Part 1:

1) A Basic technique for determining the shortest Path between 2 vertices in a weighted network is dijkstra algorithm. it works on the basis of iteratively exploring the graph. from the first vertex.

Steps that Dijkstra follows:

1) Initialization; select a source node. set initial distance of the source node to 0. and other node to infinity.

2) Iterate; Reapeat until there's no unvisited n

i) select a node that has least tentative distan from the source node, and it's faces on the role of current vertex.

ii) for every unvisited adjacent vertex of current ve

1) Calculate the tentative distance from the source node to the neighbour through the current node.

2) If distance is less than previously recorded distance for the neighbour update the distance with the new shorter distance.

) Mark as visited: After updating the adjacent node mark it as visited.

Repeat: Repeat 2 to 3 until all nodes have been visited. At each step update the shortest path.

2) Under below circumstance dijkstra algorithm determines the accurate shortest path between 2 vertices.

i) No negative edge weight: Dijkstra required only positive value.

ii) Connected graph: Graph must be connected

iii) Finite graph: must have finite number of edges and vertices. Infinite path might give falsecase

iv) Dijkstra works on both Directed and undirected

v) No negative cycle (Non optimal):

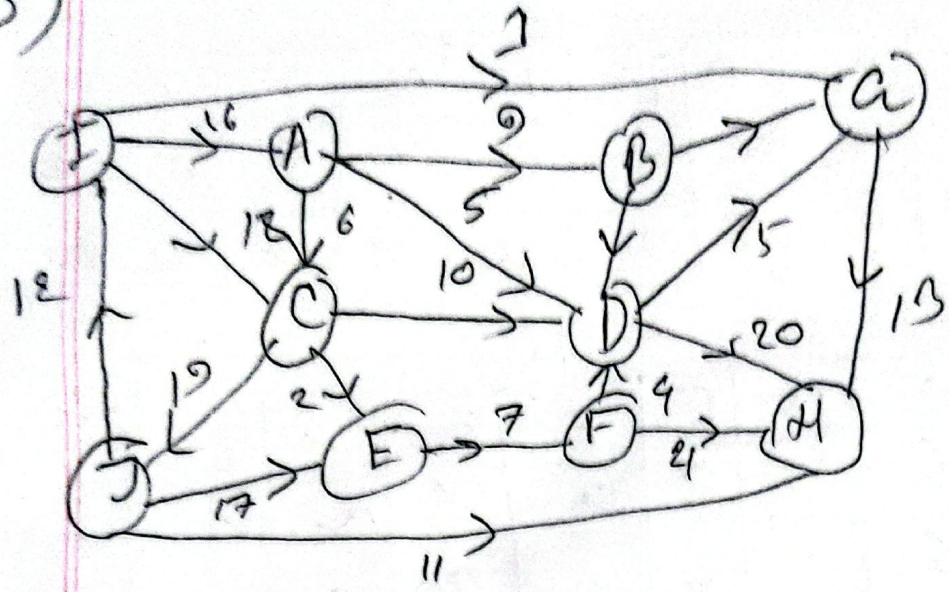
A circle is negative overall weight is said to be negative. it may trapped into. endless loop.

Q. Constraints on the types of graph where this algorithm can be applied

i) negative cycle weight

ii) negative cycle.

3)



	A	B	C	D	E	F	G	H	I	J
A	0	∞	∞	∞	∞	∞	∞	∞	∞	∞
D		9	6	5	∞		∞	∞	∞	∞
C		9	6		∞		20	25	∞	∞
E		9			8	16	20	25	∞	25
B		9				15	20	25	∞	25
F						15	20	25	∞	25
G							20	25	∞	25
H								25	∞	25
J									∞	25
I									37	

Steps:

①

	A	B	C	D	E	F	G	H	I	J
A	0	∞	∞	∞	∞	∞	∞	∞	∞	∞
		9	6	5	∞	∞	∞	∞	∞	∞

②

	A	B	C	D	E	F	G	H	I	J
D		9	6	5	∞	∞	∞	∞	∞	∞
		9	6		∞	∞	20	25	∞	∞

③

	A	B	C	D	E	F	G	H	I	J
C		9	6		~~∞~~	~~∞~~	20	25	∞	∞
		9			8	16	20	25	∞	25

④

	A	B	C	D	E	F	G	H	I	J
E		9			8	16	20	25	∞	25
		9				15	20	25	∞	25

	A	B	C	D	E	F	G	H	I	J
B		9				15	20	25	∞	25
						15	20	25	∞	25

⑥

	A	B	C	D	E	F	G	H	I	J
F						15	20	25	∞	25
							20	25	∞	25

⑦

	A	B	C	D	E	F	G	H	Z	J
G							20	25	∞	25
								25	∞	25

⑧

	A	B	C	D	E	F	G	H	I	J
H								25	∞	25
									∞	25

⑨

J
25
37

Z	J
∞	25
37	

⑩

I
37

I	J
37	

vertex	Distance from source
A	= 0
B	= 9
C	= 6
D	= 5
E	= 8
F	= 15
G	= 20
M	= 25
I	= 37
J	= 25

Part 2 :

1) Bellman - Ford is used to determine the shortest path from a single source to every other vertex. It also works on negative edge weights (as long as there's no negative cycle). His main goal is to ~~find~~ calculate quickly the shortest path between nodes.

The bellman ford algorithm is dependent on relaxation process. it improves the shortest path. by taking longer path into count. This method ensures accuracy and let the algorithm handle a large variety of graph shap. including negative edge weight.

Steps;

i) Initialization: Set Distance of source path is to 0 and all other vertices to Infinity.

ii) Relaxation Process: This step is crucial. relaxation entails going through each edge and determine whether current vertex may be used to find shorter path. If shorter path available update the distance

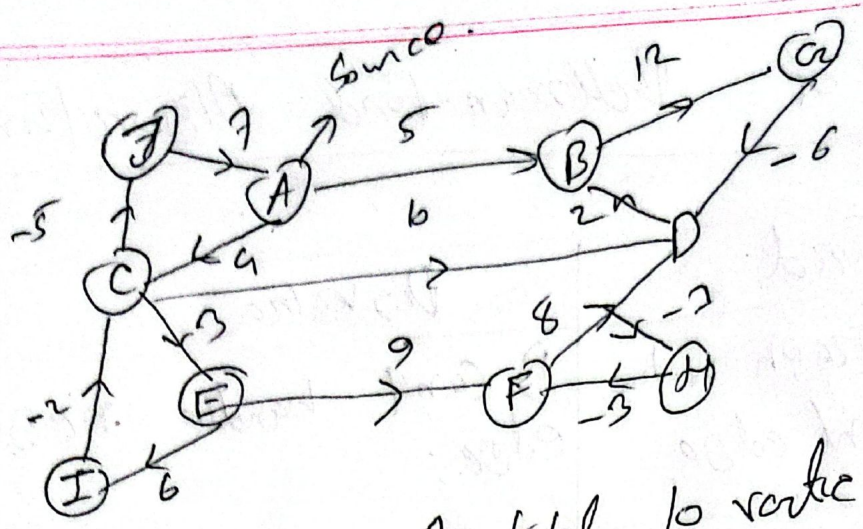
) Iteration: relaxation process repeated total V-1 times,

Find negative cycles: if there's a negative cycle means there no shortest path.

2) Advantages of Bellman-Ford Algorithm:

Bellman-Ford	Dijkstra
i) Can handle graph with negative weight edge.	i) Can't handle negative we edge.
ii) Can detect presecense of negative cycle	ii) Can't hand negative cycles.
iii) Time complexity: $O(V,E)$	iii) Time Complexity: $O((V+E)$
iv) Find shortest path, if no negative cycle.	iv) finds shortest path there's no negative we

3)



As total 10 vertices,
total Iteration = 10 - 1
= 9

Iteration : 1

	A	B	C	D	E	F	G	H	I	J
A	0	∞	∞	∞	∞	∞	∞	∞	∞	∞
B	0	5	4	∞	∞	∞	∞	∞	∞	∞
C	0	5	4	∞	∞	∞	17	∞	∞	∞
D	0	5	4	14	7	∞	17	∞	∞	∞
E	0	5	4	14	7		17	13	∞	-1
F	0	5	4	14	7	16	17	13	13	-1
G	0	5	4	14	7	16	17	13	13	-1
H	0	5	4	11	7	16	17	13	13	-1
I	0	5	4	11	7	10	17	13	13	-1
J	0	5	4	11	7	10	17	13	13	-1

Iteration : 2:

	A	B	C	D	E	F	G	H	I	J
A	0	5	4	11	7	10	17	13	13	-1
B	0	5	4	11	7	10	17	13	13	-1
C	0	5	4	11	7	10	17	13	13	-1
D	0	5	4	11	7	10	17	13	13	-1
E	0	5	4	11	7	10	17	10	13	-1
F	0	5	4	11	7	10	17	10	13	-1
G	0	5	4	11	7	10	17	10	13	-1
H	0	5	4	11	7	10	17	10	13	-1
I	0	5	4	11	7	7	17	10	13	-1
J	0	5	4	11	7	7	17	10	13	-1

Iteration 3:

	A	B	C	D	E	F	G	H	I	J
A	0	5	4	11	7	7	17	10	13	-1
B	0	5	4	11	7	7	17	10	13	-1
C	0	5	4	11	7	7	17	10	13	-1
D	0	5	4	11	7	7	17	10	13	-1
E	0	5	4	11	7	7	17	10	13	-1
F	0	5	4	11	7	7	17	10	13	-1
G	0	5	4	11	7	7	17	10	13	-1
H	0	5	4	11	7	7	17	10	13	-1
I	0	5	4	11	7	7	17	10	13	-1
J	0	5	4	11	7	7	17	10	13	-1

Vertex		Distance from src
A	=	0
B	=	5
C	=	4
D	=	11
E	=	7
F	=	7
G	=	17
H	=	10
I	=	13
J	=	-1