(9) BellmanFord & Dijkstras Algorithm

i) BellManFord Algorithm.

Steps:

i) Get the adjacency matrix.
ii) Get the source vertex
iii) Relax all the edges (n-1) times
where, n → no of vertices

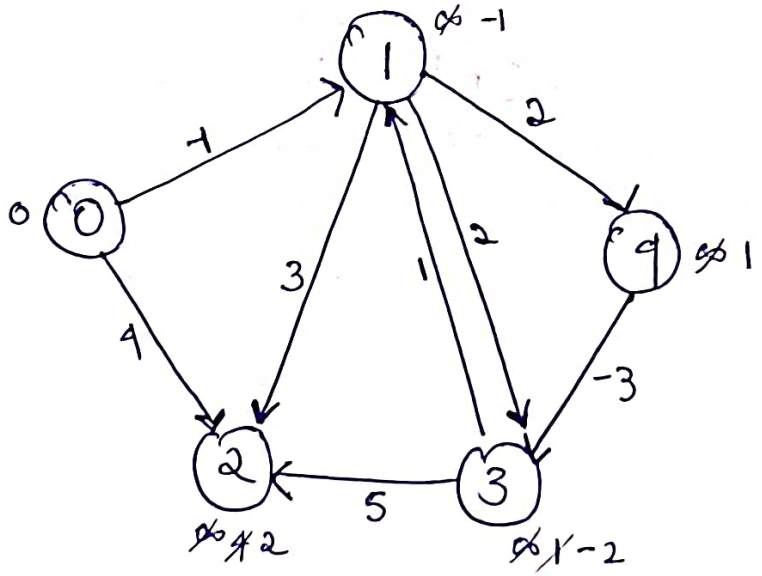
Relaxation Formula

c(u, v) → edge
if (d[u] + c(u, v) < d[v]
d[v] = d[u] + c(u, v)

iv) you will get shortest path to all the vertices.

Tracing:

Consider below Graph:



Adjacency Matrix:

Input:

	0	1	2	3	4
0	0	-1	0	0	4
1	0	0	3	2	2
2	0	0	0	0	0
3	0	1	5	0	0
4	0	0	0	-3	0

Output:

write down all the relaxed edges

(0 1) (0 2) (1 2) (1 3) (1 4) (3 1) (4 3)

Relax (n-1) times. Here 4 times ∵ n=5.

0 (A)	1 (B)	2 (C)	3 (D)	4 (E)	
0	∞	∞	∞	∞	→ I relaxation
	↓	↓	↓	↓	
0	-1	4	1	1	→ II relaxation
		↓	↓		
0	-1	2	-2	1	→ III relaxation
0	-1	2	-2	1	→ IV relaxation

∴ Output becomes:

Source = 0

Shortest Distance is

Vertex	Shortest Distance
0	0
1	-1
2	2
3	-2
4	1

Time complexity:

we relax all the edges $|V|-1$ times

$|E| \rightarrow$ no of edges

$|V| \rightarrow$ no of vertices

$\Rightarrow$ (Time complexity) $= [V-1]E$

$\approx O(VE)$

(B) Dijkstra's Algorithm :-

i) Get the adjacency Matrix & the source vertex

ii) Do relaxation.

Formula

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

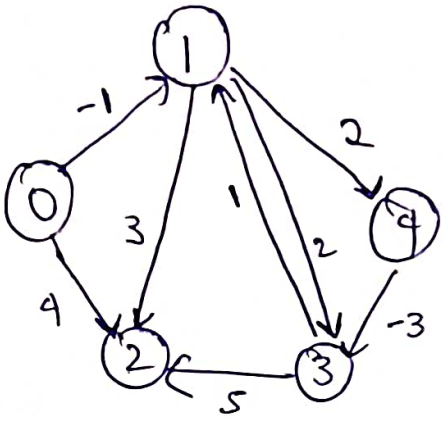
$d[v] = d[u] + c[u,v]$

- At each step do select 1 edge which has smallest cost after relaxation.

iii) Minimum cost to all the vertices will be found out.

Tracing:

Input:

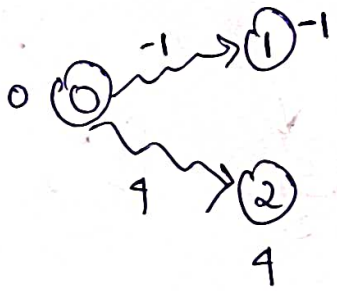


Adjacency Matrix:

	0	1	2	3	4
0	0	-1	0	0	4
1	0	0	3	2	2
2	0	0	0	0	0
3	0	1	5	0	0
4	0	0	0	~~3~~	0

| ⇒ compare.

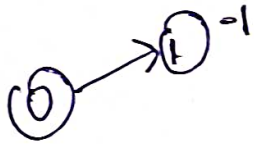
Steps:

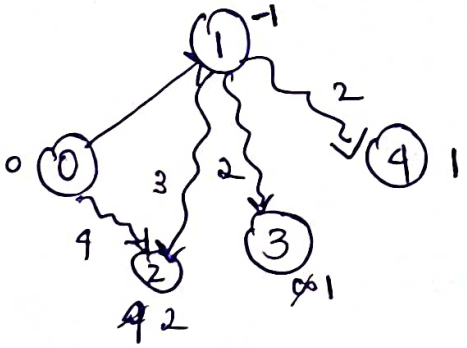


$-1 < 4$

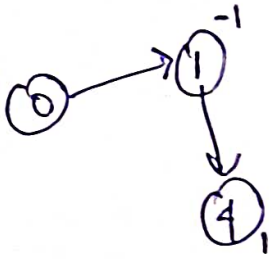
$\therefore$ select (1)

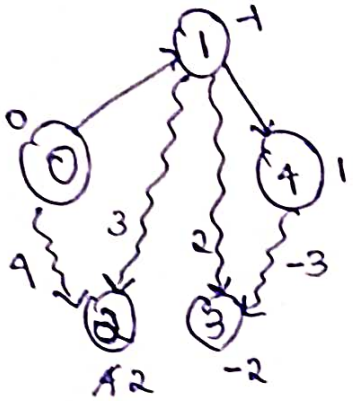
⇒



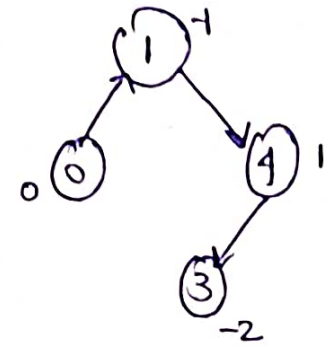


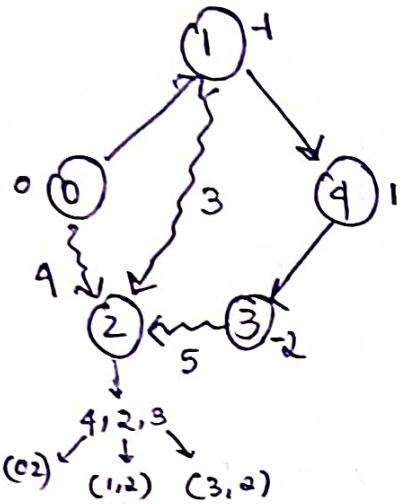
$1 \le 1 < 2$ ⇒ select (4) ⇒





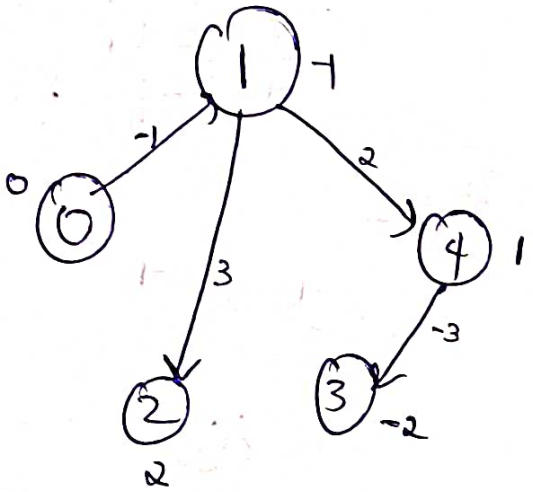
$-2 < 2 \Rightarrow$ select (3) $\Rightarrow$





$2 < 3 < 4 \Rightarrow$ select (2) i.e., (1,2)

(0,2) (3,2) (0,2)



∴ Final shortest paths:

Vertex	0	1	2	3	4
Cost	0	+1	2	-2	1

Time complexity :-

$O(V^2)$. If adjacency matrix is used.

$O(E \log V)$ If adjacency list is used. Here min heap is used to get the shortest path.

Difference b/w bellman-ford & Dijkstra

Bellman-Ford	Dijkstra
i) Single source shortest path	→ same applies.
ii) Dynamic programming approach is used	Greedy approach is used
iii) More time consuming than Dijkstra. Time complexity is $O(VE)$	Time complexity is $O(E \log V)$
iv) Does not work if negative cycle is present	May or may-not work if negative edge is present
v) Works if negative edge is present	