Aim: Bellman-Ford algorithm is to find the shortest path between a source vertex and all other vertices in a weighted directed graph, where the weights can be negative. The algorithm achieves this by iteratively relaxing the edges in the graph and updating the distances of the vertices until the shortest path is found.

Theory:

The Bellman-Ford algorithm is based on the principle of dynamic programming, which involves breaking a problem down into smaller subproblems and solving them in a bottom-up manner.

The algorithm maintains an array of distances d[] for each vertex in the graph, where d[s] is the distance from the source vertex s to itself, and d[v] is the distance from the source vertex s to the vertex v.

The algorithm then iterates over all the edges in the graph for |V|-1 times, where |V| is the number of vertices in the graph. In each iteration, the algorithm relaxes all the edges in the graph and updates the distances of the vertices if a shorter path is found.

The relaxation of an edge (u,v) involves comparing the distance of the source vertex s to u plus the weight of the edge (u,v) with the current distance of the source vertex s to v. If the former is smaller than the latter, the distance of v is updated to the former.

The algorithm terminates when all the vertices have been relaxed for |V|-1 times or when no more updates can be made to the distances of the vertices. If the algorithm terminates after |V|-1 iterations, the shortest path has been found. If the algorithm terminates with an update in the |V|-th iteration, there exists a negative cycle in the graph.

The time complexity of the Bellman-Ford algorithm is O(|V||E|), where |V| is the number of vertices in the graph, and |E| is the number of edges in the graph. This makes it slower than other shortest path algorithms such as Dijkstra's algorithm, but it is more flexible as it can handle negative edge weights.

Algorithm:

Bellmanford(G,w,s)

```
    Initialise_Single_Source(G,w,s)
    For i=1 to |G.V|-1
        For each edge(u,v) of G.E
            Relax(u,v,w)
        For each edge(u,v) of G.E
            If v.d > u.d + w(u,v)
                Return false
            Else
                Return true
Initialize_single_source(G,s)
    For each vertex v of G.V
        v.d = ∞
v. π = NIL
Relax(u,v,w)
    If v.d > u.d + w(u,v)
        v.d = u.d + w(u,v)
        v. π = u
```

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#define MAX_NODES 1000
#define MAX_EDGES 10000
int dist[MAX_NODES];
int edges[MAX_EDGES][3];

void BellmanFord(int nodes, int edges_count, int start_node) {
```

```
    for (int i = 0; i < nodes; i++) {
        if (i == start_node) {
            dist[i] = 0;
        } else {
            dist[i] = INT_MAX;
        }
    }
    for (int i = 1; i < nodes; i++) {
        for (int j = 0; j < edges_count; j++) {
            int source = edges[j][0];
            int destination = edges[j][1];
            int weight = edges[j][2];

            if (dist[source] != INT_MAX && dist[source] + weight < dist[destination]) {
                dist[destination] = dist[source] + weight;
            }
        }
    }
    for (int i = 0; i < edges_count; i++) {
        int source = edges[i][0];
        int destination = edges[i][1];
        int weight = edges[i][2];

        if (dist[source] != INT_MAX && dist[source] + weight < dist[destination]) {
            printf("Graph contains negative-weight cycle\n");
            return;
        }
    }
    for (int i = 0; i < nodes; i++) {
```

```
        printf("Shortest distance from node %d to %d is %d\n", start_node, i, dist[i]);
    }
}

int main() {
    int nodes, edges_count, start_node;
    printf("Enter number of nodes, edges and starting node: ");
    scanf("%d %d %d", &nodes, &edges_count, &start_node);
    printf("Enter the edges in the format <source> <destination> <weight>:\n");
    for (int i = 0; i < edges_count; i++) {
        scanf("%d %d %d", &edges[i][0], &edges[i][1], &edges[i][2]);
    }

    BellmanFord(nodes, edges_count, start_node);
    return 0;
}
```

Output:

```
Enter number of nodes, edges and starting node: 7 10 0
Enter the edges in the format <source> <destination> <weight>:
0 1 6
0 2 5
0 3 5
2 1 -2
3 2 -2
1 4 -1
2 4 1
3 5 -1
5 6 3
4 6 3
Shortest distance from node 0 to 0 is 0
Shortest distance from node 0 to 1 is 1
Shortest distance from node 0 to 2 is 3
Shortest distance from node 0 to 3 is 5
Shortest distance from node 0 to 4 is 0
Shortest distance from node 0 to 5 is 4
Shortest distance from node 0 to 6 is 3
```

Conclusion:

In conclusion, the Bellman-Ford algorithm is a useful algorithm for finding shortest paths in graphs with negative edge weights, and its ability to detect negative-weight cycles makes it a valuable tool in many applications. However, its time complexity of O(|V||E|) makes it less efficient than Dijkstra's algorithm for graphs with non-negative edge weights.