Module - 4

Dynamic programming

Nithin Kumar
Asst. prof Dept of CS&E
VVCE, Mysuru

We know that divide-and-Conquer technique is used to Solve the problem that can be divided into Independent Subproblem. On the other hand, dynamic programming is one Such Strategy that Can be used to Solve the problem having dependent Subproblems.

That is, in case of Some problems, their Subproblems are Shared & they Cannot be Solved Independently.

→ Consider a problem of finding n^{th} fibonacci number. The formula is given by

$$F(n) = F(n-1) + F(n-2)$$

with initial Conditions $F(0) = 0$ & $F(1) = 1$

Here, if we try to Solve $F(n-1)$ that will Contain a term $F(n-2) + F(n-3)$ so $F(n)$ & its Subproblem $F(n-1)$ are Sharing Another Subproblem $F(n-2)$. Thus Calculating These repeated terms is Simply Waste of time.

→ Instead of Solving overlapping Subproblems Again & Again, Dynamic programming Suggests Solving Each of the Smaller Subproblems only once & recording the results in a table from which we can obtain a Soln for original problem.

* Multistage Graphs :-

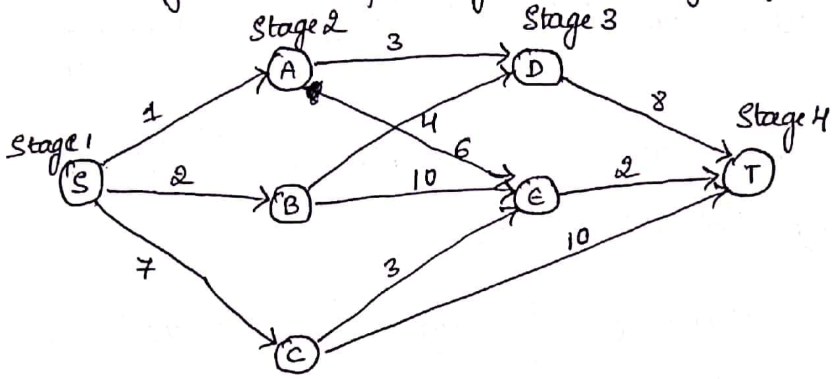
Multistage Graph is a directed graph in which the nodes can be divided into a Set of Stages (k stages) Such that all Edges are from a Stage to next Stage only.

In this graph all the Vertices are partitioned into the "k" Stages where $k >= 2$.

→ In Multistage graph problem we have to find the Shortest path from "Source" to "Sink".

The Cost of a path from Source (denoted by S) to Sink (denoted by T) is the Sum of the Costs of Edges on the path.

→ Consider the foll Example of Multistage graph G



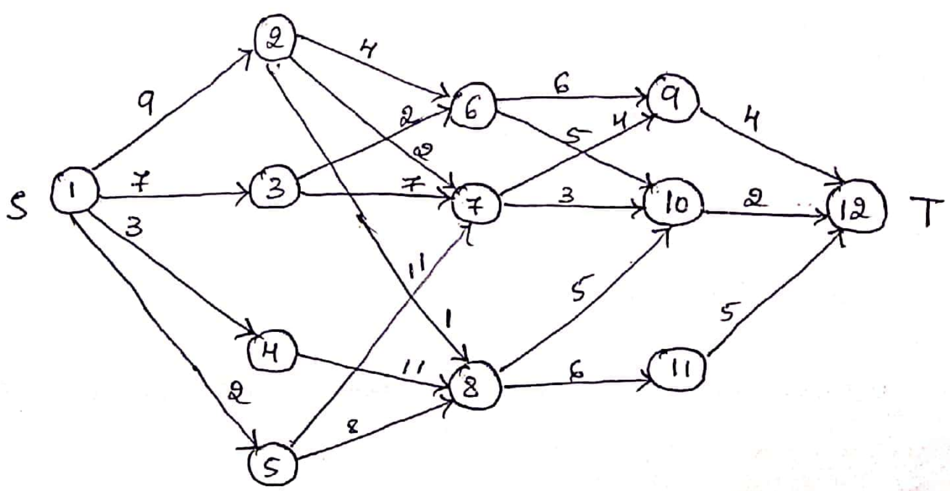
S = Source
T = Sink

→ Multistage graph problem can be Solved to find Shortest path from Source to Sink Using

* Forward Approach
* Backward Approach

→ Using dynamic Approach, the Multistage graph problem is Solved. This is because in Multistage graph problem we obtain the Minimum path at Each Current Stage by Considering the path length of Each Vertex obtained in Earlier Stage.

* Solve the given Multistage graph problem Using forward Approach to find the Shortest path from Source to Sink.



→ Firstly, make a table to indicate "vertex V", "distance of vertex cost(min)" & "reachable through vertex 'd'"

V	1	2	3	4	5	6	7	8	9	10	11	12
cost	16	7	9	18	15	7	5	7	4	2	5	0
d	2/3	7	6	8	8	10	10	10	12	12	12	12

Cost of sink i.e 12 is 0.

→ Next, take every individual vertex at a time & update Min distance in table with through which vertex (d).

Stage 4

* $Cost(4,9) = 4$ * $cost(4,10) = 2$ * $cost(4,11) = 5$

Stage 3

* $Cost(3,6) = \min\{cost(6,9) + cost(9),\ cost(6,10) + cost(10)\}$
$= \min\{6+4,\ 5+2\} = 7$ ∴ $d = 10$

* $cost(3,7) = \min\{cost(7,9) + cost(9),\ cost(7,10) + cost(10)\}$
$= \min\{4+4,\ 3+2\} = 5$ ∴ $d = 10$

* $cost(3,8) = \min\{cost(8,10) + cost(10),\ cost(8,11) + cost(11)\}$
$= \min\{5+2,\ 6+5\} = 7$ ∴ $d = 10$

Stage 2

* $cost(2,2) = \min\{cost(2,6) + cost(6),\ cost(2,7) + cost(7),\ cost(2,8) + cost(8)\}$
$= \min\{4+7,\ 2+5,\ 1+7\} = 7$ ∴ $d = 7$

* $cost(2,3) = \min\{cost(3,6) + cost(6),\ cost(3,7) + cost(7)\}$
$= \min\{2+7,\ 7+5\} = 9$ ∴ $d = 6$

* $cost(2,4) = \min\{cost(4,8) + cost(8)\}$
$= \min\{11+7\} = 18$ ∴ $d = 8$

* $cost(2,5) = \min\{cost(5,7) + cost(7),\ cost(5,8) + cost(8)\}$
$= \min\{11+5,\ 8+7\} = 15$ ∴ $d = 8$

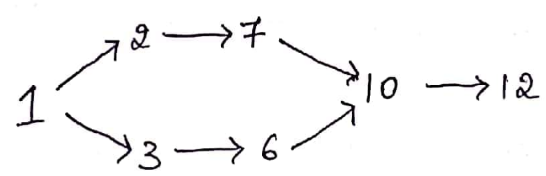
* Stage 1

* $Cost(1,2) = \min\{cost(1,2) + cost(2), cost(1,3) + cost(3), cost(1,4) + cost(4), cost(1,5) + cost(5)\}$

$= \min\{9+7,\ 7+9,\ 3+18,\ 2+15\}$

$= 16$ $\therefore d = 2$ & 3

→ Finally, Backtrack the path from Source '1' using final 'd' values i.e '2' & '3'

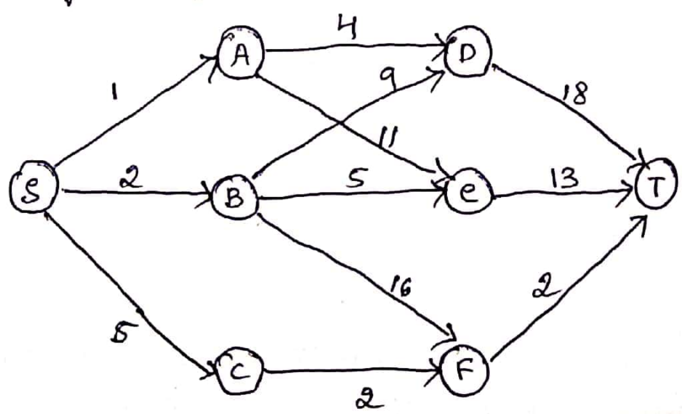


& Total Cost = 16

* Algorithm Multistage Graph (Forward Approach)

```
Multistage Graph (G, k, n)
{
    Cost [n] = 0.0;
    For (int j = n-1; j >= 1; j--)
    {
        Cost [j] = Cost [j] [r] + Cost [r]
        D [j] = r
    }
}
```

* Apply forward Approach to find Shortest path from 'S' to 'T' for given graph



→ Firstly, make a table to indicate "Vertex V", "distance (min cost)" & "reachable vertex d"

V	S	A	B	C	D	E	F	T
cost	9	22	18	4	18	13	2	0
d	C	D	E/F	F	T	T	T	T

Cost of Sink (T) is '0'.

→ Next, take every individual vertex at a time & update min distance in table with reachable vertex (d).

* Stage 3 :-

* Cost (3, D) = 18 * Cost (3, E) = 13 * Cost (3, F) = 2

* Stage 2 :-

* Cost (2, A) = min { Cost (A, D) + Cost (D), Cost (A, E) + Cost (E) }
 = min { 4 + 18, 11 + 13 } = 22 ∴ d = D

* Cost (2, B) = min { Cost (B, D) + Cost (D), Cost (B, E) + Cost (E), Cost (B, F) + Cost (F) }
 = min { 9 + 18, 5 + 13, 16 + 2 } = 18 ∴ d = E & F

* Cost (2, C) = Cost (C, F) + Cost (F)
 = 2 + 2 = 4 ∴ d = F

* Stage 1 :-

* Cost (1, S) = min { Cost (S, A) + Cost (A), Cost (S, B) + Cost (B), Cost (S, C) + Cost (C) }
 = min { 1 + 22, 2 + 18, 5 + 4 } = 9 ∴ d = C

→ Finally, Backtrack the path from Source 'S' using final 'd' value 'C'

S ⟶ C ⟶ F ⟶ T & Total Cost = 9

* Knapsack problem Using Dynamic programming :-

Consider a Knapsack problem of finding the Most Valuable Subset of n items of weights $w_1 \ldots w_n$ & Values $V_1 \ldots V_n$ that fit into a knapsack of capacity 'M'.

→ The dynamic programming Strategy for Solving this problem requires to derive a recurrance relation that Expresses a Soln to an Instance of knapsack problem interms of soln to its smaller Sub Instances.

→ Consider an Instance of a problem with first 'i' items having weights $w_1 \ldots w_i$ & Values $V_1 \ldots V_i$ & knapsack Capacity is 'M'

We have the foll two Conditions

$$* V[i,j] = \begin{cases} \max\{V[i-1,j],\ V_i + V[i-1, j-w_i]\} & \text{if } j-w_i \geq 0 \\ V[i-1,j] & \text{if } j-w_i < 0 \end{cases}$$

With the Initial Conditions

$* V[0,j] = 0$ & $* V[i,0] = 0$

Our requirement is to find $V[n, M]$ based on the above relation.

* Algorithm Dynamic Knapsack (n, M)

```
while (v[i,j] ≯ 0)
{
   if j < wi
        Val ← V[i-1,j]
   Else
        Val ← max{V[i-1,j], Vi + V[i-1, j - Wi]}
   V[i,j] ← Val
   return V[i,j]
```

* Consider the foll problem with three items & the knapsack of capacity W=4, the weights & values are as shown

Item	Weight	Value
A	3	25
B	1	20
C	2	40

⇒ $M=4$ $w_1=3$ $w_2=1$ $w_3=2$ $V_1=25$ $V_2=20$ $V_3=40$

→ Firstly, Create a table in which dimension i → indicates no of objects including '0' & j → indicates capacity of knapsack including '0'

i \ j	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	25	25
2	0	20	20	25	45
3	0	20	40	60	60

fill the first row & column with 0's, As we know that $V[0,j]=0$ & $V[i,0]=0$

→ Now for remaining values perform computation based on relation

$$*V[i,j] = \begin{cases} \max\{V[i-1,j],\ V_i + V[i-1,j-w_i]\} & j-w_i \geq 0 \\ V[i-1,j] & j-w_i < 0 \end{cases}$$

* $\underline{i=1}$:- $i=1$ & $w_1=3$ $V_1=25$

* $V[1,1] = V[i-1,j] = V[0,1] = 0$ ∵ $j-w_i = 1-3 < 0$

* $V[1,2] = V[i-1,j] = V[0,2] = 0$ ∵ $j-w_i = 2-3 < 0$

* $V[1,3] = \max\{V[i-1,j],\ V_i + V[i-1,j-w_i]\}$ ∵ $j-w_i = 3-3 = 0$
$= \max\{V[0,3],\ 25+V[0,0]\} = 25$

* $V[1,4] = \max\{V[i-1,j],\ V_i + V[i-1,j-w_i]\}$ ∵ $j-w_i > 0$ $4-3=1$
$= \max\{V[0,4],\ 25+V[0,1]\} = 25$

* $i = 2$:-

$i = 2 \quad w_2 = 1 \quad V_2 = 20$

* $V[2,1] = \max\{V[i-1,j],\ V_i + V[i-1, j-w_i]\}$
$= \max\{V[1,1],\ 20 + V[1,0]\} = \max\{0,\ 20+0\}$
$= 20$

* $V[2,2] = \max\{V[i-1,j],\ V_i + V[i-1, j-w_i]\}$
$= \max\{V[1,2],\ 20 + V[1,1]\} = \max\{0,\ 20+0\}$
$= 20$

* $V[2,3] = \max\{V[i-1,j],\ V_i + V[i-1, j-w_i]\}$
$= \max\{V[1,3],\ 20 + V[1,2]\} = \max\{25,\ 20+0\}$
$= 25$

* $V[2,4] = \max\{V[i-1,j],\ V_i + V[i-1, j-w_i]\}$
$= \max\{V[1,4],\ 20 + V[1,3]\} = \max\{25,\ 20+25\}$
$= 45$

* $i = 3$:-

$i = 3 \quad w_3 = 2 \quad V_3 = 40.$

* $V[3,1] = V[i-1, j] = V[2,1] = 20$

* $V[3,2] = \max\{V[i-1,j],\ V_i + V[i-1, j-w_i]\}$
$= \max\{V[2,2],\ 40 + V[2,0]\} = \max\{20,\ 40+0\}$
$= 40$

* $V[3,3] = \max\{V[i-1,j],\ V_i + V[i-1, j-w_i]\}$
$= \max\{V[2,3],\ 40 + V[2,1]\} = \max\{25,\ 40+20\}$
$= 60$

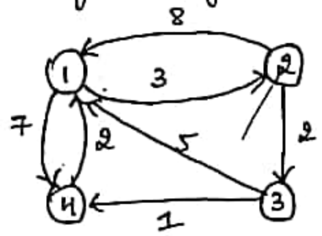
* $V[3,4] = \max\{V[i-1,j],\ V_i + V[i-1, j-w_i]\}$
$= \max\{V[2,4],\ 40 + V[2,2]\} = \max\{45,\ 40+20\}$
$= 60$

→ Here, we have

$$V[n, W] = V[3,4] = 60$$

∴ By observing the table, the soln is $\{2,3\}$ or $\{B, C\}$ with the profit 60.

* Apply Floyd's Algo to find all pair Shortest path



$$A^0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 & \infty \\ 3 & 5 & \infty & 0 & 1 \\ 4 & 2 & \infty & \infty & 0 \end{array}$$

$$A^k[i,j] = \min\{A^{k-1}[i,j],\ A^{k-1}[i,k] + A^{k-1}[k,j]\}$$

$$* A^1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & \infty & 0 \end{array}$$

$A[2,3] = \min\{2,\ A[2,1] + A[1,3]\} = \{2,\ 8+\infty\} = 2$

$A[2,4] = \min\{\infty,\ A[2,1] + A[1,4]\} = \{\infty,\ 8+7\} = 15$ //

$A[3,2] = \min\{\infty,\ A[3,1] + A[1,2]\} = \{\infty,\ 5+3\} = 8$ //

$A[3,4] = \min\{1,\ A[3,1] + A[1,4]\} = \{1,\ 5+7\} = 1$

$A[4,2] = \min\{\infty,\ A[4,1] + A[1,2]\} = \{\infty,\ 2+3\} = 5$

$A[4,3] = \min\{\infty,\ A[4,1] + A[1,3]\} = \{\infty,\ 2+\infty\} = \infty$

$$* A^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

$A[1,3] = \min\{\infty,\ A[1,2] + A[2,3]\} = \{\infty,\ 3+2\} = 5$ //

$A[1,4] = \min\{7,\ A[1,2] + A[2,4]\} = \{7,\ 3+15\} = 7$

$A[3,1] = \min\{5,\ A[3,2] + A[2,1]\} = \{5,\ 8+8\} = 5$

$A[3,4] = \min\{1,\ A[3,2] + A[2,4]\} = \{1,\ 8+15\} = 1$ //

$A[4,1] = \min\{2,\ A[4,2] + A[2,1]\} = \{2,\ 5+8\} = 2$ //

$A[4,3] = \min\{\infty,\ A[4,2] + A[2,3]\} = \{\infty,\ 5+2\} = 7$ //

$$* A^3 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 6 \\ 2 & 7 & 0 & 2 & 3 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

$A[1,2] = \min\{3,\ A[1,3] + A[3,2]\} = \{3,\ 5+8\} = 3$

$A[1,4] = \min\{7,\ A[1,3] + A[3,4]\} = \{7,\ 5+1\} = 6$ //

$A[2,1] = \min\{8,\ A[2,3] + A[3,1]\} = \{8,\ 2+5\} = 7$ //

$A[2,4] = \min\{15,\ A[2,3] + A[3,4]\} = \{15,\ 2+1\} = 3$ //

$A[4,1] = \min\{2,\ A[4,3] + A[3,1]\} = \{2,\ 7+5\} = 2$ //

$A[4,2] = \min\{5,\ A[4,3] + A[3,2]\} = \{5,\ 7+8\} = 5$ //

$$* A^4 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 6 \\ 2 & 5 & 0 & 2 & 3 \\ 3 & 3 & 6 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

$A[1,2] = \min\{3,\ A[1,4] + A[4,2]\} = \{3,\ 6+5\} = 3$

$A[1,3] = \min\{5,\ A[1,4] + A[4,3]\} = \{5,\ 6+7\} = 5$

$A[2,1] = \min\{7,\ A[2,4] + A[4,1]\} = \{7,\ 3+2\} = 5$ //

$A[2,3] = \min\{2,\ A[2,4] + A[4,3]\} = \{2,\ 3+7\} = 2$

$A[3,1] = \min\{5,\ A[3,4] + A[4,1]\} = \{5,\ 1+2\} = 3$ //

$A[3,2] = \min\{8,\ A[3,4] + A[4,2]\} = \{8,\ 1+5\} = 6$ //

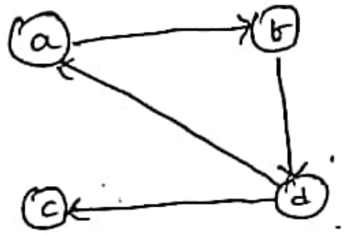
$$\therefore A = \begin{bmatrix} 0 & 3 & 5 & 6 \\ 5 & 0 & 2 & 3 \\ 3 & 6 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix}$$

* **Warshall's Algo : To find Transitive closure**

$$R^{(k)}[i,j] \leftarrow R^{(k-1)}[i,j] \text{ OR } R^{(k-1)}[i,k] \text{ AND } R^{(k-1)}[k,j]$$

Eg:

→ Consider the foll Example



$$R^0 = \begin{array}{c|cccc} & 1\,a & 2\,b & 3\,c & 4\,d \\ \hline 1\,a & 0 & 1 & 0 & 0 \\ 2\,b & 0 & 0 & 0 & 1 \\ 3\,c & 0 & 0 & 0 & 0 \\ 4\,d & 1 & 0 & 1 & 0 \end{array}$$

$$R^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array} \qquad R_0 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

$$R^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

R(2,2) = 0 OR R[2,1] AND R[1,2]
= 0 OR 0 AND 1 = 0

R(2,4) = 0 OR R[2,1] AND R[2,4] = 0 OR 0 AND 0 = 0

R(3,2) = 0 OR R[3,1] AND R[1,2] = 0 OR 0 A 1 = 0

R(3,3) = 0 OR R[3,1] AND R[1,3] = 0 OR 0 A 0

R(3,4) = 0 OR R[3,1] AND R[1,4] = 0 OR 0 A 0

R(4,2) = 0 OR R[4,1] AND R[1,2] = 0 OR 1 A 1

R(4,4) = 0 OR R[4,1] AND R[1,4] = 0 OR 1 A 0 = 0

$$R^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

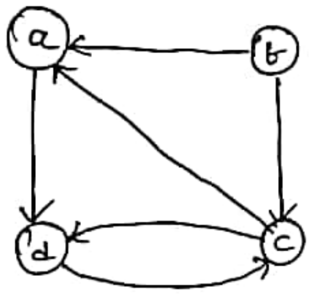
$$R^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array} \qquad R^4 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

$R^{(0)} \leftarrow A$

for $k \leftarrow 1$ to n do
  for $i \leftarrow 1$ to n do
    for $j \leftarrow 1$ to n do
      $R^{k}[i,j] \leftarrow R^{(k-1)}[i,j]$ OR $R^{(k-1)}[i,k]$ AND $R^{(k-1)}[k,j]$

return $(R^{(n)})$

*



$$R^{0} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

$$R^{1} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

$R[2,4] = 0$ OR $R[2,1]$ AND $R[1,4]$
$= 0$ OR 1 AND 1
$= 1$ //

$$R^{2} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

$$R^{3} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

$R[4,4] = 0$ OR $R[4,3]$ AND $R[3,4]$
$= 0$ OR 1 AND $1 = 1$ //

$$R^{4} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

//

* **Traveling Salesman problem**

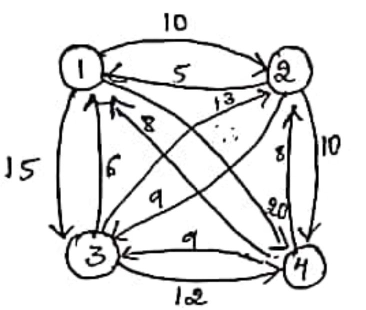
Travelling Sales Man problem consists of Salesman & a set of cities. The Salesman has to visit Each city Starting from home city & return to the same city.

→ The person wants to Minimize the total length of the trip.

→ let $g(i, S)$ be the length of Shortest path starting at vertex i, going through all vertices in S & terminating at vertex i

$$* \; g(i,S) = \min_{j \in S} \{ C_{ij} + g(j, S - \{j\}) \} //$$

→ Consider the foll Example



$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 10 & 15 & 20 \\ 2 & 5 & 0 & 9 & 10 \\ 3 & 6 & 13 & 0 & 12 \\ 4 & 8 & 8 & 9 & 0 \end{array}$$

→ * Firstly for set $\{\phi\}$

$g\{i, \phi\} = C_{i1} \Rightarrow g\{1, \phi\} = C_{11} = 0 \quad g\{3, \phi\} = C_{31} = 6$

$g\{2, \phi\} = C_{21} = 5 \quad g\{4, \phi\} = C_{41} = 8$

→ Next for all other vertices from src vertex $\{1\}$

$* \; g\{1, \{2,3,4\}\} = \min\{ C_{12} + g(2, \{3,4\}),\ C_{13} + g(3, \{2,4\}),\ C_{14} + g(4, \{2,3\}) \}$

$= \min\{10 + 25,\ 15 + 25,\ 20 + 23\} = 35 //$

$* \; g(2, \{3,4\}) = \min\{ C_{23} + g(3, \{4\}),\ C_{24} + g(4, \{3\}) \}$

$= \min\{ 9 + g(3, \{4\}),\ 10 + g(4, \{3\}) \}$

$= \min\{ 9 + 20,\ 10 + 15 \}$

$= 25 //$

* $g(3,\{4\}) = \min\{C_{34} + g(4,\phi)\}$
$= 12 + 8 = 20$ //

* $g(4,\{3\}) = \min\{C_{43} + g(3,\phi)\}$
$= 9 + 6 = 15$ //

→ * $g(3,\{2,4\}) = \min\{C_{32} + g(2,\{4\}),\ C_{34} + g(4,\{2\})\}$
$= \min\{13 + g(2,\{4\}),\ 12 + g(4,\{2\})\}$
$= \{13 + 18,\ 12 + 13\}$
$= 25$

* $g(2,\{4\}) = \min\{C_{24} + g(4,\phi)\}$
$= 10 + 8 = 18$ //

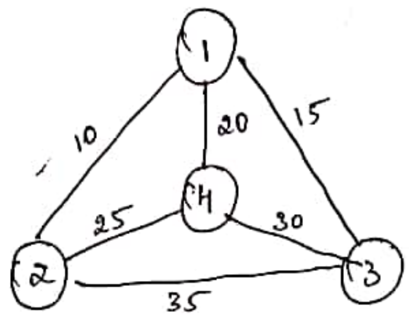
* $g(4,\{2\}) = \min\{C_{42} + g(2,\phi)\}$
$= \min\{8 + 5\} = 13$ //

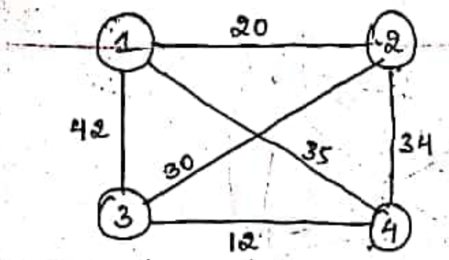
* $g(4,\{2,3\}) = \min\{C_{42} + g(2,\{3\}),\ C_{43} + g(3,\{2\})\}$
$= \min\{8 + g(2,\{3\}),\ 9 + g(3,\{2\})\}$
$= \underbrace{8 + 15}_{=23},\ 9 + 18$

* $g(2,\{3\}) = \min\{C_{23} + g(3,\phi)\}$
$= 9 + 6 = 15$

* $g(3,\{2\}) = \min\{C_{32} + g(2,\phi)\}$
$= 13 + 5 = 18$

o $1 \to 2 \to 4 \to 3 = 35$ //





$g(i, s) = \min\{C_{ij} + g(j, \{s\} - j)\}$

→ Firstly for set $\{\phi\}$

* $g(i,\phi) \Rightarrow g(1,\phi) = C_{11} = 0 \quad g(2,\phi) = C_{21} = 20 \quad g(3,\phi) = C_{31} = 42 \quad g(4,\phi) = C_{41} = 35$

→ Next for all vertices from src $\{1\}$

* $g(1,\{2,3,4\}) = \min\{\underline{C_{12} + g(2,\{3,4\})},\ C_{13} + g(3,\{2,4\}),\ \underline{C_{14} + g(4,\{2,3\})}\}$
  $= \min\{20 + 77,\ 42 + 66,\ 35 + 62\} = 97$

* $g(2,\{3,4\}) = \min\{\underline{C_{23} + g(3,\{4\})},\ C_{24} + g(4,\{3\})\}$
  $= \min\{30 + 47,\ 34 + 54\} = 77$

  $g(3,\{4\}) = \min\{C_{34} + g(4,\phi)\} = 12 + 35 = 47$
  $g(4,\{3\}) = \min\{C_{43} + g(3,\phi)\} = 12 + 42 = 54$

* $g(3,\{2,4\}) = \min\{C_{32} + g(2,\{4\}),\ \underline{C_{34} + g(4,\{2\})}\}$
  $= \min\{30 + 69,\ 12 + 54\} = 66$

  $g(2,\{4\}) = \min\{C_{24} + g(4,\phi)\} = 34 + 35 = 69$
  $g(4,\{2\}) = \min\{C_{42} + g(2,\phi)\} = 34 + 20 = 54$

* $g(4,\{2,3\}) = \min\{C_{42} + g(2,\{3\}),\ \underline{C_{43} + g(3,\{2\})}\}$
  $= \min\{34 + 72,\ 12 + 50\} = 62$

  $g(2,\{3\}) = \min\{C_{23} + g(3,\phi)\} = 30 + 42 = 72$
  $g(3,\{2\}) = \min\{C_{32} + g(2,\phi)\} = 30 + 20 = 50$

$1 \to 4 \to 3 \to 2 \to 1$ $\qquad$ $1 \to 2 \to 3 \to 4 \to 1$

Scanned with CamScanner

* 0/1 Knapsack :-

$$* \; V[i,j] = \begin{cases} \max\{V[i-1,j],\; V_i + V[i-1, j-w_i]\} & \text{if } j - w_i \geq 0 \\ V[i-1,j] & \text{if } j - w_i < 0 \end{cases} \quad \& \; V[0,j] = 0 \quad V[i,0] = 0$$

⇒ ~~[illegible]~~ $M = 5$

Item	Weight	Value
1	2	12
2	1	10
3	3	20
4	2	15

→ j, i ↓

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	12	12	12	12
2	0	10	12	22	22	22
3	0	10	12	22	30	32
4	0	10	15	25	30	37

* i = 1 :- $i = 1 \quad w_1 = 2 \quad V_1 = 12$

$V[1,1] = V[i-1,j] = V[0,1] = 0 \qquad V[1,2] = \max\{V[0,2],\; 12 + V[0,0]\} = 12$

$V[1,3] = \max\{V[0,3],\; 12 + V[0,1]\} = 12 \qquad V[1,4] = \max\{V[0,4],\; 12 + V[0,2]\} = 12$

$V[1,5] = \max\{V[0,5],\; 12 + V[0,3]\} = 12$

* i = 2 :- $i = 2 \quad w_2 = 1 \quad V_2 = 10$

$V[2,1] = \max\{V[1,1],\; 10 + V[1,0]\} = 10 \qquad V[2,2] = \max\{V[1,2],\; 10 + V[1,1]\} = 12$

$V[2,3] = \max[V[1,3],\; 10 + V[1,2]\} = 22 \qquad V[2,4] = \max\{V[1,4],\; 10 + V[1,3]\} = 22$

$V[2,5] = \max[V[1,5],\; 10 + V[1,4]\} = 22$

* i = 3 :- $i = 3 \quad w_3 = 3 \quad V_3 = 20$

$V[3,1] = V[2,1] = 10 \qquad V[3,2] = V[2,2] = 12$

$V[3,3] = \max\{V[2,3],\; 20 + V[2,0]\} = 22 \qquad V[3,4] = \max\{V[2,4],\; 20 + V[2,1]\} = 30$

$V[3,5] = \max\{V[2,5],\; 20 + V[2,2]\} = 32$

* i = 4 :- $i = 4 \quad w_4 = 2 \quad V_4 = 15$

$V[4,1] = V[3,1] = 10 \qquad V[4,2] = \max\{V[3,2],\; 15 + V[3,0]\} = 15$

$V[4,3] = \max\{V[3,3],\; 15 + V[3,1]\} = 25 \qquad V[4,4] = \max\{V[3,4],\; 15 + V[3,2]\} = 30$

$V[4,5] = \max\{V[3,5],\; 15 + V[3,3]\} = 37$

Scanned with CamScanner