**Bellman-Ford BF Algorithm**

**Iheb Gafsi**\*

INSAT Student

**Iheb.engineer@gmail.com**

**Definition:**

The Bellman-Ford algorithm iteratively relaxes the edges, minimizing the distance to each vertex, and continues this process for V-1 iterations, where V is the number of vertices in the graph. During each iteration, the algorithm examines all edges and updates the distances to the vertices if a shorter path is found. This process is repeated until no further updates are possible, ensuring that the shortest distances have been calculated. Bellman-Ford guarantees the correct shortest paths. The time complexity of Bellman-Ford is , where V is the number of vertices and E is the number of edges in the graph.

**Use cases:**

This Algorithm might not be the fastest SSSP algorithm but it comes in handy for determining negative cycles and where they occur.

Detecting negative cycles is an essential task in various applications, including financial trading, where it helps identify opportunities for arbitrage. In network systems, finding negative cycles is crucial to prevent the occurrence of loops that could lead to instability or undesired feedback effects. Additionally, in graph algorithms, identifying negative cycles is fundamental for solving problems such as finding the shortest path with negative edge weights, where these cycles may affect the optimal route.

**Algorithm:**

1. # Variables

2. graph = adjacency list

3.

4. #Bellman-Ford Algorithm

5. def bf(graph, start):

6. n = len(graph)

7. distances = [float('inf')] \* n

8. distances[start] = 0

9. for i in range(n - 1):

10. for arg in range(n):

11. for to, weight in graph[arg]:

12. distances[ to ] = min(distances[ to ] , distances[arg] + weight)

13. for i in range(n - 1):

14. for arg in range(n):

15. for to, weight in graph[arg]:

16. if distances[ to ] > distances[arg] + weight:

17. distances[ to ] = -float('inf')

18.

19. return distances

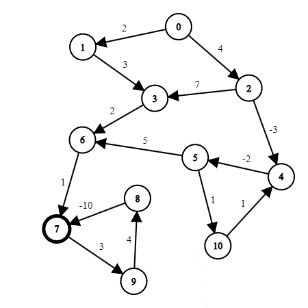
20.

21. print( bf(graph, 0) )

22.

**Example:**

Here’s a small example illustrating an example of input outputs for the Bellman-Ford:



We will use the Python code down below to outline the output of the algorithm on this graph:

1. # Variables

2. graph = [

3.     [(1, 2), (2, 4)],

4.     [(3, 3)],

5.     [(3, 7), (4, -3)],

6.     [(6, 2)],

7.     [(5, -2)],

8.     [(6, 5), (10, 1)],

9.     [(7, 1)],

10.     [(9, 3)],

11.     [(7, -10)],

12.     [(8, 4)],

13.     [(4, 1)]

14. ]

15.

16. #Bellman-Ford Algorithm

17. def bf(graph, start):

18.     n = len(graph)

19.     distances = [float('inf')] \* n

20.     distances[start] = 0

21.

22.     for i in range(n - 1):

23.         for arg in range(n):

24.             for to, weight in graph[arg]:

25.                 distances[ to ] = min(distances[ to ] , distances[arg] + weight)

26.

27.     for i in range(n - 1):

28.         for arg in range(n):

29.             for to, weight in graph[arg]:

30.                 if distances[ to ] > distances[arg] + weight:

31.                     distances[ to ] = -float('inf')

32.

33.     return distances

34.

35. print( bf(graph, 0) )

36.

The corresponding output is:

1. Python >> [0, 2, 4, 5, 1, -1, 4, -inf, -inf, -inf, 0]

