Bellman-Ford Algorithm

Implement Bellman-Ford Algorithm (50 points): The python code will be included with the assignment along with comments that outlined my thought process.

Test the Algorithm (30 points): Node 1 will be the first screen shot followed by Node 2 and Node 3

```
71  # Test the function
72  shortest_path = bellman_ford(graph, 1)
73  print(shortest_path)
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\Leetcode>
exe" "c:/Users/Ruben Ortega/OneDrive - California State University, Sacramento/Desktop/Leetco
[0, 2, 3]
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\Leetcode>
```

```
71  # Test the function
72  shortest_path = bellman_ford(graph, 2)
73  print(shortest_path)
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\
exe" "c:/Users/Ruben Ortega/OneDrive - California State University, Sacramen
[2, 0, 1]
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\
```

```
70
71  # Test the function
72  shortest_path = bellman_ford(graph, 3)
73  print(shortest_path)
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\L
exe" "c:/Users/Ruben Ortega/OneDrive - California State University, Sacramento/Deskto
[3, 1, 0]
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\L
```

Bellman-Ford Algorithm

Handle Negative Weight Cycles (10 points): My program detects when there is a negative weight cycle and returns prints out a message saying “Graph contains a negative-weight cycle “

```
54  # Graph with negative weight cycle shown in Figure 2
55  graph1 = [
56      (1, 2, -2),
57      (2, 3, -2),
58      (3, 1, 1),
59   ]
60  # (source, destination, cost)
61  bigGraph =[
62      (1,2,3),
63      (1,3,5),
64      (3,4,4),
65      (2,4,6),
66      (2,5,7),
67      (4,5,2),
68
69  ]
70
71  # Test the function
72  shortest_path = bellman_ford(graph1, 1)
73  print(shortest_path)
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\Leetcode> &
exe" "c:/Users/Ruben Ortega/OneDrive - California State University, Sacramento/Desktop/Leetcode/
Graph contains a negative-weight cycle
None
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\Leetcode>
```

Bellman-Ford Algorithm

Test your code with a bigger network (10 points): I created a graph named big graph based figure 3 with 5 nodes. I then calculated the shortest paths starting at node 1 to node 5. The screen shots will be provided in numerical order.

```
61  bigGraph =[
62      (1,2,3),
63      (1,3,5),
64      (3,4,4),
65      (2,4,6),
66      (2,5,7),
67      (4,5,2),
68
69  ]
70
71  # Test the function
72  shortest_path = bellman_ford(bigGraph, 1)
73  print(shortest_path)
```

```
PROBLEMS 7  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\Leetcode>
exe" "c:/Users/Ruben Ortega/OneDrive - California State University, Sacramento/Desktop/Leetcod
[0, 3, 5, 9, 10]
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\Leetcode>
```

```
60  # (source, destination, cost)
61  bigGraph =[
62      (1,2,3),
63      (1,3,5),
64      (3,4,4),
65      (2,4,6),
66      (2,5,7),
67      (4,5,2),
68
69  ]
70
71  # Test the function
72  shortest_path = bellman_ford(bigGraph, 2)
73  print(shortest_path)
```

```
PROBLEMS 7  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\Leetcode>
exe" "c:/Users/Ruben Ortega/OneDrive - California State University, Sacramento/Desktop/Leetcod
[3, 0, 8, 6, 7]
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\Leetcode>
```

```
60  # (source, destination, cost)
61  bigGraph =[
62      (1,2,3),
63      (1,3,5),
64      (3,4,4),
65      (2,4,6),
66      (2,5,7),
67      (4,5,2),
68
69  ]
70
71  # Test the function
72  shortest_path = bellman_ford(bigGraph, 3)
73  print(shortest_path)
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\Leetcode>
exe" "c:/Users/Ruben Ortega/OneDrive - California State University, Sacramento/Desktop/Leetcod
[5, 8, 0, 4, 6]
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\Leetcode>
```

```
60  # (source, destination, cost)
61  bigGraph =[
62      (1,2,3),
63      (1,3,5),
64      (3,4,4),
65      (2,4,6),
66      (2,5,7),
67      (4,5,2),
68
69  ]
70
71  # Test the function
72  shortest_path = bellman_ford(bigGraph, 4)
73  print(shortest_path)
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\Leetcode>
exe" "c:/Users/Ruben Ortega/OneDrive - California State University, Sacramento/Desktop/Leetcod
[9, 6, 4, 0, 2]
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\Leetcode>
```

```
]
# (source, destination, cost)
bigGraph =[
    (1,2,3),
    (1,3,5),
    (3,4,4),
    (2,4,6),
    (2,5,7),
    (4,5,2),

]

# Test the function
shortest_path = bellman_ford(bigGraph, 5)
print(shortest_path)
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\Leetcode>
exe" "c:/Users/Ruben Ortega/OneDrive - California State University, Sacramento/Desktop/Leetcod
[10, 7, 6, 2, 0]
PS C:\Users\Ruben Ortega\OneDrive - California State University, Sacramento\Desktop\Leetcode>
```