in this algo are

Let $T_{ij} = OBST(a_{i+1} \ldots a_j)$

$C_{ij} = cost(T_{ij})$

$W_{ij} =$ weight of each T_{ij}

$T_{0n} =$ final tree obtained

$T_{00} =$ Empty.

$T_{i,j+1} =$ single node tree that has element a_{i+1}.

- During the computations the root values are computed & r_{ij} stores root value of T_{ij}.

STEP-2 :-
The OBST can be build using the principle of optimality.

The process of creating the OBST is

*Let Ton be an OBST for the elements $a_1 < a_2 < a_3 \cdots < a_n$ and let L and R, be its left subtree and right subtree, the root of Ton is a_k for some k.

*The elements in the left subtree L are $a_1, a_2 \cdots a_{k-1}$ and the elements in the right subtree R are $a_{k+1}, a_{k+2} \cdots a_n$

*The cost of computing the Ton can be given as $C(Ton) = C(L) + C(R) + P_1 + P_2 \cdots + P_n + q_0 + q_1 + \cdots q_n$

i.e $C(Ton) = C(L) + C(R) + w$

where $w = \underbrace{P_1 + P_2 + \cdots P_n}_{\text{successful search}} + \underbrace{q_0 + q_1 + \cdots + q_n}_{\text{unsuccessful search}}$

STEP-3 :-
Apply the following formula for computing the sequence

$$C(i,j) = \left\{ i < k \overset{\min}{\leq} j \; \{ C(i,k-1) + C(k,j) \} + w(i,j) \right\}$$

$$w(i,j) = w[i,j-1] + P[j] + q[j];$$

$$r[i,j] = k$$

$w_{i,i} = q_i$
$r_{i,i} = 0$
$c_{i,i} = 0$

$w_{i,i+1} = q_i + q_{(i+1)} + P_{(i+1)}$
$r_{i,i+1} = i+1$
$c_{i,i+1} = q_i + q_{(i+1)} + P_{(i+1)}$
$w_{i,j} = w_{i,j-1} + P_j + q_j$
$r_{i,j} = k.$

*Ex:- consider n=4 and (q_1, q_2, q_3, q_4) = (do, if, int, while). The values of P's and q's given as P(1:4) = (3,3,1,1) and q(0:4) = (2,3,1,1,1). Construct a optimal binary search tree.

∧:- Construct the tables for values of w, c & r

$P_1 = 3$ $q_0 = 2$
$P_2 = 3$ $q_1 = 3$
$P_3 = 1$ $q_2 = 1$
$P_4 = 1$ $q_3 = 1$
$q_4 = 1$

Let i=0 $\boxed{w_{i,i} = q_i}$

$w_{ii} = w_{00} = q_0 = 2$
i=1 $w_{11} = q_1 = 3$
i=2 $w_{22} = q_2 = 1$
i=3 $w_{33} = q_3 = 1$
i=4 $w_{44} = q_4 = 1$

$\boxed{w_{i,i+1} = q_i + q_{i+1} + P_{i+1}}$

$w_{01} = q_0 + q_1 + P_1 = 8$
$w_{12} = q_1 + q_2 + P_2 = 7$
$w_{23} = q_2 + q_3 + P_3 = 3$
$w_{34} = q_3 + q_4 + P_4 = 3$

$\boxed{w_{i,j} = w_{i,j-1} + P_j + q_j}$

$w_{0,2} = w_{0,1} + P_2 + q_2$
$= 8 + 3 + 1 = 12$
$w_{1,3} = w_{1,2} + P_3 + q_3$
$= 7 + 1 + 1 = 9$
$w_{2,4} = w_{2,3} + P_4 + q_4$
$= 3 + 1 + 1 = 5$
$w_{0,3} = w_{0,2} + P_3 + q_3$
$= 12 + 1 + 1 = 14$
$w_{1,4} = w_{1,3} + P_4 + q_4$
$= 9 + 1 + 1 = 11$

i, j-1

	0	1	2	3	4
0	w_{00} =2	w_{11} =3	w_{22} =1	w_{33} =1	w_{44} =1
1	w_{01} =8	w_{12} =7	w_{23} =3	w_{34} =3	
2	w_{02} =12	w_{13} =9	w_{24} =5		
3	w_{03} =14	w_{14} =11			
4	w_{04} =16				

$w_{0,4} = w_{0,3} + P_4 + q_4$
$= 14 + 1 + 1$
$= 16$

compute the values of c and r.

$c_{i,i+1} = q_i + q_{i+1} + P_{i+1}$

$c_{0,1} = q_0 + q_1 + P_1$
$= 2+3+3 = 8$

$r_{i,i+1} = i+1$

$r_{01} = 0+1 = 1$

$c_{00}=0$ $r_{00}=0$	$c_{11}=0$ $r_{01}=0$	$c_{22}=0$ $r_{22}=0$	$c_{33}=0$ $r_{33}=0$	$c_{44}=0$ $r_{44}=0$
$c_{01}=8$ $r_{01}=1$ ✓	$c_{12}=7$ $r_{12}=2$	$c_{23}=3$ $r_{23}=3$	$c_{34}=3$ $r_{34}=4$	
$c_{02}=19$ $r_{02}=1$	$c_{13}=12$ $r_{13}=2$	$c_{24}=8$ $r_{24}=3$		
$c_{03}=25$ $r_{03}=2$	$c_{14}=19$ $r_{14}=2$			
$c_{04}=32$ $r_{04}=2$				

$i < k \le j$

$c_{12} = q_1 + q_2 + P_2$
$= 3+1+3 = 7$

$c_{23} = q_2 + q_3 + P_3$
$= 1+1+1 = 3$

$c_{34} = q_3 + q_4 + P_4$
$= 1+1+1 = 3$

0,1,2
i k j
$k \le j$

$r_{12} = i+1 = 1+1 = 2$
$r_{23} = 3$
$r_{34} = 4$

Now compute $c_{i,j}$ and $r_{i,j}$ for $j-i \ge 2$. Hence

$c_{i,j} = \min_{i<k\le j} \{ c(i,k-1) + c(k,j) \} + w(i,j) \}$

for c_{02} : $i=0, j=2$ then $r_{i,j-1}$ to r_{i+1} to j then

r_{01} & $r_{1,2}$ $\Rightarrow k = 1, 2$ $(i < k \le j)$

$r_{01} = 1$
$r_{12} = 2$

value of $k=1$,

And compute c_{ij}. Similarly with $k=2$ compute c_{ij} and pick up min value of $c_{i,j}$

$k=1$

$c_{ij} = \{ c(i,k-1) + c(k,j) \} + w(i,j)$

$c_{02} = c(0,0) + c_{1,2} + w(0,2)$

$= 0 + 7 + 12 = 19$ ✓

$k=2$

$c_{02} = c(0,1) + c_{2,2} + w(0,2)$

$= 8 + 0 + 12 = 20$

$r=1$

$k = 2,3 \quad C_{ij} = C_{i,k-1} + C_{k,j} + w_{ij}$

$C_{13} = C_{1,1} + C_{2,3} + w_{1,3}$

$k=2 \quad = 0 + 3 + 9 = 12$ ✓

$C_{13} \Rightarrow C_{1,2} + C_{3,3} + w_{13}$

$k=3 \quad = 7 + 0 + 9 = 16.$

$2 \le k \le j$

$k = 3,4$

$C_{24} \Rightarrow C_{2,2} + C_{3,4} + w_{24}$

$k=3$

$= 0 + 3 + 5 = 8$ ✓

$k=4 \Rightarrow C_{2,3} + C_{4,4} + w_{24}$

$= 3 + 0 + 5 = 8$

$k = 1,2,3$

$C_{03} \Rightarrow C_{0,0} + C_{13} + w_{03}$

$k=1 \quad = 0 + 12 + 14 = 26$

$k=2 \quad C_{0,1} + C_{23} + w_{03}$

$= 8 + 3 + 14 = 11 + 14 = 25$ ✓

$k=3 \quad = C_{0,2} + C_{33} + w_{03}$

$= 19 + 0 + 14 = 33$

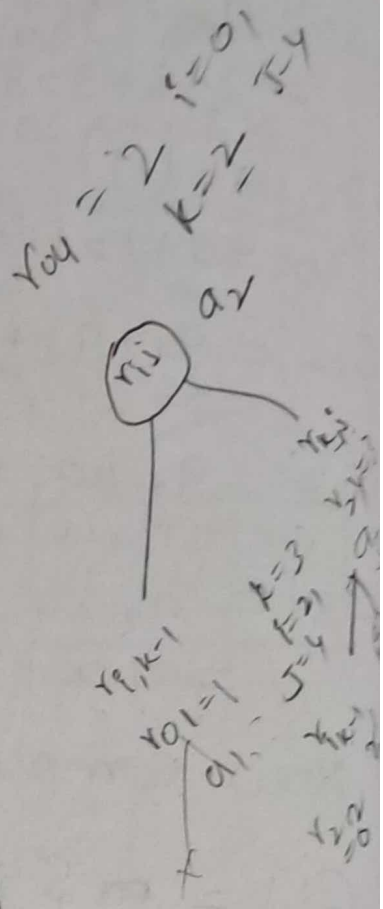
$1 < k < 4$

$C_{14} \Rightarrow C_{11} + C_{24} + w_{14}$

$k=2 \quad 0 + 8 + 11 = 19$ ✓

$k=3 \Rightarrow C_{12} + C_{34} + w_{14} = 7 + 3 + 11 = 21$

$k=4 \Rightarrow C_{13} + C_{44} + w_{14} = 12 + 0 + 11 = 23$

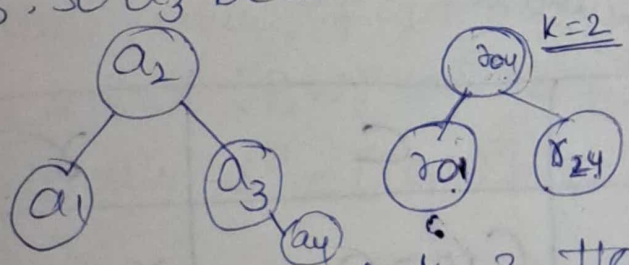


$C_{04} \Rightarrow$ $C_{00} + C_{14} + w_{04}$
$k=1,2,3,4.$ $= 0 + 19 + 16 = 35.$
$k=1$

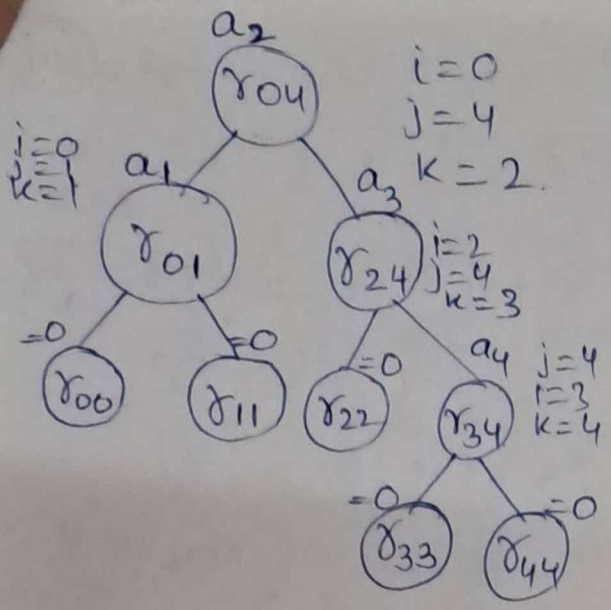
$k=2 \Rightarrow C_{01} + C_{24} + w_{04} = 8 + 8 + 16 = 16+16 = 32$ ✓
$k=3 \Rightarrow C_{02} + C_{34} + w_{04} = 19 + 3 + 16 = 38$
$k=4 \Rightarrow C_{03} + C_{44} + w_{04} = 25 + 0 + 16 = 41$
$r_{04} = 2$ k

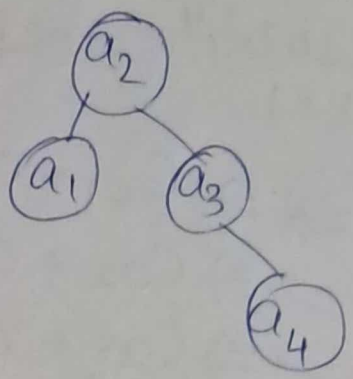
∴ T_{04} has a root r_{04}. The value of r_{04} is 2
from $(a_1, a_2, a_3, a_4) = (do, if, int, while)$
a_2 becomes root node $\underline{r_{ij} = k}$
a_k $\underline{r_{04} = 2}$.

then $r_{i,k-1}$ becomes the left child of r_{04} and r_{kj} becomes the right child i.e q_{01} becomes left child & r_{24} becomes right child.
Here, $r_{01} = 1$. So, a_1 becomes left child of a_2 and $r_{24} = 3$. So a_3 becomes right child of a_2

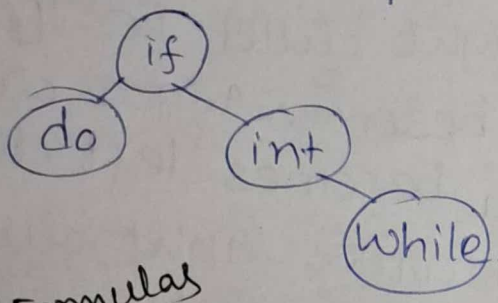


For r_{34} $i=2, j=4$ and $k=3$. Hence left child of it is $r_{i,k-1} = r_{22} = 0$ that means the left child of $r_{34} = a_3$ is empty. The right child of r_{24} is $r_{kj} \Rightarrow r_{34} = 4$. Hence a_4 becomes the right child of a_3.
Thus all $n=4$ nodes are used to build a binary search tree.



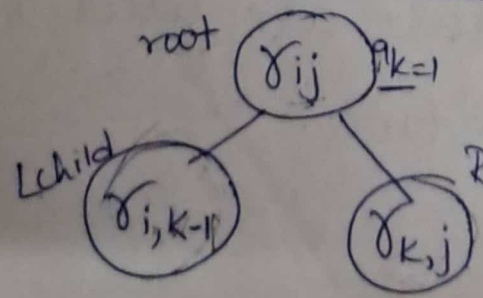


∴ The optimal binary search tree is



Remembers Formulas

w	c	r
$w_{ii} = q_i$	$c_{ii} = 0$	$r_{ii} = 0$
$w_{i,i+1} = q_i + q_{i+1} + p_{i+1}$	$c_{i,i+1} = q_i + q_{i+1} + p_{i+1}$	$r_{i,i+1} = i+1$
$w_{ij} = w_{i,j-1} + p_j + q_j$	$c_{ij} = \min_{i<k\le j} \{c_{i,k-1} + c_{kj}\} + w_{ij}$	$r_{i,j} = k$



0/1 Knapsack problem:-

*problem description:- If there are n objects given and a knapsack (or) a bag in which the object i that has weight w_i is to be placed. the knapsack has a capacity 'W' then the profit that can be earned is $P_i x_i$. The objective of the problem is to obtain filling of knapsack with maximum profit earned i.e maximized $\sum_{n} P_i x_i$ subject to constraint $\sum_{n} w_i x_i \leq W$ where $1 \leq i \leq n$ and n is ~~total~~ number of objects and $x_i = 0$ (or) 1.

To solve this problem using Dynamic programming method, the following steps can be performed.

STEP-1:- The notations are used. Let $f_i(Y_i)$ be the value of optimal solution then S^i is a pair (P, W) where $P = f(Y_j)$ and $W = Y_j$. Initially $S^0 = [(0,0)]$. compute $S^{i+1} = S^i$

STEP-2:- Let x_n be the optimal sequence. Then there are 2 instances $\{x_n\}$ and $\{x_{n-1}, x_{n-2} \cdots x_1\}$ So, from $\{x_{n-1}, x_{n-2} \cdots x_1\}$ will choose the optimum sequence with respect to x_n.

→ The selection of sequence from remaining set should be able to fulfill the condition of filling knapsack of capacity W with maximum profit. This proves that 0/1 knapsack problem is solved by using

principle of optimality.

STEP-3 :- The formulae that are used while solving 0/1 knapsack is let $f_i(Y_j)$ be the value of optimal solution, then $f_i(Y) = \max\{f_{i-1}(Y), f_{i-1}(Y-w_i)+P_i\}$.

Initially, $S^0 = \{(0,0)\}$. $S_i^i = \{(P,W) / ((P-P_i),(W-w_i)) S_j^i\}$

S_1^{i+1} can be computed by merging s^i and s_1^i.

→ purging rule :- (2M)

If S^{i+1} contains (P_j, W_j) and (P_k, W_k); these two pairs such that $P_j \le P_k$ and $W_j \ge W_k$, then (P_j, W_j) can be eliminated. ~~(OR)~~ this rule is called as purging rule (OR) dominance rule.

* Solve the knapsack instance $M=6$, and $n=3$, let P_i and w_i are as shown below.

i	P_i	w_i
1	1	2
2	2	3
3	5	4

A:- Let, build the sequence of decision S^0, S^1, S^2

$S^0 = \{(0,0)\}$ initially

$S_1^0 = \{(1,2)\}$. $S^1 = \{$ merge S^0 and $S_0^1\} = \{(0,0),(1,2)\}$

$S_1^1 = \{$ select next (P,W) pair and add it with $S^1\}$

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

$= \{(2,3),(2,3),(3,5)\} \Rightarrow \{(2,3),(3,5)\}$

$S^2 = \{$ merge s^1 and $s_1^1\}$

$= \{(0,0),(1,2),(2,3),(3,5)\}$

S_1^2 = {select next (P,W) and add it with S^2}

= {(5,4), (5,4), (6,6), (7,7), (8,9)}

= {(5,4), (6,6), (7,7), (8,9)}

S^3 = {merge S^2 and S_1^2}

= {(0,0), (1,2), (2,3), (3,5), (5,4), (6,6), (7,7), (8,9)}

$P_j \leq P_k$ and $W_j \geq W_k$.

By applying purging rule on (3,5), (5,4)

$W_j \geq W_k$	$P_j \leq P_k$
$5 \geq 4$	$3 < 5$ ✓

So Remove $(P_j, W_j) \Rightarrow$ (3,5)

S^3 = {(0,0), (1,2), (2,3), (5,4), (6,6), (7,7), (8,9)}

As M=6, find the tuple (or) order pair, denoting the weight 6 i.e (6,6) belongs to S^3. Hence will set $x_3 = 1$ (x_3 = (5,4) placed fully in the knapsack). Now the balance in the knapsack is $(6-P_3, 6-W_3) \Rightarrow (1,2)$. Now check, $(1,2) \in ?$

$(6-5, 6-4)$

It belongs to S^2 and S^1 but it is originated from S^1 i.e $x_1 = 1$ (x_1 = (1,2) placed fully in the knapsack). ~~(Fill the 1)~~ So balance in the knapsack = 0, so $x_2 = 0$.

$\therefore (x_1, x_2, x_3) = (1,0,1)$

***

→ All pairs shortest path :-

Description :- When a weighted Graph represented by a weight matrix W then the objective is to find the distance b/w every pair of nodes.

- The dynamic programming is applied to solve this problem.

STEP-1 :- Decompose the given problem into sub problems.

Let $A^k_{(i,j)}$ be the length of shortest path from node i to j such that the label for every intermediate node will be <= k, ~~such~~ compute A^k for k = 1 ..... n of n nodes.

STEP-2 :- Divide the paths from i node to j node for every intermediate node say k.

Then there arises 2 cases :

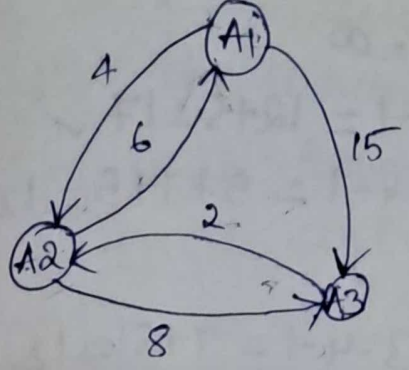
1) path going from i to j through k.
2) path which is not going through k.
3) Select only shortest path from these 2 cases.

STEP-3 :- The shortest path can be computed using bottom up computation in the following recursive method.

$$A^0 = w[i,j]$$

$$A^k_{(i,j)} = \min \{ A^{k-1}_{i,j}, (A^{k-1}_{i,k} + A^{k-1}_{k,j}) \}$$

Ex:-



$$X^0 = \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \end{array} \begin{array}{c} \begin{array}{ccc} A_1 & A_2 & A_3 \end{array} \\ \begin{bmatrix} 0 & 4 & 15 \\ 6 & 0 & 8 \\ \infty & 2 & 0 \end{bmatrix} \end{array}$$

$$x^1 = \begin{array}{c} \\ A1 \\ A2 \\ A3 \end{array} \begin{array}{c} \begin{array}{ccc} A1 & A2 & A3 \end{array} \\ \begin{bmatrix} 0 & 4 & 12 \\ 6 & 0 & 8 \\ \infty & 2 & 0 \end{bmatrix} \end{array}$$

$A_1 - A_2 - A_3 = 4+8 = 12$ ✓

$A_1 - A_3 = 15$

2

$$x^2 = \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \end{array} \begin{array}{c} \begin{array}{ccc} A_1 & A_2 & A_3 \end{array} \\ \begin{bmatrix} 0 & 4 & 12 \\ 6 & 0 & 8 \\ \infty & 2 & 0 \end{bmatrix} \end{array}$$

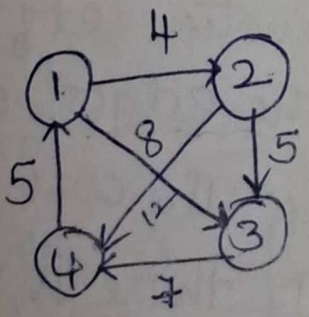
$$x^3 = \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \end{array} \begin{array}{c} \begin{array}{ccc} A_1 & A_2 & A_3 \end{array} \\ \begin{bmatrix} 0 & 4 & 12 \\ 6 & 0 & 8 \\ 8 & 2 & 0 \end{bmatrix} \end{array}$$

$A_3 - A_2 - A_1 = 2+6 = 8$ ✓

$A_3 - A_1 = \infty$

∴ final solution $x^3 = \begin{bmatrix} 0 & 4 & 12 \\ 6 & 0 & 8 \\ 8 & 2 & 0 \end{bmatrix}$

(2)



$$X^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 4 & 8 & \infty \\ \infty & 0 & 5 & 12 \\ \infty & \infty & 0 & 7 \\ 5 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

$$x^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 4 & 8 & 15 \\ \infty & 0 & 5 & 12 \\ \infty & \infty & 0 & 7 \\ 5 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

$1-2-3 = 4+5 = 9$

$1-3 = 8$ ✓

$1-4 = \infty$

$1-2-3-4 = 4+5+7 = 16$

$1-2-4 = 4+12 = 16$

$1-3-4 = 8+7 = 15$ ✓

$$x^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 4 & 8 & 15 \\ 17 & 0 & 5 & 12 \\ \infty & \infty & 0 & 7 \\ 5 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

2-1 = ∞

2-4-1 = 12+5 = 17 ✓

2-3-4-1 = 5+7+5 = 12+5 = 17

$$x^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 4 & 8 & 15 \\ 17 & 0 & 5 & 12 \\ 12 & 16 & 0 & 7 \\ 5 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

3-4-1 = 7+5 = 12 ✓

3-4-1-2

= 7+5+4 = 16

$$x^4 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 4 & 8 & 15 \\ 17 & 0 & 5 & 12 \\ 12 & 16 & 0 & 7 \\ 5 & 9 & 13 & 0 \end{bmatrix} \end{array} //$$

$$\therefore \text{Final soln} = \begin{bmatrix} 0 & 4 & 8 & 15 \\ 17 & 0 & 5 & 12 \\ 12 & 16 & 0 & 7 \\ 5 & 9 & 13 & 0 \end{bmatrix}$$

→ Travelling sales person problem :-

Problem Description :- Let G be a directed graph denoted by (V, E) where V denotes set of vertices and E denotes set of edges. The edges are given along with their cost C_{ij}. The cost C_{ij} is greater than 0 $[C_{ij} > 0]$ $\forall i, j$. If there is no edge between i & j then $C_{ij} = \infty$. A tour for the Graph should be such that all the vertices should be visited only once & cost of the tour is sum of cost of edges on the tour.

∴ The travelling sales person problem is to find the tour of min cost. A Dynamic programming

is used to solve this problem.

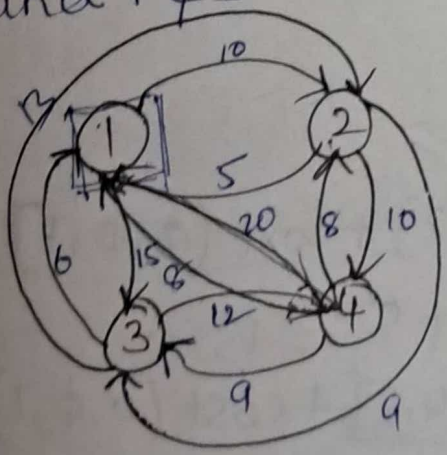
STEP-1:- Let the function $c(1, V-[1])$ is the total length of tour terminating at 1. The objective of TSP is that cost of this tour should be min. Let $d[i,j]$ be the shortest path between 2 vertices i and j

STEP-2:- Let $V_1, V_2 \ldots V_n$ be the sequence of vertices followed in optimal tour, then $(V_1, V_2 \ldots V_n)$ must be a shortest path from v_1 to V_n which passes through each vertex exactly once. The principle of optimality is used. The path $V_i, V_{i+1} \ldots V_j$ must be optimal for all paths beginning at V_i & ending at V_j, and passes through all intermediate vertices V_{i+1} to V_{j-1}

STEP-3:- Following formula can be used to obtain the optimum cost tour.

$$cost(i,S) = \min\{ d[i,j] + cost(j, S-[j]) \}$$ where $j \in S$ and $i \notin S$

Ex:-



The distance matrix can be given by

	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

First select any arbitary vertex say, Select 1.

STEP-1 :- let $S = \phi$ then (No intermediate nodes)

$cost(2, \phi, 1) = d(2,1) = 5$

$cost(3, \phi, 1) = d(3,1) = 6$

$cost(4, \phi, 1) = d(4,1) = 8$

STEP-2 :- let $S = 1$,

$cost(i, s) = \min \{ d[i,j] + cost[j - s[j]) \}$

from vertex 2 to 1 :-

$cost(2, \{3\}, 1) = d[2,3] + cost[3, \phi, 1)$

$= 9 + 6 = 15$

$cost(2, \{4\}, 1) = d[2,4] + cost(4, \phi, 1)$

$= 10 + 8 = 18.$

vertex 3 to 1 :-

$cost(3, \{2\}, 1) = d[3,2] + cost(2, \phi, 1)$

$= 13 + 5 = 18.$

$cost(3, \{4\}, 1) = d[3,4] + cost(4, \phi, 1)$

$= 12 + 8 = 20$

vertex 4 to 1 :-

$cost(4, \{2\}, 1) = d[4,2] + cost(2, \phi, 1)$

$= 8 + 5 = 13$

$cost(4, \{3\}, 1) = d[4,3] + cost(3, \phi, 1)$

$= 9 + 6 = 15.$

STEP-3 :- consider s=2 {two intermediate nodes}

$cost(2,\{3,4\},1) = \min\{[d[2,3] + cost(3,\{4\},1)], [d[2,4] + cost(4,\{3\},1)]\}$

$= \min\{[9+20], [10+15]\} = \min\{(29, 25)\} = 25$

$cost(3,\{2,4\},1) = \min\{[d(3,2) + cost(2,\{4\},1)], [d[3,4] + cost(4,\{2\},1)]\}$

$= \min\{[13+18], [12+13]\}$

$= \min\{[31, 25]\} = 25$

$cost(4,\{2,3\},1) = \min\{[d[4,2] + cost(2,\{3\},1)], [d[4,3] + cost(3,\{2\},1)]\}$

$= \min\{[8+15], [9+18]\}$

$= \min\{[23, 27]\} = 23.$

STEP-4 :- consider s=3. i.e

$cost(1,\{2,3,4\},1)$

$\Rightarrow cost(1,\{2,3,4\},1) =$

$\min([d(1,2) + cost(2,\{3,4\},1)], [d[1,3] + cost(3,\{2,4\},1)], [d[1,4] + cost(4,\{2,3\},1)])$

$= \min\{[10+25, 15+25, 20+23]\}$

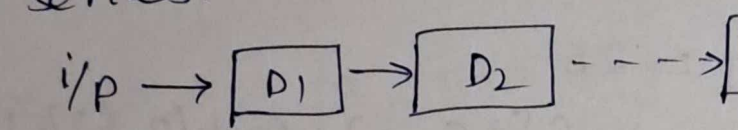
$= \min\{[35, 40, 43]\} = 35$

∴ from vertex 1, obtain optimal path as d(1,2). Hence select vertex 2, and obtain optimal path, i.e d(4,3)

∴ Optimal tree is 1, 2, 4, 3, 1

→ Reliability design:-

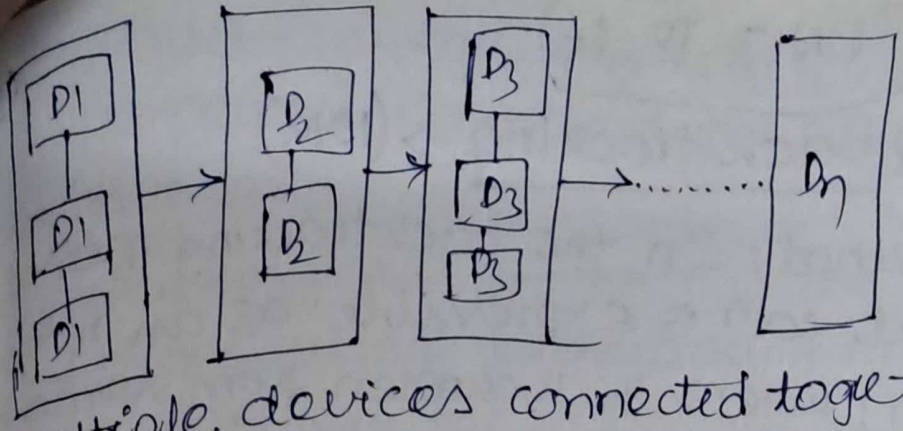
The problem is based on multiplicative optimization function. Here we will consider a system in which many devices are connected in series.



When the devices are connected together then it is necessary that each device should work properly. The probability of that device is i will work properly is called reliability of that device.

→ Let r_i be the reliability of device D_i then the reliability of entire system is $\prod r_i$.

It may happen that even if reliability of individual device is very good but reliability of entire system may not be good. Hence to obtain the good performance from entire system can duplicate individual devices & can connect them in a series. To do so we will attach switching circuits. The job of switching circuit is to determine which device in a group is working properly.



multiple devices connected together.

→ Let m_i be the copies of device D_i, then $(1-r_i)^{m_i}$ be the probability that all m_i have a malfunction Hence, stage reliability is $(1-(1-r_i)^{m_i})$, will denote reliability of stage 'i' by $\phi_i(m_i)$.

$\therefore \phi_i m_i = 1-(1-r_i)^{m_i}$

In, reliability design, to get maximize reliability, use device duplication i.e maximize reliability, $\prod_{1\le i\le n} \phi_i(m_i)$, which is subjected to $\sum_{1\le i\le n} c_i m_i \le c$, where c_i is the cost of each device & c is maximum allocable cost of system.

The dynamic programming sol^n s' consists of tuples, of form (f, x) where $f = f_i(x)$. The $f_i(x)$ can be computed as

$$f_i(x) = \max_{1\le m_i\le u_i} \{\phi(m_i) f_{i-1}(x - c_i m_i)\}$$

$$f_0(x) = 1$$

The upper bound u_i on the cost can be determined as :

$$u_i = \left[(c + c_i - \sum_{1}^{n} c_j)/c_i\right]$$