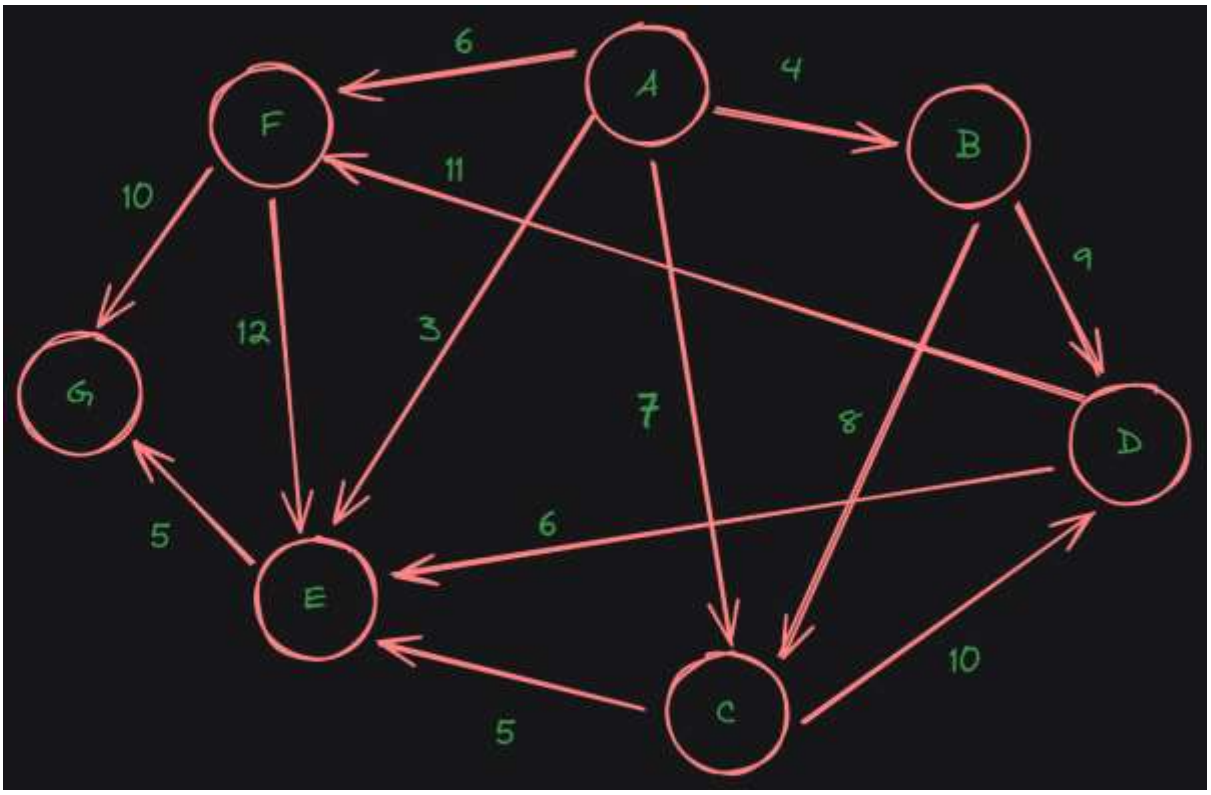

Dijkstra's Algorithm:

```
DijkstraAlgorithm(graph, source):
  distances[source] = 0
  priorityQueue = createPriorityQueue()

  for each vertex in graph:
    if vertex != source:
      distances[vertex] = infinity
    priorityQueue.add(vertex)

  while priorityQueue is not empty:
    current = priorityQueue.extractMin()

    for each neighbor of current:
      tempDistance = distances[current] + distance between current and neighbor
      if tempDistance < distances[neighbor]:
        distances[neighbor] = tempDistance

  return distances
```

```
output:
Distances from A:
A: 0
B: 4
C: 7
D: 13
E: 3
F: 6
G: 8
```

Bellman Ford:

```
BellmanFordAlgorithm(graph, source):
  distances[source] = 0

  for i from 1 to |V| - 1:
    for each edge in graph:
      if distances[edge.start] + edge.weight < distances[edge.end]:
        distances[edge.end] = distances[edge.start] + edge.weight

  for each edge in graph:
    if distances[edge.start] + edge.weight < distances[edge.end]:
      // Negative weight cycle detected
      return "Graph contains negative weight cycle"

  return distances
```

```
Output:
Distances from A:
A: 0
B: 4
C: 7
D: 13
E: 3
F: 6
G: 8
```