

## Module -4

### Dynamic Programming.

This technique also divides the problem into sub-problems where

- (1) Sub problems are dependent / overlapping
- (2) Solution of the last subproblem is the sol of original problem

(3) Solution of every subproblem is saved which can avoid resolving of similar instance of the subproblem P

$P_1, P_2, P_3, \dots, P_K$

Set of original sub problems

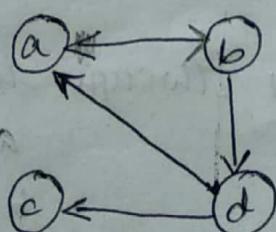
#### Warshall's Algorithm

This is used to find path from one node to another node. Generally, warshall's algorithm is used to obtain path matrix (transitive closure) of a given directed, unweighted graph.

If  $G = (V, E)$  is a graph of  $n$  vertices then, path matrix is a matrix of size  $n \times n$   
 Where  $P[i, j] = \begin{cases} 1 & \text{if there is a path from } i \text{ to } j \\ 0 & \text{if there is no path} \end{cases}$

Adj. Matrix ( $4 \times 4$ )

Example:



	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	0	1	0

path matrix ( $4 \times 4$ )

	a	b	c	d
a	1	1	1	1
b	1	1	1	1
c	0	0	0	0
d	1	1	1	1

Starting from A I can reach C, D

Logic behind Warshall's Algorithm

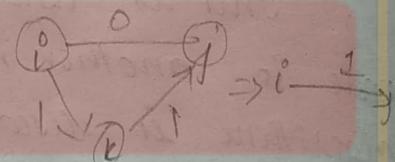
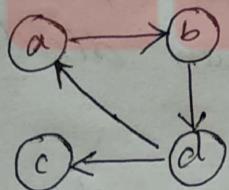
If,  $P[i, j] = 1$  it means that there is a path from  $i$  to  $j$

If,  $P[i, j] = 0$  it means that there is no path from  $i$  to  $j$  but, if there exist a path from  $(i$  to  $k)$   $(i, k)$  and  $(k, j)$  then it implies that there exists a path from  $i$  to  $j$   $(i, j)$

Eg: Apply Warshall's transitive closure for the

algorithm to obtain given graph.

Graph



Adjacency Matrix is

	a	b	c	d
a	1	0	0	0
b	0	1	0	0
c	0	0	1	0
d	0	0	0	1

Step 1: Consider the paths through the vertex  $a$  (first subproblem)  
i/p ii: Adjacency Matrix

$$(i, k) \quad (k, i) \Rightarrow (i, i)$$

$$(d, a) = 1 \times (a, b) = 1 \Rightarrow (d, b) = 1$$

	a	b	c	d
a	1	0	0	0
b	0	1	0	0
c	0	0	1	0
d	0	0	0	1

Since we are finding path through the vertex 'a' consider  $a^{\text{th}}$  row &  $a^{\text{th}}$  column from the above adjacency matrix and also consider only the pairs with entries 1.

Here 'k' is nothing but vertex 'a' and also we know that if  $(i,k)=1 \wedge (k,j)=1$  then  $(i,j)=1$   
Since  $(d,a)=1 \wedge (a,b)=1$  then  $(d,b)=1$

Step 2: Consider the paths through the vertex 'b' obtained by considering  $b^{\text{th}}$  row &  $b^{\text{th}}$  column of the matrix obtained in step 1.

$$(a,b)=1 \wedge (b,d)=1 \therefore (a,d)=1$$

$$(d,b)=1 \wedge (b,d)=1 \therefore (d,d)=1$$

	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	1	1	1

Step 3: Consider paths through vertex 'c' obtained by considering  $c^{\text{th}}$  row &  $c^{\text{th}}$  column of the matrix obtained in step 1.

Step 2.

Note that  $(d,c)=1$  but, all the entries in  $c^{\text{th}}$  row are 0. Which means that there is no path through the vertex c

$\therefore$  The resultant matrix is unchanged.

	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	1	1	1

Step 4: Consider paths through vertex 'd'. Consider  $d^{\text{th}}$  row and  $d^{\text{th}}$  column of the matrix obtained in step 3

Start from  $a$ th column , combine with row.

$$\begin{array}{l} (a,a)=1 \quad \wedge \quad (d,a)=1 \quad \therefore \quad (a,a)=1 \\ (a,d)=1 \quad \wedge \quad (d,b)=1 \quad \therefore \quad (a,b)=1 \\ (a,d)=1 \quad \wedge \quad (d,c)=1 \quad \therefore \quad (a,c)=1 \\ (a,d)=1 \quad \wedge \quad (d,d)=1 \quad \therefore \quad (a,d)=1 \end{array}$$

$$\begin{array}{l} (b,d)=1 \quad \wedge \quad (d,a)=1 \quad \therefore \quad (b,a)=1 \\ (b,d)=1 \quad \wedge \quad (d,b)=1 \quad \therefore \quad (b,b)=1 \\ (b,d)=1 \quad \wedge \quad (d,c)=1 \quad \therefore \quad (b,c)=1 \\ (b,d)=1 \quad \wedge \quad (d,d)=1 \quad \therefore \quad (b,d)=1 \end{array}$$

$$\begin{array}{l} (d,d)=1 \quad \wedge \quad (d,a)=1 \quad \therefore \quad (d,a)=1 \\ (d,d)=1 \quad \wedge \quad (d,b)=1 \quad \therefore \quad (d,b)=1 \\ (d,d)=1 \quad \wedge \quad (d,c)=1 \quad \therefore \quad (d,c)=1 \end{array}$$

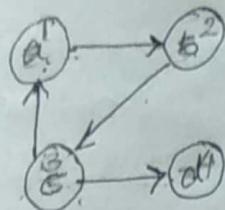
The matrix with the updated values

$$\begin{matrix} & a & b & c & d \\ a & 1 & 1 & 1 & 1 \\ b & 1 & 1 & 1 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 1 \end{matrix}$$

Finally the path matrix is given below.

$$\begin{matrix} & a & b & c & d \\ a & 1 & 1 & 1 & 1 \\ b & 1 & 1 & 1 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 1 \end{matrix}$$

9.



Step adjacency matrix i)

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

Step 1: Consider a path through matrix '1'

$$(3, 1) = 1 \wedge (1, 2) = 1 \Rightarrow (3, 2) = 1$$

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

Step 2: consider a path through matrix '2'.

$$(1, 2) = 1 \wedge (2, 3) = 1 \Rightarrow (1, 3) = 1$$

$$(3, 2) = 1 \wedge (2, 3) = 1 \Rightarrow (3, 3) = 1$$

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

Step 3: consider a path through matrix '3'

$$(1, 3) = 1 \wedge (3, 1) = 1 \Rightarrow (3, 3) = 1 \wedge (1, 1) = 1$$

$$(1, 3) = 1 \wedge (3, 2) = 1 \Rightarrow (3, 3) = 1 \wedge (1, 2) = 1$$

$$(1, 3) = 1 \wedge (3, 3) = 1 \Rightarrow (1, 3) = 1$$

$$(1, 3) = 1 \wedge (3, 4) = 1 \Rightarrow (1, 4) = 1$$

$$(2, 3) = 1 \wedge (3, 1) = 1 \Rightarrow (2, 1) = 1$$

$$(2, 3) = 1 \wedge (3, 2) = 1 \Rightarrow (2, 2) = 1$$

$$(2, 3) = 1 \wedge (3, 3) = 1 \Rightarrow (2, 3) = 1$$

$$(2, 3) = 1 \wedge (3, 4) = 1 \Rightarrow (2, 4) = 1$$

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

Step 4: All the entries in 4<sup>th</sup> row is 0 which means there is no path through 4.

Finally path Matrix is

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

Warshall's Algorithm.

Algorithm Warshall ( $A[1..n][1..n]$ ,  $\alpha$ )

{ // Initialization

for  $i \leftarrow 1$  to  $n$  do

for  $j \leftarrow 1$  to  $n$  do

$P[i, j] = A[i, j];$

$A$ : Adjacency matrix

$P$ : Path matrix

$i \xrightarrow{0} i \Rightarrow i \xrightarrow{1} i$

for every  $k$  check( $i, k$ )

corresponding for  $i$

// Checking paths

for  $k \leftarrow 1$  to  $n$  do

for  $i \leftarrow 1$  to  $n$  do

for  $j \leftarrow 1$  to  $n$  do

if ( $P[i, j] == 0$   $\wedge$  ( $P[i, k] == 1$   $\wedge$   $P[k, j] == 1$ ))

$P[i, j] = 1;$

Ind for  $j$

Ind for  $i$

Ind for  $k$

return  $P$ ;

Note

- Initially the path matrix will be same as the adjacency matrix.
- for every intermediate vertex 'k' ranging from 1...n we need to check the condition of transitive closure for every pair (i,j) both ranging from 1...n. Hence, for loop w.r.t to 'k' is the outermost for loop, followed with i and j for loop.

### Analysis

- IIP size : n (no of vertices)
- Basic operation : condition within 'if'  
 $p[i,j] = 0 \text{ AND } (p[i,k] == 1 \text{ AND } p[k,j] == 1)$

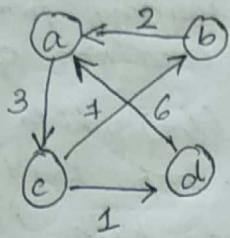
- No of times B.O executed

$$\begin{aligned} T(n) &= \sum_{k=1}^n \sum_{i=1}^n \sum_{j=1}^n 1 \\ &= \sum_{k=1}^n \sum_{i=1}^n (n-1+1) \\ &= \sum_{k=1}^n n \sum_{i=1}^n 1 \\ &= \sum_{k=1}^n n^2 \\ &= n^2 \sum_{k=1}^n 1 \\ &= n^3 \end{aligned}$$

$$\therefore T(n) \in \Theta(n^3)$$

# Floyd's Algorithm [All pair shortest path problem] (APSP)

- Edge starts & ends at one vertex : 0 (self loop)
- No edge :  $\infty$



→ Every vertex to all other vertices (self loop = 0)

Adjacency Matrix :  $(4 \times 4)$

	a	b	c	d
a	0	$\infty$	3	$\infty$
b	2	0	$\infty$	$\infty$
c	$\infty$	7	0	1
d	6	$\infty$	$\infty$	0

→ Floyd's Algorithm is used to find shortest path from every vertex to all other vertices in the directed and weighted graph.

→ To represent a weighted graph we use cost adjacency matrix to the following conditions.

(i)  $\text{cost}(i, i) = 0$  // for self loops.

(ii)  $\text{cost}(i, j) = w$  i.e., assign the weight if there is a edge from  $i$  to  $j$

(iii)  $\text{cost}(i, j) = \infty$  // No edge between  $i$  &  $j$

Shortest path Matrix :  $(4 \times 4)$

	a	b	c	d
a	0	10	3	4
b	2	0	5	6
c	7	7	0	1
d	6	16	9	0

C: Adjacency Matrix

D: Shortest Path Matrix

$$\begin{array}{l} i \xrightarrow{x} j \\ y \xrightarrow{z} k \\ x + z \text{ is test} \end{array}$$

## Lloyd's Algorithm

Algorithm Lloyd (  $C[1..n][1..n], x$  )

for  $i \leftarrow 1$  to  $n$  do

    for  $j \leftarrow 1$  to  $n$  do

$$D[i, j] = C[i, j];$$

    for  $k \leftarrow 1$  to  $n$  do

        for  $i \leftarrow 1$  to  $n$  do

            for  $j \leftarrow 1$  to  $n$  do

$$D[i, j] = \min(D[i, j], D[i, k] + D[k, j]);$$

        end for  $j$

    end for  $i$

end for  $k$

return  $D$ ;

}

### Analysis

→ I/P size :  $n$  (no of vertices)

→ B.O :  $D[i, j] = \min(D[i, j], D[i, k] + D[k, j])$

→ No. of times B.O executed.

$$T(n) = \sum_{k=1}^n \sum_{i=1}^n \sum_{j=1}^n 1.$$

$$= \sum_{k=1}^n \sum_{i=1}^n (n-1+1)$$

$$= \sum_{k=1}^n n \sum_{i=1}^n 1.$$

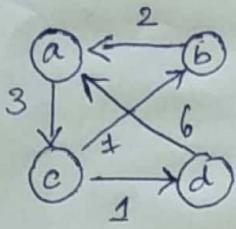
$$= n^2 \sum_{k=1}^n 1$$

$$= n^3$$

$$T(n) \in \Theta(n^3)$$

### Problem

1. Apply Floyd's Algorithm to solve 'All pair shortest path problem' for the given digraph



The cost adjacency matrix is as below.

	a	b	c	d
a	0	$\infty$	3	$\infty$
b	2	0	$\infty$	$\infty$
c	$\infty$	7	0	1
d	6	$\infty$	$\infty$	0

Step 1: Consider the shortest path distance through the vertex 'a'. Hence consider  $a^{\text{th}}$  row &  $a^{\text{th}}$  column from the above cost adjacency matrix and the entries which are non zero and non infinity.

Consider column a, 2 corresponds to (b,a)  
Combine with  $a^{\text{th}}$  row, 3 corresponds to (a,c)

$$\text{So, } (b,a) = 2 \text{ & } (a,c) = 3$$

$$\begin{aligned} \text{Hence } (b,c) &= \min \{ (b,c), (b,a)+(a,c) \} \\ &= \min \{ \infty, 2+3 \} \end{aligned}$$

$$\therefore \boxed{(b,c) = 5}$$

(consider column club with its corresponding row  
 $(a,a) = 6 \text{ & } (a,c) = 3$ )

$$\begin{aligned} \text{Hence } (d,c) &= \min \{ (d,c), (d,a)+(a,c) \} \\ &= \min \{ \infty, 6+3 \} \end{aligned}$$

$$\therefore \boxed{(d,c) = 9}$$

Hence the resultant matrix is.

	a	b	c	d
a	0	$\infty$	3	$\infty$
b	2	0	5	$\infty$
c	$\infty$	7	0	1
d	6	$\infty$	9	0

Step 2 : consider the shortest path distance through vertex b consider 5th row & 6th column from the matrix obtained in step 1.

$$\text{So, } (c, b) = ? \quad \& \quad (b, a) = 2$$

$$(c, a) = \min \{ (c, a), (c, b) + (b, a) \}$$
$$= \min \{ \infty, 9 \}$$

$$\boxed{(c, a) = 9}$$

$$(c, b) = ? \quad \& \quad (b, c) = 5$$

$$(c, c) = \min \{ (c, c), (c, b) + (b, c) \}$$
$$= \min \{ 0, 12 \}$$

$$\therefore \boxed{(c, c) = 0}$$

Hence the resultant matrix is

	a	b	c	d
a	0	$\infty$	3	$\infty$
b	2	0	5	$\infty$
c	9	7	0	1
d	6	$\infty$	9	0

Step 3 : consider the shortest path distance through vertex c consider 6th row & 6th column from matrix obtained in step 2

$$\text{So, } (a, c) = 3 \quad \& \quad (c, a) = 9$$

$$(a, a) = \min \{ (a, a), (a, c) + (c, a) \}$$
$$= \min \{ 0, 12 \}$$

$$\boxed{(a, a) = 0}$$

$$(a,c) = 3 \quad \wedge \quad (c,b) = 7$$

$$(a,b) = \min \{ (a,b), (a,c) + (c,b) \}$$

$$= \min \{ \infty, 10 \}$$

$$\boxed{(a,b) = 10}$$

$$(a,c) = 3 \quad \wedge \quad (c,d) = 1$$

$$(a,d) = \min \{ (a,d), (a,c) + (c,d) \}$$

$$= \min \{ \infty, 4 \}$$

$$\boxed{(a,d) = 4}$$

$$(b,c) = 5 \quad \wedge \quad (c,a) = 9$$

$$(b,a) = \min \{ (b,a), (b,c) + (c,a) \}$$

$$= \min \{ 2, 14 \}$$

$$\boxed{(b,a) = 2}$$

$$(b,c) = 5 \quad \wedge \quad (c,b) = 7$$

$$(b,b) = \min \{ (b,b), (b,c) + (c,b) \}$$

$$= \min \{ 0, 12 \}$$

$$\boxed{(b,b) = 0}$$

$$(b,c) = 5 \quad \wedge \quad (c,d) = 1$$

$$(b,d) = \min \{ (b,d), (b,c) + (c,d) \}$$

$$= \min \{ \infty, 6 \}$$

$$\boxed{(b,d) = 6}$$

$$(d,c) = 9 \quad \wedge \quad (c,a) = 9$$

$$(d,a) = \min \{ (d,a), (d,c) + (c,a) \}$$

$$= \min \{ 6, 18 \}$$

$$\boxed{(d,a) = 6}$$

$$(d,c) = 9 \quad \wedge \quad (c,b) = 7$$

$$(d,b) = \min \{ (d,b), (d,c) + (c,b) \}$$

$$= \min \{ \infty, 16 \}$$

$$(d, b) = 16$$

$$(d, c) = 9 \quad \text{and} \quad (c, d) = 7$$

$$(d, d) = \min \{ (d, d), (d, c) + (c, d) \}$$

$$= \min \{ 0, 10 \}$$

$$\boxed{(d, d) = 0}$$

Hence the resultant matrix is

$$\begin{array}{l} a \quad b \quad c \quad d \\ \hline a & 0 & 10 & 3 & 4 \\ b & 2 & 0 & 5 & 6 \\ c & 9 & 7 & 0 & 1 \\ d & 6 & 16 & 9 & 0 \end{array}$$

Step 4: consider vertex 'd', consider  $d^{\text{th}}$  row

&  $d^{\text{th}}$  column

$$(d, d) = 4 \quad \cancel{2 \neq d}$$

$$(a, d) = 4 \quad \text{and} \quad (d, a) = 6$$

$$(a, a) = \min \{ 0, 10 \}$$

$$\boxed{(a, a) = 0}$$

$$(a, d) = 4 \quad \text{and} \quad (d, b) = 16$$

$$d^{\text{th}} \text{ col } (a, b) = \min \{ 10, 20 \}$$

$$\boxed{(a, b) = 10}$$

$$(a, d) = 4 \quad \text{and} \quad (d, c) = 9$$

$$(a, c) = \min \{ 3, 13 \}$$

$$\boxed{(a, c) = 3}$$

$$(b, d) = 6 \quad \text{and} \quad (d, a) = 6$$

$$(b, a) = \min \{ 2, 12 \}$$

$$\boxed{(b, a) = 2}$$

$$(b, d) = 6 \quad \text{and} \quad (a, b) = 16$$

$$(b, b) = \min \{ 0, 12 \}$$

$$\boxed{(b, b) = 0}$$

$$(b, d) = 6 \quad \text{and} \quad \frac{(a, c) = 1}{(b, c) = \min \{ 5, 7 \}}$$

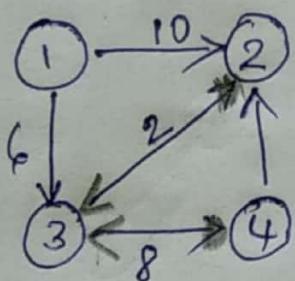
$$\boxed{(b, c) = 5}$$

$$\begin{array}{l}
 (c,d) = 1 \quad \text{and} \quad (d,a) = 6 \\
 (c,a) = \min \{ 9, 1 \} \quad |(c,a) = 1| \\
 (c,d) = 5 \quad \text{and} \quad (d,b) = 16 \\
 (c,b) = \min \{ 7, 16 \} \quad |(c,b) = 7| \\
 (c,d) = 1 \quad \text{and} \quad (d,c) = 9 \\
 (c,c) = \min \{ 0, 10 \} \quad |(c,c) = 0|
 \end{array}$$

Resultant matrix is

$$\begin{matrix}
 & a & b & c & d \\
 a & 0 & 10 & 3 & 4 \\
 b & 2 & 0 & 5 & 6 \\
 c & 7 & 7 & 0 & 1 \\
 d & 6 & 16 & 9 & 0
 \end{matrix}$$

2. Solve the All pair shortest path problem for the given digraph

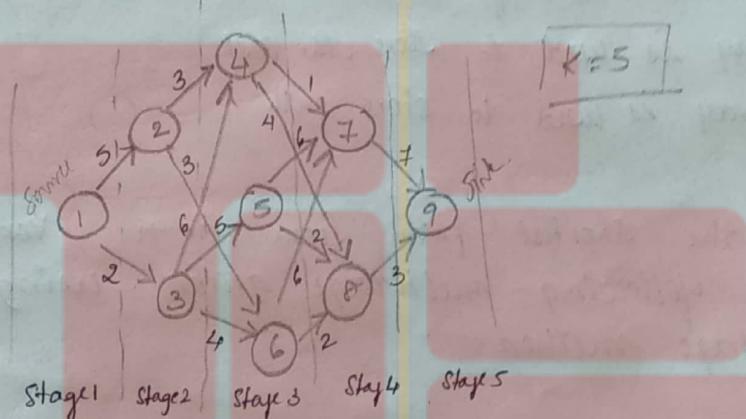


## Multi Stage Graph Problem

$$G = (V, E)$$

- Multi stage Graph is a directed graph where the vertices are partitioned into 'k' stages Where  $k \geq 2$  (vertices are repeated)
- Every stage contains disjoint sets  $V_i$ ,  $1 \leq i \leq k$  (No common vertices)
- At every stage number of vertices should not exceed 'k'
- $|V_1| = |V_k| = 1$
- (No of vertices in 1st stage) = (No of vertices in  $k^{\text{th}}$  stage)

Example



Multi stage problem is to find the shortest path from the source vertex 'S' to the sink vertex 'f'

To find the shortest path we have two approaches

- (i) Forward approach
- (ii) Backward approach.

### 1. FORWARD APPROACH

General procedure       $n$ : no of vertices

$$c[n] = 0;$$

$$P[n] = 0; \quad 'P' \text{ holds shortest path.}$$

for  $j=n-1$  down to 1      start from vertex  $i-1$   
 Let  $r \in V$ , for each edge  $\langle j, r \rangle$   $r: \text{vertex } \in V$  there will  
 be a  $\rightarrow$  edge  
 minimum of  $a[j, r] + c[r]$        $a[j, r]$ : respective weight.

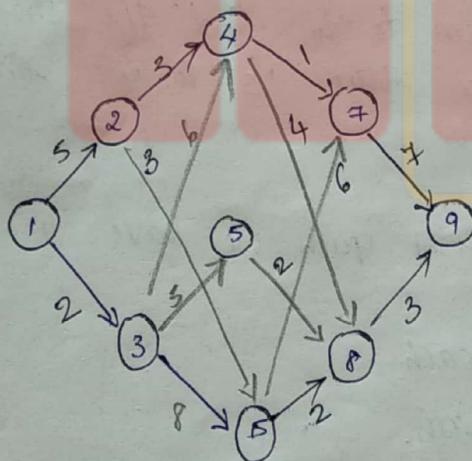
$$c[j] = u[j, r] + c[r];$$

$$P[j] = r$$

Find for

- When  $n$ : no of vertices
- C array is used to store the cost
- P array is used to store path

1. Obtain the shortest path from  $s \rightarrow t$ , vertex 1 to 9  
 in the following multistage graph using  
 multistage method.



Initialization :  $n=9$ ,  $s=1$ ,  $t=9$ ,  $c[9]=0$ ,  $P[9]=9$

$$\begin{aligned} \text{1) Vertex 8 : } c[8] &= \min \{ a[8, 9] + c[9] \} \\ &= \min \{ 3 + 0 \} \end{aligned}$$

$$C[8] = 3$$

$$P[8] = 9$$

2. Vertex 7 :  $C[7] = \min \{ a[7, 9] + C[9], 7 + 0 \}$

$$= \min \{ 7 + 0 \}$$

$C[7] = 7$	$P[7] = 9$
------------	------------

3. Vertex 6 :  $C[6] = \min \{ a[6, 7] + C[7], a[6, 8] + C[8] \}$ ,  
6 dy 7 from 6 to 7

$$C[6] = \min \{ 6 + 7, 6 + 8 \}$$

$$= \min \{ 13, 8 \}$$

$C[6] = 5$	$P[6] = 8$ <small>(because it plus minimum path)</small>
------------	--

4. Vertex 5 :  $C[5] = \min \{ a[5, 7] + C[7], a[5, 8] + C[8] \}$

$$C[5] = \min \{ 6 + 7, 6 + 8 \}$$

$$C[5] = \min \{ 13, 8 \}$$

$C[5] = 5$	$P[5] = 8$
------------	------------

5. Vertex 4 :  $C[4] = \min \{ a[4, 7] + C[7], a[4, 8] + C[8] \}$

$$C[4] = \min \{ 8 + 7, 8 + 8 \}$$

$$= \min \{ 15, 16 \}$$

$C[4] = 7$	$P[4] = 8$
------------	------------

6. Vertex 3 :  $C[3] = \min \{ a[3, 4] + C[4], a[3, 5] + C[5], a[3, 6] + C[6] \}$

$$C[3] = \min \{ 6 + 7, 6 + 8, 5 + 8 \}$$

$$= \min \{ 13, 14, 13 \}$$

$C[3] = 10$	$P[3] = 5$
-------------	------------

$$7. \text{ Vertex } 2 : c[2] = \min \left\{ \begin{array}{l} a[2, 4] + c[4], \\ a[2, 6] + c[6] \end{array} \right\}$$

$$c[2] = \min \{ 3+7, 3+5 \}$$

$$= \min \{ 10, 8 \}$$

$$\boxed{c[2] = 8}$$

$$\boxed{p[2] = 6}$$

$$8. \text{ Vertex } 1 : c[1] = \min \left\{ \begin{array}{l} a[1, 2] + c[2], \\ a[1, 3] + c[3] \end{array} \right\}$$

$$c[1] = \min \{ 5+8, 2+10 \}$$

$$\boxed{c[1] = 12}$$

$$\boxed{p[1] = 3}$$

Note : The cost obtained w.r.t the source vertex i.e.,  $c[S]$  is the shortest path from source to destination.

Therefore in the above problem shortest path distance from 1 to 9 is  $c[1] = 12$

To obtain the shortest path which leads to the cost 12 use the array P as follows

P	1	2	3	4	5	6	7	8	9
	3	6	5	8	8	8	9	9	9

(i) To print the path: print the source first

(ii) print  $P[1] = 3$        $1 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 9$

(iii) print  $P[3] = 5$

(iv) print  $P[5] = 8$

(v) print  $P[8] = 9$

Stop as '9' is sink

30/04/18

## Multistage graph problem

is used to find the shortest path from a node in stage one to a node in stage k. Though source & sink nodes nor not specific do consider the vertex at stage 1 as source & at stage-k as sink.

### Backward approach

The approach uses the following method.

$$C[1] = 0$$

$$P[1] = 1;$$

for  $j \leftarrow 2$  to  $\infty$  do.

for  $r \in V$ , for each edge  $\langle r, j \rangle \in E$ , find

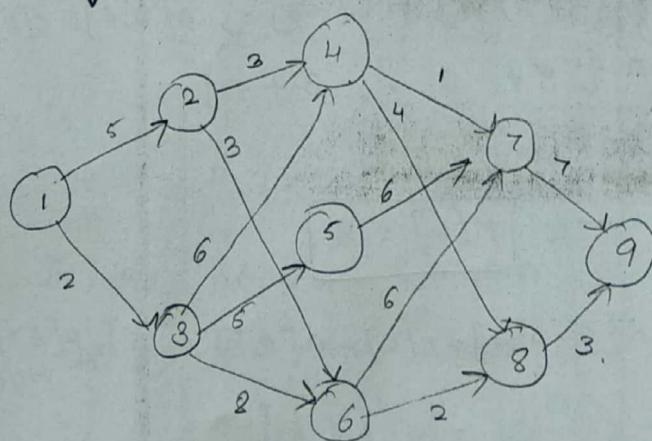
minimum of  $a[r,j] + C[r]$  then

$$C[j] \leftarrow a[r,j] + C[r];$$

$$P[j] \leftarrow r;$$

end for  $r$

Apply multistage graph problem to the following graph using backward approach.



→ In the above graph  $n=9$ ,  $s=1$ ,  $d=9$  &  
 $C[1]=0$ ,  $P[1]=1$

Vertex 2:  $c[2] = \min \{a[1, 2] + c[1]\}$

$$= \min \{5 + 0\}$$

$$\boxed{c[2] = 5} \quad \& \quad \boxed{P[2] = 1}$$

Vertex 3:  $c[3] = \min \{a[1, 3] + c[1]\}$

$$= \min \{2 + 0\}$$

$$\boxed{c[3] = 2} \quad \& \quad \boxed{P[3] = 1}$$

Vertex 4:  $c[4] = \min \{a[2, 4] + c[2], a[3, 4] + c[3]\}$

$$= \min \{3 + 5, 6 + 2\}$$

$$\boxed{c[4] = 8} \quad \& \quad \boxed{P[4] = 2 \text{ or } P[4] = 3}$$

Vertex 5:  $c[5] = \min \{a[3, 5] + c[3]\}$

$$= \min \{5 + 2\}$$

$$\boxed{c[5] = 7} \quad \& \quad \boxed{P[5] = 3}$$

Vertex 6:  $c[6] = \min \{a[3, 6] + c[3], a[2, 6] + c[2]\}$

$$= \min \{8 + 2, 3 + 5\}$$

$$= \min \{10, 8\}$$

$$\boxed{c[6] = 8} \quad \& \quad \boxed{P[6] = 2}$$

Vertex 7:  $c[7] = \min \{a[4, 7] + c[4], a[5, 7] + c[5], a[6, 7]\}$

$$= \min \{1 + 8, 6 + 2, 6 + 8\}$$

$$= \min \{9, 13, 14\}$$

$$\boxed{c[7] = 9} \quad \& \quad \boxed{P[7] = 4}$$

vertex 8:  $c[8] = \min \{ c[4, 8] + c[u], c[5, 8] + c[v], c[6, 8] + c[w] \}$

$$= \min \{ 4 + 8, 2 + 7, 4 + 8 \}.$$

$$= \min \{ 12, 9, 10 \}$$

$$\boxed{c[8] = 9} \quad \& \quad \boxed{P[8] = 5}$$

vertex 9:  $c[9] = \min \{ c[7, 9] + c[z], c[8, 9] + c[y] \}$

$$= \min \{ 7 + 9, 8 + 9 \}$$

$$= \min \{ 16, 18 \}$$

$$\boxed{c[9] = 12} \quad \& \quad \boxed{P[9] = 8}$$

→ The shortest path distance from source to

sink is  $c[9] = 12$

→ To print the path use the array 'P'.

P	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
	1	1	1	2   3	3	2	u	5   8	/   /

First, print the sink node: 9

Then print  $P[9]: 8$

$P[8]: 5$

$P[5]: 3$

$P[3]: 1$  (stop box 1 is the source node)

∴ path is

$9 \leftarrow 8 \leftarrow 5 \leftarrow 3 \leftarrow 1$

Notes: In the above example  $P[4]$  has 2 entries (2 & 3) this may lead to a different paths if & only iff  $P[4]$  was used while printing the path.

→ If  $P[4]$  was used to print the path then we need to consider 2 paths with the same shortest distance.

algorithm for both the approaches.

Algorithm Forward-graph ( $G, n, a, s, d$ )

{  $c[n] = 0$

$P[n] = n;$

for  $j \leftarrow n-1$  down to 1 do

for  $v \in V$ , for each edge  $\langle j, v \rangle \in E$ , find

minimum of  $a[j, v] + c[v]$  then

$c[j] \leftarrow a[j, v] + c[v];$

$P[j] \leftarrow v;$

end for

Print (shortest path distance is  $c[j]$ );

$j = s;$

while ( $j \neq d$ )

print ( $j, \rightarrow$ );

$j \leftarrow P[j];$

end while

} Print ( $j$ );

Backward approach:

Algorithm Backward-graph ( $G, n, a, s, d$ )

{

$c[s] = 0$

$P[s] = s;$

for  $j \leftarrow 2$  to  $n$

for  $\sigma \in V$ , for each edge  $e \in \sigma, j \geq e \in E$ , find  
minimum of  $a[\sigma, j] + c[\sigma]$  then

$c[j] \leftarrow a[\sigma, j] + c[\sigma];$

$p[j] \leftarrow \sigma;$

end for

Print (shortest path distance is  $c[n]$ );

$j = d;$

while ( $j \neq \infty$ )

print ( $j, \leftarrow$ );

$j \leftarrow p[j];$

end while

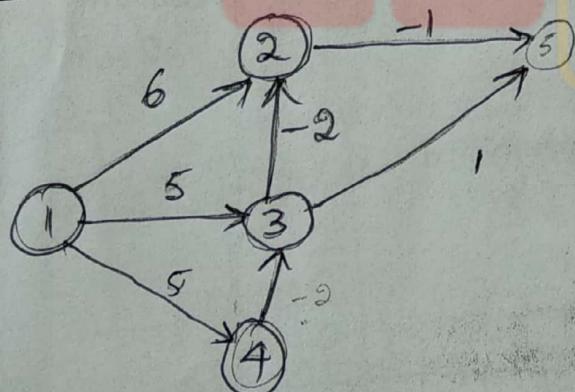
print ( $j$ );

g.

Bellman Ford Algorithm

source node = 1

ex:



shortest Path distance

$1 \rightarrow 1 = 0$

$1 \rightarrow 2 = 1$

$1 \rightarrow 3 = 3$

$1 \rightarrow 4 = 5$

$1 \rightarrow 5 = 0$

Bellman Ford algorithm is to find shortest path from a source vertex to all other vertices in a given directed weighted graph.  
which has negative edges.

algorithm (use the following design)

\* for  $\leftarrow 1$  to  $n$  do

{

$d[i] = \text{aff}, i];$  // distance array

$P[i] = s;$  // path array

g

\* for  $i \leftarrow 1$  to  $n-1$  do // iterations.

    for each edge  $e \in u, v \in E$  where  $u \neq v$

<sup>source</sup>L

            if ( $d[u] + a[u, v] < d[v]$ )

<sup>source</sup>  $\rightarrow$  vector

$d[v] = d[u] + a[u, v];$

$P[v] = u;$

                end if

        end for

Apply Bellman Ford algorithm to the following graph

→ initializing the distance matrix  $d$  & path array  $P$ .

	(1)	(2)	(3)	(4)	(5)
0	0	6	5	5	∞
1	0	6	5	5	∞
2	0	6	5	5	∞
3	0	6	5	5	∞
4	0	6	5	5	∞
5	0	6	5	5	∞

	(1)	(2)	(3)	(4)	(5)
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
5	1	1	1	1	1

according to graph given as  $\Rightarrow P[i] = s$  ( $s = 1$ )

Since there are 5 vertices, 4 iterations are required. we can stop the iteration if we get same value for  $d$  &  $P$  arrays at that of a previous iteration.

Iteration 1:

Edge C	$d[v] = \min\{d[v], d[u] + a[u, v]\}$	$P[v] = u$
$\langle 1, 2 \rangle$	$d[2] = \min\{6, 0 + 6\} = 6$	-
$\langle 1, 3 \rangle$	$d[3] = \min\{5, 0 + 5\} = 5$	-
$\langle 1, 4 \rangle$	$d[4] = \min\{5, 0 + 5\} = 5$	-
$\langle 2, 5 \rangle$	$d[5] = \min\{6, 6 + (-1)\} = 5$	$P[5] = 2$
$\langle 3, 2 \rangle$	$d[2] = \min\{6, 5 + (-2)\} = 3$	$P[2] = 3$
$\langle 3, 5 \rangle$	$d[5] = \min\{5, 5 + (1)\} = 5$	-
$\langle 4, 3 \rangle$	$d[3] = \min\{5, 5 + (-3)\} = 3$	$P[3] = 4$

update #1

	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{5\}$
$d$	0	6	5	5	10
Iteration 1	0	3	3	5	5

	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{5\}$
$P$	1	1	1	1	1
	1	3	4	1	2

Repeat the same for Iteration 2 to 4, After the iterations, we get the  $d$  &  $P$  arrays as below.

	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{5\}$
$d$	0	6	5	5	10
Iteration -1	0	3	3	5	5
Iteration -2	0	1	3	5	2
Iteration -3	0	1	3	5	0
Iteration -4	0	1	3	5	0

	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{5\}$
$P$	1	1	1	1	1
I-1	1	3	4	1	2
I-2	1	3	4	1	2
I-3	1	3	4	1	2
I-4	1	3	4	1	2

Now, we can get the shortest path distance and shortest path using the final arrays  $d$  &  $P$  as shown below.

$d$	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{5\}$
0	1	3	4	5	0

From	To	Distance	Path
1	1	0	$1 \rightarrow 1$
1	2	1	$1 \rightarrow 4 \rightarrow 3 \rightarrow 2$
1	3	3	$1 \rightarrow 4 \rightarrow 3$
1	4	5	$1 \rightarrow 4$
1	5	0	$1 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 5$

# Module - IV

## Dynamic Programming

### 0/1 Knapsack Problem

'n' objects,  $p_i$  (Profit) &  $w_i$  (Weight)  
 'm' knapsack capacity such that  $\sum p_i x_i$  is maximum &  $\sum w_i x_i \leq m$ ,  $x_i \rightarrow 0 \text{ or } 1$ .

$i$	0	1	2	3	4
0	0	0	0	0	0
1	0		x		
2	0				
3	0				

Entries in table denote profit.  
 $N+1$  : rows ( $N$ : objects)  
 $M+1$  : column ( $M$ : knapsack capacity)  
 $V[i][j]$  : profit.

Profit table [ $V$ ]

if  $i=0$ ; no objects  
 if  $j=0$ ; knapsack capacity is 0

Step 1 :  $V[i][j] = 0$

if  $i=0$  (or)  $j=0$

Step 2 :  $i$ th object weight is greater than current  $j$  of  $i$ th object, then  $i$ th object must be rejected.

$$V[i, j] =$$

$V[i-1, j]$   
 previous row  
 same column.  $i$ th object not selected.

Step 3 :

if  $w_i \leq j$ ,

$$V[i, j] = \max \left\{ V[i-1, j], V[i-1, j-w_i] + p_i \right\}$$

$i$ th object is selected.

Profit incurred (by selecting  $i$ th object) by making sufficient space.

$V[i-1, j]$  : profit by not selecting  $i$ th object.

$V[i-1, j-w_i] + p_i$  : profit by selecting  $i$ th object.

(make space)

Step 4 : After computing all the entries in profit table  $V$ , the last entry  $V[n, m]$  gives the max. profit

Obtained for the problem.

Note:

Construct the  $n+1$  rows &  $m+1$  columns where the index starts from 0 to  $n$  and 0 to  $m$  respectively. Every entry in the table denotes the profit w.r.t  $i^{th}$  object and knapsack capacity  $j$ .

Solve the following 0/1 knapsack problem using dynamic programming.

Item	Weight	Profit
1	2	12
2	1	10
3	3	20

$m=4$

Step 1 : Construct the profit table of  $N+1$  rows and  $M+1$  columns i.e., 4 rows, 5 columns ( $n=3, m=4$ )

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	12	12	12
2	0	10	12	22	22
3	0	10	12	22	30

Query all is sub  
problem

Step 2 : Initialize  $V[i, j] = 0$  When  $i=0$  or  $j=0$

i.e.,  $V[i, j] = 0$  for  $i=0$  and  $j=0$  to 4

$V[i, j] = 0$  for  $j=0$  and  $i=0$  to 3

Updating this in the above table.

Step 3 : Starting from  $i=1$ ,

where  $i=1, W[i] = 2, P[i] = 12$

Using the below relation compute the entries.

$$V[i, j] = \begin{cases} V[i-1, j] & , w_i > j \\ \max \{ V[i-1, j], V[i-1, j-w_i] + p_i \} & , w_i \leq j \end{cases}$$

Computation is shown below.

$w_i$	$j$	$i=1$	$V[i, j]$
2	1	$V[1, 1] = V[0, 1] = 0$	
2	2	$V[1, 2] = \max \{ V[0, 2], V[0, 0] + 12 \} = \max \{ 0, 12 \} = 12$	
2	3	$V[1, 3] = \max \{ V[0, 3], V[0, 1] + 12 \} = \max \{ 0, 12 \} = 12$	
2	4	$V[1, 4] = \max \{ V[0, 4], V[0, 2] + 12 \} = \max \{ 0, 12 \} = 12$	

Update the above entries in the profit table.

Step 4: Considering  $i=2$ , then  $w[i]=1$ ,  $p[i]=10$   
using the above recurrence relation, computation is  
as follows.

$w_i$	$j$	$i=2$	$V[i, j]$
1	1	$V[2, 1] = \max \{ V[1, 1], V[1, 0] + 10 \} = \max \{ 0, 10 \} = 10$	
1	2	$V[2, 2] = \max \{ V[1, 2], V[1, 1] + 10 \} = \max \{ 12, 12 \} = 12$	
1	3	$V[2, 3] = \max \{ V[1, 3], V[1, 2] + 10 \} = \max \{ 12, 22 \} = 22$	
1	4	$V[2, 4] = \max \{ V[1, 4], V[1, 3] + 10 \} = \max \{ 12, 22 \} = 22$	

Update the above entries in the profit table

Step 5: Considering

$w_i$	$j$	$V[i, j]$
3	1	$V[3, 1] = V[2, 1] = 10$
3	2	$V[3, 2] = V[2, 2] = 12$
3	3	$V[3, 3] = \max \left\{ V[2, 3], V[2, 0] + 20 \right\} = \max \left\{ 22, 20 \right\} = 22$
3	4	$V[3, 4] = \max \left\{ V[2, 4], V[2, 1] + 20 \right\} = \max \left\{ 22, 30 \right\} = 30$

Update the above entries in the profit table. Hence, the maximum profit obtained for the problem is  $V[n, m]$  i.e.,  $V[3, 4]$

$$\therefore \text{profit} = 30$$

The procedure for finding which objects are selected is as follows.

$$\rightarrow V[3, 4] \neq V[2, 4]$$

$\alpha[3] = 1$   $\wedge$   $i=2$ ,  $j=4-3=1$

object selected

$$\rightarrow V[2, 1] \neq V[1, 1]$$

$\alpha[2] = 1$   $\wedge$   $i=1$   $\wedge$   $j=1-1=0$

object selected

```

for i ← 1 to n do
    x[i] ← 0

i = n
j = m
while (i != 0 AND j != 0)
    if v[i, j] != v[i-1, j] then
        x[i] ← 1; // i-th object selected
        j ← j - w[i]; // remaining knapsack capacity.
    end if

```

$i \leftarrow i-1$ ; // Move to the previous  $i$ th object.

end while

```

for i ← 1 to n do
    if (x[i] == 1)
        write "object selected is", i
    end if
end for

```

Algorithm of 0/1 knapsack problem & Analysis.

Algorithm knapsack (n, m, w, f)

// To solve 0/1 knapsack problem using dynamic programming.

// 'n' no of objects, 'm' knapsack capacity,  
// 'w' weight of objects, 'P[1:n]' profit of objects

// O/P : To return the profit table V

$V[0:n, 0:m]$

Where,  $V[n, m]$  denotes maximum profit.

for i:0 to n do ( $\geq$ )

for j:0 to m do

if  $i=0$  &  $j=0$

if  $i \neq 0$  &  $j \neq 0$  then

$v[i, j] \leftarrow 0$

Else if  $w[i] > j$  then

$$V[i, j] \leftarrow V[i-1, j]$$

Else

$$V[i, j] \leftarrow \max \{ V[i-1, j], V[i-1, j-w[i]] + p_{i,j} \}$$

End if

End for  $j$

End for  $i$

return  $V$ ;

g

dualysis

- IIP size : No of objects and knapsack capacity i.e.,  $n$  and  $m$  respectively.

- Basic operation : The basic operation here is to compute the values of  $V[i, j]$

The statement  $V[i, j] = \max \{ V[i-1, j], V[i-1, j-w[i]] + p_{i,j} \}$  is executed maximum times, therefore this is the

B.O.

- Setting up the summation formula that indicates how many times the B.O. is executed.

$$T(n) = \sum_{i=0}^n \sum_{j=0}^m 1$$

$$= \sum_{i=0}^n (m+1)$$

$$= \sum_{i=0}^n m+1$$

$$= (m+1) \sum_{i=0}^n 1.$$

$$= (m+1) (n+1)$$

$$T(n) = mn + m+n+1$$

$T(n) \approx mn$  (for every large value of  $m+n$ )

$$T(n) \in \Theta(mn)$$

Assignment: Solve the problem using dynamic programming  
Where  $n=4$  and  $m=5$

Item	weight	profit
1	2	12
2	1	10
3	3	20
4	2	15

### Travelling Salesman Problem (TSP)

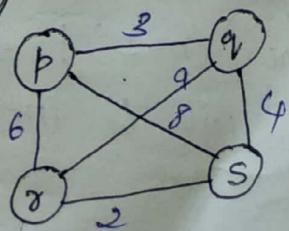
Problem statement: Given  $n$  cities the travelling sales man has to start from one city (source) and visit all  $(n-1)$  cities exactly once. And return to the city from where he has started. The objective of this problem is to find a route through the cities that minimizes the cost there by maximizing profit.

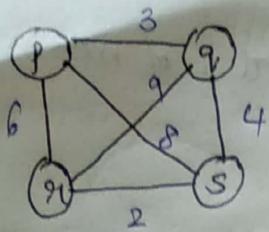
### Design

→ consider a graph (undirected),  
Where vertices →  $n$  cities  
weight of edge → 'cost' of reaching/travelling from one city to another.

- 2 ways to solve.
  - \* Brute Force approach. (Trial & Error with our knowledge)
  - \* Dynamic Programming.

- Example





Optics: P, Q, R, S  
Source: P

by Brute force.

The paths are:

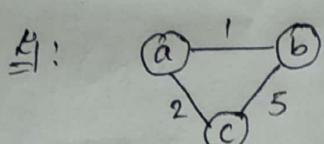
- \*  $P \xrightarrow{3} Q \xrightarrow{9} R \xrightarrow{2} S \xrightarrow{6} P$  (Cost = 22)
- \*  $P \xrightarrow{3} Q \xrightarrow{4} S \xrightarrow{2} R \xrightarrow{6} P$  (Cost = 15)
- \*  $P \xrightarrow{6} R \xrightarrow{2} S \xrightarrow{4} Q \xrightarrow{3} P$  (Cost = 15)
- \*  $P \xrightarrow{8} S \xrightarrow{4} Q \xrightarrow{9} R \xrightarrow{6} P$  (Cost = 27)
- \*  $P \xrightarrow{6} R \xrightarrow{9} Q \xrightarrow{4} S \xrightarrow{8} P$  (Cost = 27)
- \*  $P \xrightarrow{8} S \xrightarrow{2} R \xrightarrow{9} Q \xrightarrow{3} P$  (Cost = 22)

Note: In Brute force we take different permutations of paths and select the one, which has the least cost.

- In general, if we have 2 cities
- E.g.:  $a \xrightarrow{8} b$ , the paths are  $a \xrightarrow{8} b$ ,  $b \xrightarrow{8} a$

$$\therefore \text{No of paths} = 1.$$

- Consider 3 cities.



$$a \xrightarrow{1} b \xrightarrow{5} c \xrightarrow{2} a$$

$$a \xrightarrow{2} c \xrightarrow{5} b \xrightarrow{1} a$$

$$\therefore \text{No of paths} = 2$$

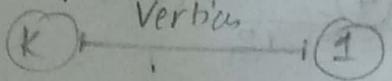
- Consider 4 cities.  $\therefore \text{No of paths} = 6$

$\therefore$  In general if we have  $n$  cities, then we have  $(n-1)!$  paths

Thus an algorithm that uses Brute force approach need to determine  $(n-1)!$  paths. Which means, time complexity will be  $(n-1)!$

# TSP Using dynamic programming

- The various points do be remembered.
- The tour may start from any of the edge  $(1, k)$  where  $k \in V - \{1\}$  remove source vertex  $\uparrow$  source city
- There is a path from  $k$  to 1.  
The path from vertex  $k$  to 1 goes through each vertex in  $V - \{1, k\}$
- Let  $g(i, S)$  - denotes the shortest path from vertex 'i' moving through the vertices in  $S$  and then terminating at vertex 1.



$$\rightarrow g(1, V - \{1\}) = \min_{2 \leq k \leq n} \{ C_{1k} + g(k, V - \{1, k\}) \}$$

Shortest path from 1  
moving through vertices  $V - \{1\}$   
and terminating at vertex 1.  
 $k =$  start from 1.  
Minimum cost vertex from source vertex 1.

$$\rightarrow \text{In General}$$

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

$|S| < n-1$  (set of vertices),  $i \neq 1$

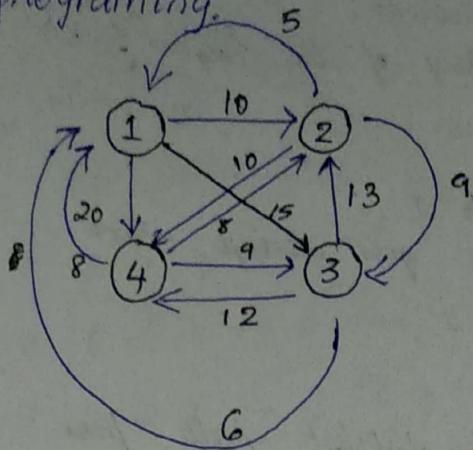
$i \notin S$ ,  $1 \notin S$

→ Base Case

$$g(i, \emptyset) = C_{i1} \quad \text{for } 2 \leq i \leq n \quad |S| = 0$$

$i \rightarrow 1$   
vertex set

Solve the following problem using dynamic programming.



Cost adjacency matrix:

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

In the above graph, we have '4' cities and the problem is solved using the relations as shown below.

$\phi$ : No intermediate cities.

case(i) : Here  $S = \emptyset$  (NULL set) i.e.,  $|S| = 0$   
in this case we solve the following relation

$$2 \leq i \leq n \quad |S| = 0$$

$$g(i, \phi) = C_{i,1} \quad \text{for}$$

$$g(2, \phi) = C_{2,1} = 5$$

$$g(3, \phi) = C_{3,1} = 6$$

$$g(4, \phi) = C_{4,1} = 8$$

case(ii) : Consider  $|S| = 1$  in this case  
 $i \neq 1$ ,  $i \notin S$ ,  $1 \notin S$ . Using this solve the following relation

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

When  $i = 2$  we have  $S = \{3\} \quad \& \quad S = \{4\}$

$$g(2, \{3\}) = \min_{j=3} \{ C_{2,3} + g(3, \emptyset) \}$$

$$= \min \{ 9 + 6 \} = 15$$

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

When  $i=3$ , we have  $S = \{2\}$  &  $S = \{4\}$

$$g(3, \{2\}) = \min_{j=2} \{ C_{3,2} + g(2, \emptyset) \} = \min \{ 13 + 5 \}$$

$$= 18$$

$$g(3, \{4\}) = \min_{j=4} \{ C_{3,4} + g(4, \emptyset) \} = \min \{ 12 + 8 \}$$

$$= 20$$

When  $i=4$ , we have  $S = \{2\}$  &  $S = \{3\}$   
 shortest path from 4 to travel the graph come back to source.

$$g(4, \{2\}) = \min \{ C_{4,2} + g(2, \{2\}) \} = \min \{ C_{4,2} + g(2, \emptyset) \}$$

$$= \min \{ 8 + 5 \}$$

$$g(4, \{3\}) = \min \{ C_{4,3} + g(3, \emptyset) \} = \min \{ 9 + 6 \}$$

$$g(4, \{3\}) = 15$$

$$g(4, \{2\}) = 13$$

case (iii) : Consider  $|S| = 2$  (don't include 1, 2, i.e.  $i=2$ )

When  $i=2$ , we have  $S = \{3, 4\}$ ,  $i \notin S$   
 $S$  doesn't include source &  $i$   
 $i(S) = 2$  (means  $S$  should not include 1, 2)

$$i=2, S = \{3, 4\}$$

$$g(2, \{3, 4\}) = \min_{j=3,4} \{ C_{2,j} + g(3, \{j\}) \}$$

$$\text{consider edge } (2, 3) \quad (C_{24} + g(4, \{3\}))$$

$$= \min \{ 9 + 20, 10 + 15 \} = \min \{ 29, 25 \}$$

$$g(2, \{3, 4\}) = 25$$

case (iv) : When  $i=3$ ,  $S = \{2, 4\}$

$$g(3, \{2, 4\}) = \min_{j=2,4} \{ C_{3,j} + g(2, \{j\}), C_{3,j} + g(4, \{j\}) \}$$

$$= \min \{ 13 + 18, 12 + 13 \} = 25$$

case(V) : When  $i=4 = \{2, 3\}$

$$g(4, \{2, 3\}) = \min_{j=2,3} \{ C_{4,2} + g(\{2, \{3\}\}), C_{4,3} + g(\{3, \{2\}\}) \}$$

$$= \min \{ 8 + 15, 9 + 18 \} = \min \{ 23, 27 \}$$

$$g(4, \{3\}) = 23$$

We should not consider  $|S| = 3$  because  $|S| < n-1$ , i.e.,  $|S| < 4-1$ ,  $|S| < 3$  therefore use the final relation to get the optimal tour.

$V$ : set of vertices

$$g(\{1, V-\{1\}\}) = \min_{2 \leq k \leq n} \{ C_{1,k} + g(k, V-\{1\}, \{k\}) \}$$

$$g(\{1, V-\{1, 2\}\}) = \min \{ C_{1,2} + g(2, V-\{1, 2\}, \{2\}), C_{1,3} + g(3, V-\{1, 3\}, \{3\}), C_{1,4} + g(4, V-\{1, 4\}, \{4\}) \}$$

$$V = \{1, 2, 3, 4\}$$

$$g(\{1, \{1, 2, 3, 4\} - \{1\}\}) = \min \{ C_{1,2} + g(2, \{3, 4\}), C_{1,3} + g(3, \{2, 4\}), C_{1,4} + g(4, \{2, 3\}) \}$$

$$g(\{2, \{2, 3, 4\}\}) = \min \{ 10+25,$$

Starting from 1 moving to all vertices  $\{2, 3, 4\}$

and back to source

$$15+25,$$

$$20+23$$

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

$$g(\{1, \{2, 3, 4\}\}) = 35$$

$\therefore$  the shortest path distance for the above TSP problem is 35

$$1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$$

To get the path just retain each value of  $g(i, s)$  that gives the minimum value on RHS

$$\begin{aligned}
 g(1, \{2, 3, 4\}) &= C_{12} + g(2, \{3, 4\}), \\
 &\quad \vdots \qquad \qquad \qquad \overbrace{C_{24} + g(4, \{3\})}^{\text{minimum}} \\
 &\quad \vdots \qquad \qquad \qquad \overbrace{C_{43} + g(3, \emptyset)}^{\text{minimum}} \\
 &\quad \vdots \qquad \qquad \qquad \overbrace{C_{31}}^{\text{minimum}} \\
 &\quad \downarrow \qquad \downarrow \qquad \downarrow \qquad \downarrow \\
 \text{Shortest path: } & \boxed{1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1} = 35
 \end{aligned}$$

### Dijkstra's Algorithm (Greedy Method)

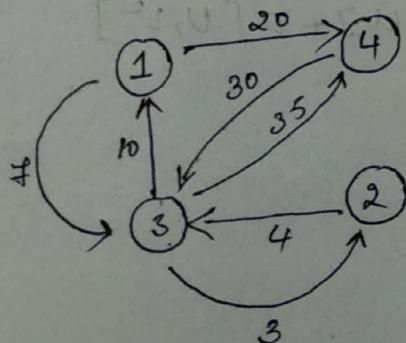
This algorithm is used to find the shortest path from a source vertex to all other vertices in a given graph  $G = \{V, E\}$ . This is also referred as "single source shortest path problem".

Note: Dijkstra works only for positive edge weights whereas Bellman Ford works for both +ve and -ve edge weights.

Floyd's: All pair shortest path

Dijkstra: Single source

Source vertex be 1



$$1 \rightarrow 1 = 0$$

$$1 \rightarrow 2 = 10$$

$$1 \rightarrow 3 = \infty$$

$$1 \rightarrow 4 = 20$$

Cost

Adjacency Matrix:

	1	2	3	4
1	0	$\infty$	$\infty$	20
2	$\infty$	0	4	$\infty$
3	10	3	0	35
4	$\infty$	$\infty$	30	0

Algorithm Dijkstra ( $n, a, \text{source}, d$ )

1) purpose: To solve single source shortest path problem

2) Input: No of vertices,  $a$ : cost adjacency matrix  
source: source vertex

3) Output: The matrix  $d[1:n]$  which gives shortest distance from source to all other vertices.

for  $i \leftarrow 1$  to  $n$  do source row w.r.t all col.

$d[i] \leftarrow a[\text{Source}][i]$  vertices which are not visited:  
 $p[i] \leftarrow \text{source}$ ; // tells vertices which are visited:  
 $s[i] \leftarrow 0$ ; // edge path

end for

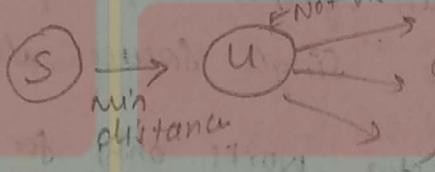
$s[\text{source}] \leftarrow 1$ ; // add source to  $S$   
Source vertex is visited.

for  $i \leftarrow 1$  to  $n-1$  do // to find  $n-1$  paths.

Find  $u$  &  $d[u]$  such that  $d[u]$  is minimum &  $u \in V-S$  ( $u$  is not visited)

$u$ : index ; find a vertex from source.

$d[u]$ : entry



add  $u$  to  $S$  i.e.,  $s[u] \leftarrow 1$  visited.

for every  $v \in V-S$  do

if  $(d[u] + a[u, v] < d[v])$

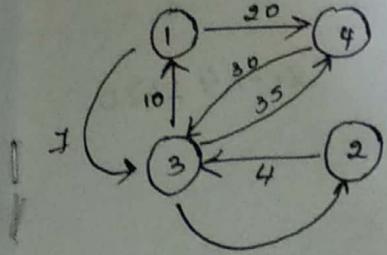
$d[v] \leftarrow d[u] + a[u, v]$

$p[v] \leftarrow u$

end if

end for.

# Apply Dijkstra's Algorithm



S: Vertices which are visited.

V-S : Vertices which are not visited.

	1	2	3	4
d	0	$\infty$	7	20
A	0	10	7	20
B	0	10	7	20
C	0	10	7	20

P	1	2	3	4
A	1	3	1	1
B	1	3	1	1
C	1	3	1	1

S	1	2	3	4
	1	1	1	1

S	V = V - S	$d[v] = \min\{d[u], d[u] + a[u, v]\}$	$p[v] = u$	$u \in V - S$
{1}	{2, 3, 4}	-	-	3, 7
{1, 3}	{2, 4}	$d[2] = \min\{\infty, 7 + 3\} = 10$ $d[4] = \min\{20, 7 + 35\} = 20$ only when $d[u] + a[u, v] = \min$ update $p[v] = u$	$p[2] = 3$ reaching 2 via 3 $p[4] = 7$	2, 20
(A)	{4}	$d[4] = \min\{20, 10 + \infty\} = 20$	-	4, 20
(B)	{1, 2, 4}	-	-	-

Final entries in the d array gives the shortest distance from source to all other vertices

d	1	2	3	4
	0	10	7	20

Source  $\rightarrow 1 = 0$       Source  $\rightarrow 2 = 10$       Source  $\rightarrow 3 = 7$       Source  $\rightarrow 4 = 20$

$$1 \rightarrow 1 = 10$$

$$1 \rightarrow 2 = 10$$

$$1 \rightarrow 3 = 7$$

$$1 \rightarrow 4 = 20$$

To find the paths use  $\phi$  array

1, 7	1, 2, 7	1, 3, 7	1, 4
P	1	3	1

To find the path from  $1 \rightarrow 2$ ,  
[first print 2, then  $P[2]$  i.e.]

first print 2 : 2

$P[2] : 3$

print  $P[3] : 1$  (stop at source?)

~~2  $\rightarrow$  3  $\rightarrow$  1 = 10~~

$1 \rightarrow 3 \rightarrow 2 = 10$

To print  $1 \rightarrow 3$ ,

print 3 : 3

$P[3] = 1$  (stop)

~~3  $\neq$  1~~

$1 \rightarrow 3 = 7$

To print  $1 \rightarrow 4$ ,

print 4 : 4  
 $P[4] = 1$  (stop)

$1 \rightarrow 4 = 20$