109) Bellman Ford algorithm

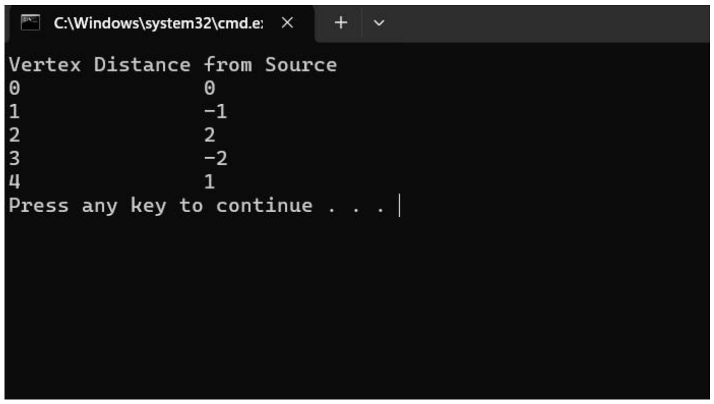
CODE:

```
class Edge:     def __init__(self, u, v,
weight):         self.u = u         self.v =
v         self.weight = weight
 def bellman_ford(edges, V, E,
source):     dist = [float('inf')] * V
dist[source] = 0
        for _ in range(V - 1):         for edge in edges:             u, v,
weight = edge.u, edge.v, edge.weight             if dist[u] != float('inf')
and dist[u] + weight < dist[v]:                 dist[v] = dist[u] + weight
        for edge in edges:         u, v, weight = edge.u, edge.v,
edge.weight         if dist[u] != float('inf') and dist[u] + weight <
dist[v]:             print("Graph contains negative weight cycle")
return

    print("Vertex Distance from Source")
for i in range(V):
print(f"{i}\t\t{dist[i]}")
 if __name__ ==
"__main__":
    V = 5
    E = 8
    edges = [
Edge(0, 1, -1),
        Edge(0, 2, 4),
        Edge(1, 2, 3),
        Edge(1, 3, 2),
        Edge(1, 4, 2),
        Edge(3, 2, 5),
        Edge(3, 1, 1),
        Edge(4, 3, -3)
    ]
    source = 0

    bellman_ford(edges, V, E, source)
```

OUTPUT:



```
Vertex Distance from Source
0               0
1               -1
2               2
3               -2
4               1
Press any key to continue . . .
```

TIME COMPLEXITY : O(V*E)