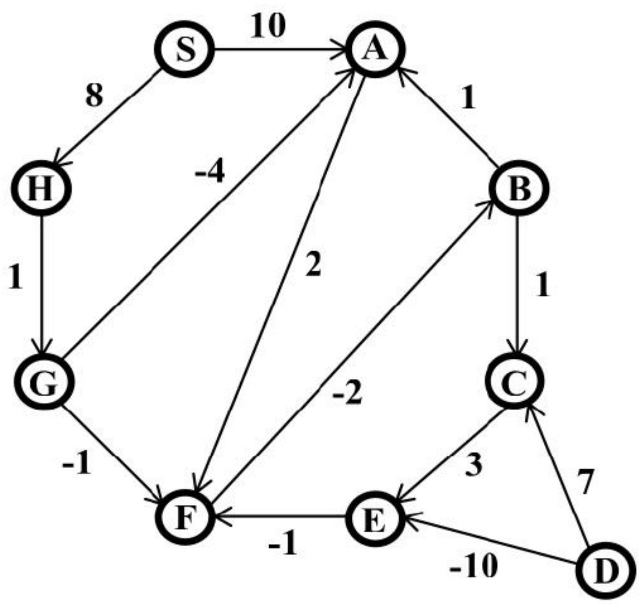
RHS

Eastern University

Course Code: 06132211/CSE 327, **Course Title:** Design and Analysis of Algorithm

Assignment-01; Fall - 2025; Full marks: 10

Observe the following graph:



Q.1	Which shortest path algorithm will be suitable for this graph and why? Find out all the shortest paths from source **'S'** to all other vertices of this graph. Complete all the iterations and show the state of the graph after each iteration. Finally, show the shortest path tree that you have found.	**[06]**	**[CLO3]**
Q.2	i. Suppose you have a graph with 100 vertices, but the shortest paths are all found after just 5 iterations. The standard Bellman-Ford algorithm continues for 94 more useless iterations. Propose a modification to the algorithm so that it can detect when the algorithm has finished early and terminate immediately. ii. Why is Breadth-First Search (BFS) insufficient for finding the shortest path in a weighted graph, even though it works for unweighted graphs? Explain with a suitable example.	**[04]**	**[CLO3]**

Ans: to the Q. No. 1

The Bellman Ford algorithm is the most suitable shrotest path algorithm for this graph.

Reason:

i) The graph contains negative edge weights (eg. −4, −2, −1, −10)

ii) Dijkstra's algorithm cannot handle negative weights.

iii) Bellman Ford works with negative edge weights, Detects negative cycles (if any)

iv) Bell man Ford algorithm finds shortest paths from a single source (s)

Hence, Bellman Ford algorithm is appropriate.

Graph nodes: S, A, B, C, D, E, F, G, H

Source node – S

Initially, $d[s] = 0$ and all others are ∞.

~~Initial state~~:

⇒ Iteration needed, $v - 1 = 9 - 1 = 8$ times.

if (dist[u] + weight[v] < dist[v])

then dist[v] = dist[u] + weight[v]

Iteration 1:

	S	A	B	C	D	E	F	G	H
S	$\boxed{0}$	∞	∞	∞	∞	∞	∞	∞	∞
A	0	$\boxed{10}$	∞	∞	∞	∞	∞	∞	8
B	0	10	$\boxed{\infty}$	∞	∞	∞	12	∞	8
C	0	10	∞	$\boxed{\infty}$	∞	∞	12	∞	8
D	0	10	∞	∞	$\boxed{\infty}$	∞	12	∞	8
E	0	10	∞	∞	∞	$\boxed{\infty}$	12	∞	8
F	0	10	∞	∞	∞	∞	$\boxed{12}$	∞	8
G	0	10	10	∞	∞	∞	12	$\boxed{\infty}$	8
H	0	10	10	∞	∞	∞	12	∞	$\boxed{8}$
	0	10	10	∞	∞	∞	12	9	8

Iteration 2 :

	S	A	B	C	D	E	F	G	H
S	$\boxed{0}$	10	10	α	α	α	12	9	8
A	0	$\boxed{10}$	10	α	α	α	12	9	8
B	0	10	$\boxed{10}$	α	α	α	12	9	8
C	0	10	10	$\boxed{11}$	α	α	12	9	8
D	0	10	10	11	$\boxed{\alpha}$	14	12	9	8
E	0	10	10	11	α	$\boxed{14}$	12	9	8
F	0	10	10	11	α	14	$\boxed{12}$	9	8
G	0	10	10	11	α	14	12	$\boxed{9}$	8
H	0	5	10	11	α	14	8	9	$\boxed{8}$
	0	5	10	11	α	14	8	9	8

Iteration 3 :

	S	A	B	C	D	E	F	G	H
S	$\boxed{0}$	5	10	11	α	14	8	9	8
A	0	$\boxed{5}$	10	11	α	14	8	9	8
B	0	5	$\boxed{10}$	11	α	14	7	9	8
C	0	5	10	$\boxed{11}$	α	14	7	9	8
D	0	5	10	11	$\boxed{\alpha}$	14	7	9	8
E	0	5	10	11	α	$\boxed{14}$	7	9	8
F	0	5	10	11	α	14	$\boxed{7}$	9	8
G	0	5	5	11	α	14	7	$\boxed{9}$	8
H	0	5	5	11	α	14	7	9	$\boxed{8}$
	0	5	5	11	α	14	7	9	8

Iteration 4 :

	S	A	B	C	D	E	F	G	H
S	$\boxed{0}$	5	5	11	α	14	7	9	8
A	0	$\boxed{5}$	5	11	α	14	7	9	8
B	0	5	$\boxed{5}$	11	α	14	7	9	8
C	0	5	5	$\boxed{6}$	α	14	7	9	8
D	0	5	5	6	$\boxed{\alpha}$	9	7	9	8
E	0	5	5	6	α	$\boxed{9}$	7	9	8
F	0	5	5	6	α	9	$\boxed{7}$	9	8
G	0	5	5	6	α	9	7	$\boxed{9}$	8
H	0	5	5	6	α	9	7	9	$\boxed{8}$
	0	5	5	6	α	9	7	9	8

Iteration 5 :

	S	A	B	C	D	E	F	G	H
S	$\boxed{0}$	5	5	6	α	9	7	9	8
A	0	$\boxed{5}$	5	6	α	9	7	9	8
B	0	5	$\boxed{5}$	6	α	9	7	9	8
C	0	5	5	$\boxed{6}$	α	9	7	9	8
D	0	5	5	6	$\boxed{\alpha}$	9	7	9	8
E	0	5	5	6	α	$\boxed{9}$	7	9	8
F	0	5	5	6	α	9	$\boxed{7}$	9	8
G	0	5	5	6	α	9	7	$\boxed{9}$	8
H	0	5	5	6	α	9	7	9	$\boxed{8}$
	0	5	5	6	α	9	7	9	8

Iteration 6-8:

No further updates occour.

Since, no distance changes, the algorithm converges.

Final shortest Distance from S:

S → S : 0

S → A : 5

S → B : 5

S → C : 11

S → D : ∝ (unreachable)

S → E : 9

S → F : 7

S → G : 9

S → H : 8

hortest Path Tree:-

S → H → G → A → F → B → C → E

Cost=9

Ans: to the Q No. 2

Early Termination in Bellman Ford Algorithm:

Problem: standard Bellman Ford runs for V-1 iterations even if no updates occour early.

Solution (modification):

i) use a flag variable
ii) If during ~~at~~ an iteration no distance value changes, stop the algorithm.

modified pseudocode: (JAVA)

```
for i = 1 to V-1:
  updated = false
  for each edge (u,v, w):
    if dist[u]+w < dist[v]:
        dist[v] = dist[u] + w
      updated = true
  if updated = = false;
     break
```

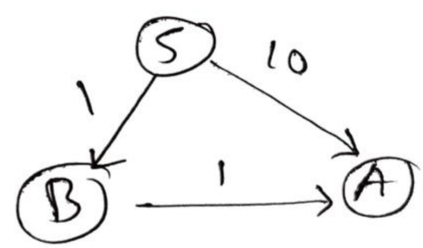
Benefit: This reduce unnecessary iterations and improves performance.

BFS Fails for weighted graph: ~~Beac~~

Reason:

1) BFS Assumes all edges have equal weight.
2) It finds the path with minimum number of edges, not minimum total weight.

Example:



⇒ BFS chooses: S → A (cost = 10)

⇒ But actual shortest path: S → B → A (cost = 2)

So, BFS ~~Givens~~ gives an incorrect shortest path in weighted graphs.