Single Source Shortest Path
(Dynamic Programming)

graphs, Single source shortest path is to find shortest paths to all reachable vertices from a given source.

Variants of SSSP problem

1) Single Destination shortest Path :-
   (i) Find the transpose of the graph (Reverse the edges) &
   (ii) Use SSSP algorithms.

2) Single pair shortest path :-
   This problem can be solved no faster than simply using SSSP algorithms to all vertices.

3) All pair shortest path :- Find a shortest path from u to v for every pair of vertices u and v.

** Single Source shortest path problem :-

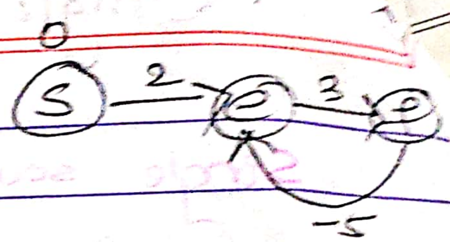
- Given a directed graph $G(V,E)$ with weighted edges $w(u,v)$, then weight $w(p)$ of path $p = \{v_0, v_1, \ldots v_k\}$ is the sum of the weight of its constituent edges.

i.e $$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

- For a given source vertex 's', the minimum weight paths to every vertex reachable from 's' is denoted by,

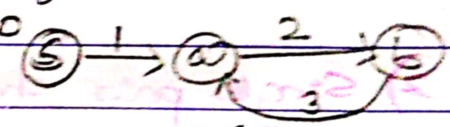
$$\delta(s,v) = \begin{cases} \min\{w(p) : s \xrightarrow{p} v\} & \text{if there is a path from } s \text{ to } v. \\ \infty & \text{otherwise.} \end{cases}$$

→ Criterias for final solutions:



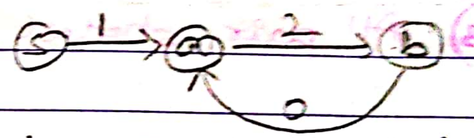
- Negative Weight Cycles:
  - Graph should not contain a negative weight cycle reachable from source.
  - Otherwise there would be no minimum path since we could simply continue to follow the negative weight cycles producing a path weight of $-\infty$)

S→a→b
S→a→b→a→b



- Positive Weight Cycles:-
  - Solution should not have any positive weight cycle because cycle could be simply removed giving a lower weight path.

S→a→b
S→a→b→a→b



- Zero weight cycles:-
  - Solution can be assumed to have no zero weight cycles because if we remove them path may change but it will not affect the minimum value.

Therefore, we can say, Solution should not contain cycles i.e they are simple paths.

So, for a graph G(V,E), solution i.e SSSP will contain atmost |V| distinct vertices and atmost |V|-1 distinct edges.

** BELLMAN - FORD ALGORITHM

is a SSSP algorithm using dynamic programming approach.

Notations:

Given a graph $G(V,E)$, for each vertex $v \in V$

(i) $v.\pi$, predecessor, is either a vertex or NIL.

(ii) $v.d$, weight of SP, an upper bound on the weight of shortest path from s to v.

(shortest path estimate)

- Bellman Ford Algorithm uses following two procedures:

[1] **Initialization**: In this, for every $v \in V$, its shortest path estimate i.e $v.d$ & predecessor i.e $v.\pi$ is initialized.

Algorithm:-

Initialize_Single_Source(G, s)	
1. for each vertex $v \in V$	$\Theta\|V\|$
2. $v.d = \infty$	$\Theta\|V\|$
3. $v.\pi$ = NIL	$\Theta\|V\|$
4. $s.d = 0$	$\Theta(1)$

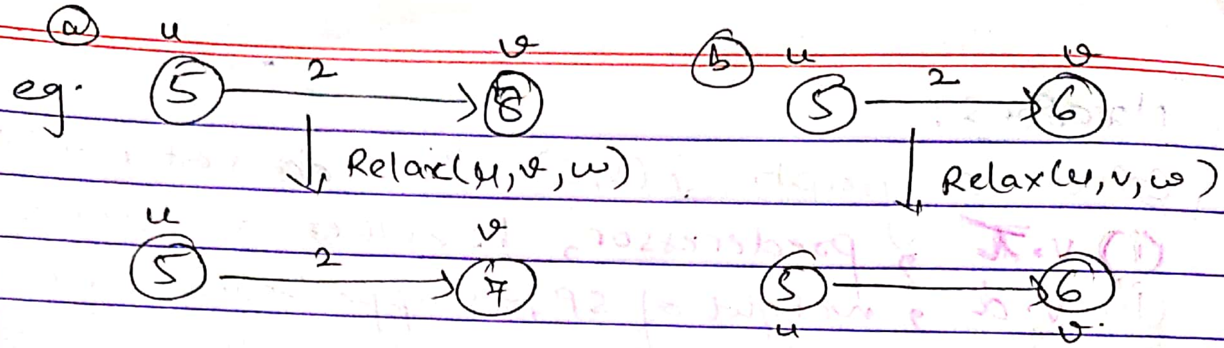
Since for loop runs for each vertex, complexity of this procedure is $\Theta(|V|)$.

[2] **Relaxation**: Process of relaxing an edge (u,v) consist of testing whether shortest path of v can be improved. If so, update $v.d$ & $v.\pi$.

Algorithm:- Relax(u, v, w)

1. if $v.d > u.d + w(u,v)$	$\Theta(1)$
2. $v.d = u.d + w(u,v)$	$\Theta(1)$
3. $v.\pi = u$	$\Theta(1)$

eg.



[3] Bellman Ford Algorithm :-

- It uses relaxation to find SSSP's on directed graphs that may contain negative weight edges.
- The algorithm will also detect if there are any negative weight cycles by returning False indicating no well defined solution.

Algorithm

Bellman_Ford (G, w, s)	
1. Initialize_Single_Source (G, s)	$\Theta(\lvert V\rvert)$
2. for i = 1 to |G.V| − 1	$O(\lvert V\rvert)$
3. for each edge (u, v) ∈ G.E	$O(\lvert V\rvert \cdot \lvert E\rvert)$
4. Relax (u, v, w)	$\Theta(1)$
// To check negative weight cycle.	
5. for each edge (u, v) ∈ G.E	$\Theta(\lvert E\rvert)$
6. if v.d > u.d + w(u, v)	$\Theta(\lvert E\rvert)$
7. return FALSE	$\Theta(1)$
8. return TRUE.	$\Theta(1)$

Complexity :

It runs in $\Theta(VE)$ time, since

(i) line 1 i.e initialization takes $\Theta(V)$

(ii) lines 2 to 4 i.e each of the |V| − 1 passes over the edges takes $O(E)$ time, in totality $O(VE)$.

(iii) lines 5 − 7 i.e for loop for checking −ve weight cycle takes $O(E)$.

4 Printing Path

The following procedure prints out the vertices on a shortest path from s to v assuming that SSSP algorithm has already computed v.π

ALGORITHM :

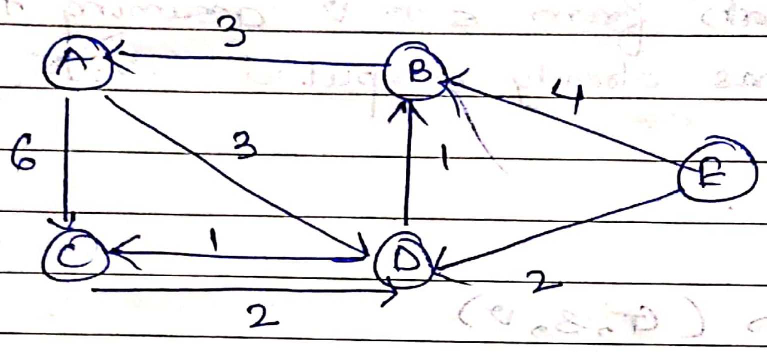
print_path (G, s, v)
1. if v == s
2. print s

3. elseif v.π == NIL
4. print "No path from 's' to 'v' exists.

5. else { print_path (G, s, v.π)
6. print v
}

Problem

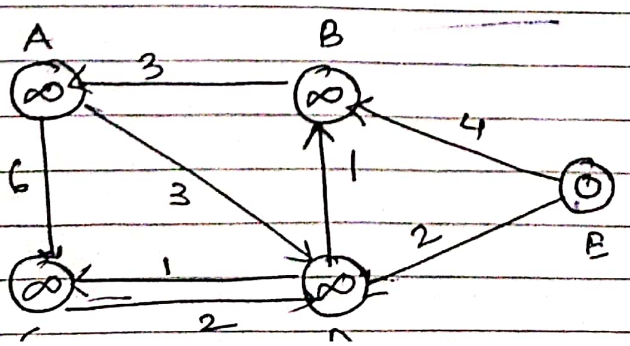
Q Run the Bellman-Ford Algorithm on the directed graph show using vertex E as source.



⇒ Edge table :

(E, B)	4
(E, D)	2
(B, A)	3
(D, B)	1
(D, C)	1
(A, D)	3
(A, C)	6
(C, D)	2

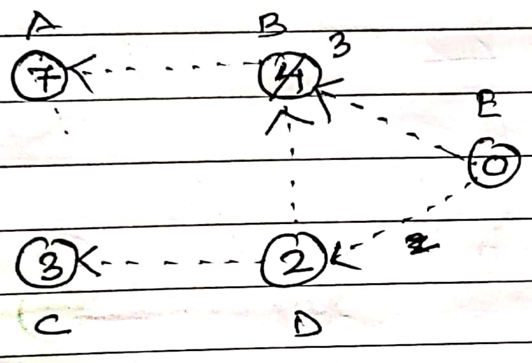
⇒ Step I : Initialization



	A	B	C	D	E
d	∞	∞	∞	∞	0
π	–	–	–	–	–

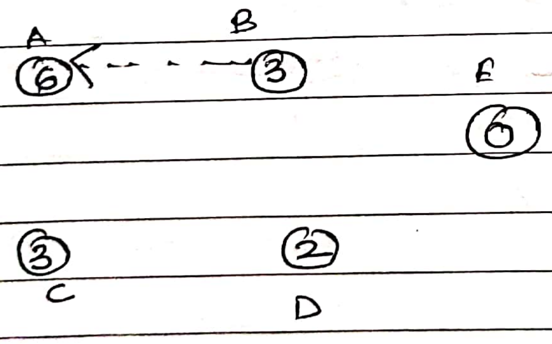
Step II Relaxation.

Iteration 1 : Relax edges (E,B), (E,D), (B,A) (D,B), (D,C)).



	A	B	C	D	E
d	7	3 ~~4~~	3	2	0
π	B	~~E~~ D	D	E	—

Iteration 2 : Relax edges. (B,A)

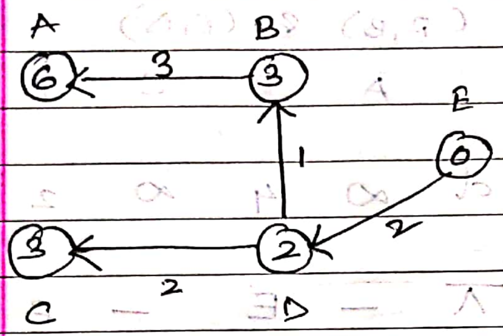


	A	B	C	D	E
d	6	3	3	2	0
π	B	D	D	E	—

Iteration 3 : No relaxation of edges

Iteration 4 : No edge Relaxation

so final shortest path from E is :-



	A	B	C	D	E
d	6	3	3	2	0
π	B	D	D	E	—

Step III Checking negative weight cycle.

1) $E \to B \Rightarrow B.d > E.d + w(E,B) \Rightarrow 3 > 0+4$ ✗

2) $E \to D \Rightarrow D.d > E.d + w(E,D) \Rightarrow 2 > 0+2$ ✗

3) $B \to A \Rightarrow A.d > B.d + w(B,A) \Rightarrow 6 > 3+3$ ✗

4) $D \to B \Rightarrow B.d > D.d + w(D,B) \Rightarrow 3 > 2+1$ ✗

5) $D \to C \Rightarrow C.d > D.d + w(D,C) \Rightarrow 3 > 2+1$ ✗

6) $A \to D \Rightarrow D.d > A.d + w(A,D) \Rightarrow 2 > 6+3$ ✗

7) $A \to C \Rightarrow C.d > A.d + w(A,C) \Rightarrow 3 > 6+6$ ✗

8) $C \to D \Rightarrow D.d > C.d + w(C,D) = 2 > 3+2$ ✗

Since, all inequalities are not true ie there is no negative weight cycle, Hence problem has SSSP solution.

IV Printing Path

1) $E \to A$

$PP(E, A) \to PP(E, B) \to PP(E, D) \to PP(E, E)$

$A. \leftarrow B \leftarrow D \leftarrow E$

$\Rightarrow E \to D \to B \to A$

2) $E \to C$

$E \to D \to C$

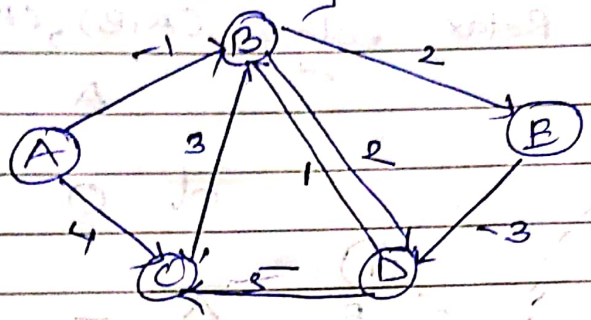
3) $E \to B$

$E \to D \to B$

4) $E \to D$

$E \to D$

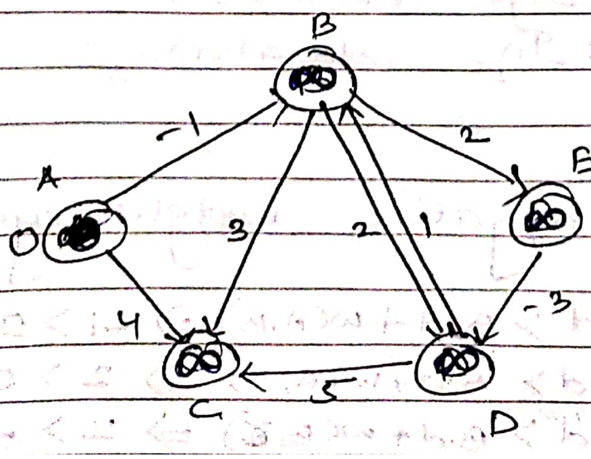
Q Run the Bellman-Ford Algorithm on the directed graph show using vertex (A) as source.



⇒ Edge Table

(A,B)	-1
(A,C)	4
(B,C)	3
(B, D)	2
(B,E)	2
(D, B)	1
(D,C)	5
(E,D)	-3

⇒ Step I : Initialization

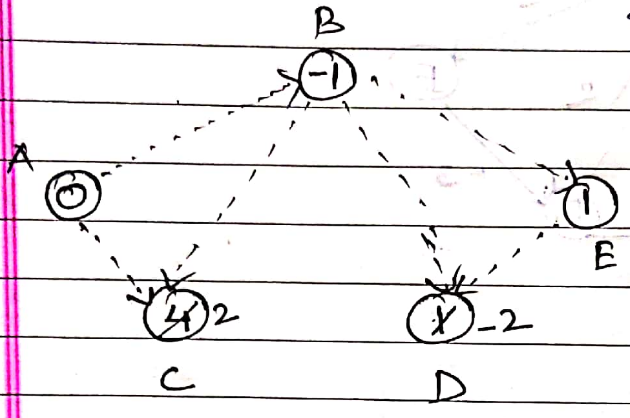


	A	B	C	D	E
d	0	∞	∞	∞	∞
π	–	–	–	–	–

Scanned by CamScanner

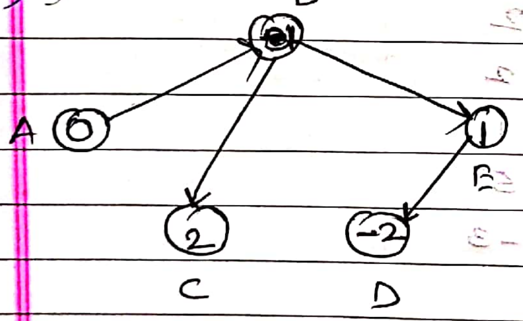
Step II Relaxation

1) Iteration 1 : Relax edges (A,B), (A,C), (B,C), (E,D)



	A	B	C	D	E
d	0	-1	~~4~~ 2	~~1~~ -2	1
π	—	A	~~A~~ B	~~B~~ E	B

2) Iteration 2 : No Edge Relaxation .



	A	B	C	D	E
d	0	-1	2	-2	1
π	—	A	B	E	B

3) Iteration 3 : No Edge relaxation .

4) Iteration 4 : No Edge relaxation

Step III : Checking negative weight cycle

1) $A \rightarrow B \Rightarrow B.d > A.d + w(A,B) \Rightarrow -1 > 0 + -1$
2) $A \rightarrow C \Rightarrow C.d > A.d + w(A,C) \Rightarrow 2 > 0 + 4$
3) $B \rightarrow C \Rightarrow C.d > B.d + w(B,C) \Rightarrow 2 > -1 + 3$
4) $B \rightarrow D \Rightarrow D.d > B.d + w(B,D) \Rightarrow -2 > -1 + 2$
5) $B \rightarrow E \Rightarrow E.d > B.d + w(B,E) \Rightarrow 1 > -1 + 2$

6) $D \to C \Rightarrow C.d > D.d + w(D, C) \Rightarrow 2 > -2 + 5$

7) $D \to B \Rightarrow B.d > D.d + w(D, B) \Rightarrow -1 > -2 + 1$

8) $E \to D \Rightarrow D.d > E.d + w(E, D) \Rightarrow -2 > 1 + -3$

Hence, no negative cycle, so there is an solution possible.

Step IV Printing path

1) $A \to B$

$A \to B$

2) $A \to C$

$A \to B \to C$

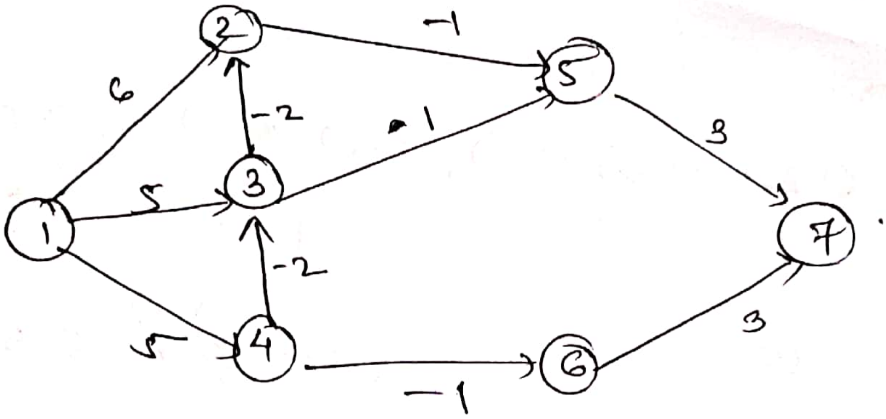
3) $A \to D$

$A \to B \to E \to D$

4) $A \to E$

$A \to B \to E$

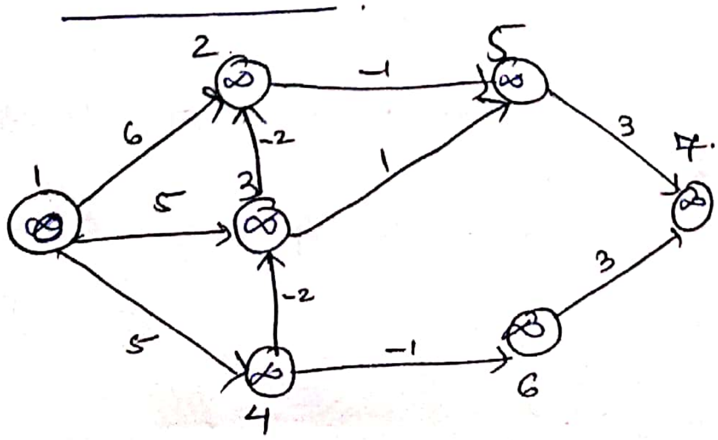
Scanned by CamScanner



⇒ Edge Table

(1,2)	6
(1,3)	5
(1,4)	5
(2,5)	-1
(3,2)	-2
(3,5)	1
(4,3)	-2
(4,6)	-1
(5,7)	3
(6,7)	3

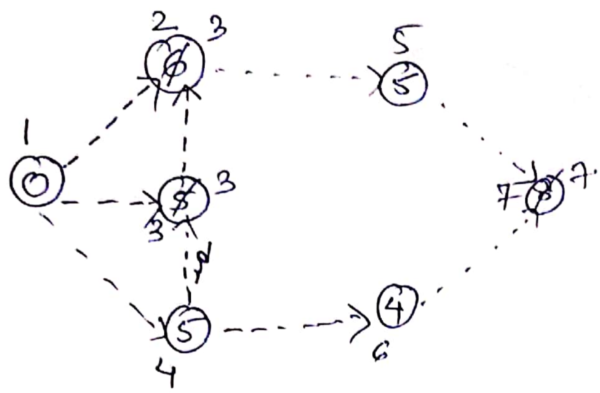
1] Initialization



	1	2	3	4	5	6	7
d	0	∞	∞	∞	∞	∞	∞
π	–	–	–	–	–	–	–

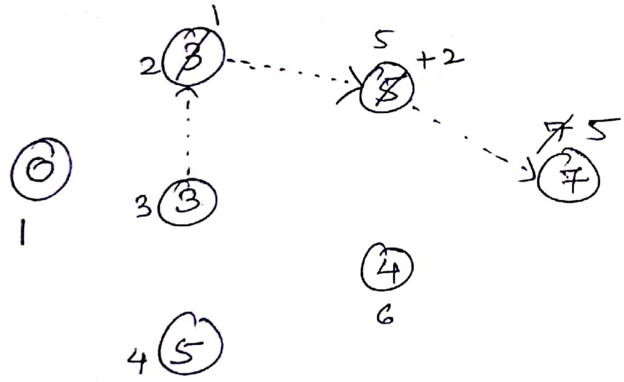
2] Relaxation

Iteration 1 : (1,2), (1,3), (1,4), (2,5) (3,2) (4,3) (5,6) (5,7)



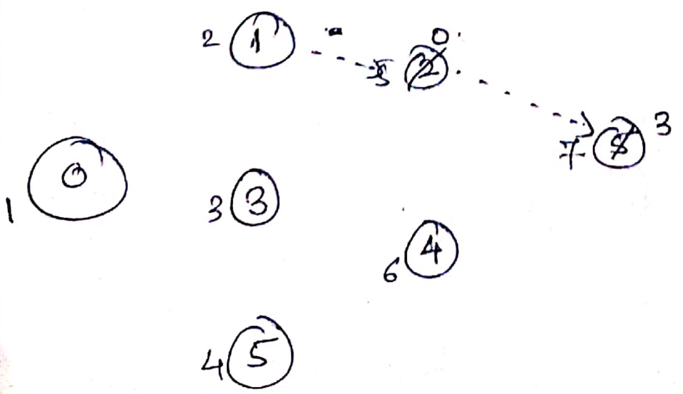
	1	2	3	4	5	6	7
d	0	~~6~~ 3	~~8~~ 3	5	5	4	~~8~~ 7
π	-	~~1~~ 3	~~1~~ 4	1	2	4	~~5~~ 6

Iteration 2 : (2,5) (3,2) (5,7)



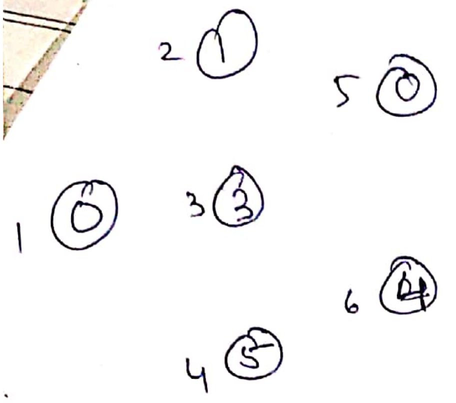
	1	2	3	4	5	6	7
d	0	1	3	5	2	4	5
π	-	3	4	1	2	4	5

Iteration 3 : (2,5) (5,7)

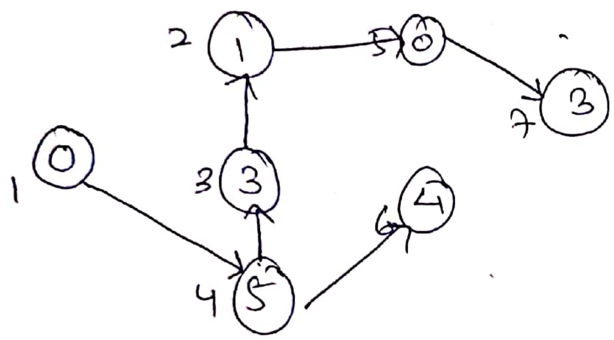


	1	2	3	4	5	6	7
d	0	1	3	5	0	4	3
π	-	3	4	1	2	4	5

ion 4: No edge relaxation.



	1	2	3	4	5	6	7
d	0	1	3	5	0	4	3
π	-	3	4	1	2	4	5



3] Checking Negative weight cycle.

1) $1 \to 2 \Rightarrow 2.d > 1.d + w(1,2) \Rightarrow 1 > 0+6$

2) $1 \to 3 \Rightarrow 3.d > 1.d + w(1,3) \Rightarrow 3 > 0+5$

3) $1 \to 4 \Rightarrow 4.d > 1.d + w(1,4) \Rightarrow 5 > 0+5$

4) $2 \to 5 \Rightarrow 5.d > 2.d + w(2,5) \Rightarrow 0 > 1-1$

5) $3 \to 2 \Rightarrow 2.d > 3.d + w(3,2) \Rightarrow 1 > 3+(-2)$

6) $3 \to 5 \Rightarrow 5.d > 3.d + w(3,5) \Rightarrow 0 > 3+1$

7) $4 \to 3 \Rightarrow 3.d > 4.d + w(4,3) \Rightarrow 3 > 5+(-2)$

8) $4 \to 6 \Rightarrow 6.d > 4.d + w(4,6) \Rightarrow 4 > 5-1$

9) $5 \to 7 \Rightarrow 7.d > 5.d + w(5,7) \Rightarrow 3 > 0+3$

10) $6 \to 7 \Rightarrow 7.d > 6.d + w(6,7) \Rightarrow 3 > 4+3$

No negative wt cycle, ∵ none of the inequalities hold true.

4] <u>printing path</u>.

1] $1 \rightarrow 2$

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

2] $1 \rightarrow 3$

$1 \rightarrow 4 \rightarrow 3$

3] $1 \rightarrow 4$

$1 \rightarrow 4$

4] $1 \rightarrow 5$

$1 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 5$

5] $1 \rightarrow 6$

$1 \rightarrow 4 \rightarrow 6$

6] $1 \rightarrow 7$

$1 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 7$