# Dijkstra Algorithm Java

**Dijkstra algorithm is** one of the prominent algorithms to find the shortest path from the source node to a destination node. It uses the greedy approach to find the shortest path. The concept of the Dijkstra algorithm is to find the shortest distance (path) starting from the source point and to ignore the longer distances while doing an update.

In this section, we will implement the **Dijkstra algorithm in Java program**. Also, we will discuss its usage and limitations.

## Dijkstra Algorithm Steps

**Step1:** All nodes should be marked as unvisited.

**Step2:** All the nodes must be initialized with the "infinite" (a big number) distance. The starting node must be initialized with zero.

**Step3:** Mark starting node as the current node.

**Step4:** From the current node, analyze all of its neighbors that are not visited yet, and compute their distances by adding the weight of the edge, which establishes the connection between the current node and neighbor node to the current distance of the current node.

**Step5:** Now, compare the recently computed distance with the distance allotted to the neighboring node, and treat it as the current distance of the neighboring node,

**Step6:** After that, the surrounding neighbors of the current node, which has not been visited, are considered, and the current nodes are marked as visited.

**Step7:** When the ending node is marked as visited, then the algorithm has done its job; otherwise,

**Step8:** Pick the unvisited node which has been allotted the minimum distance and treat it as the new current node. After that, start again from step4.

## Dijkstra Algorithm Pseudo Code

1. Method Dijkstra(G, s): // G is graph, s is source
2. distance[s] -> 0 // Distance from the source to source is always 0
3. **for** every vertex vx in the Graph G: // doing the initialization work
4. {
5. **if** vx ? s
6. {
7. // Unknown distance function from source to each node set to infinity
8. distance[vx] -> infinity
9. }
10. add vx to Queue Q // Initially, all the nodes are in Q
11. }
13. // The while loop
14. Untill the Q is not empty:
15. {
16. // During the first run, this vertex is the source or starting node
17. vx = vertex in Q with the minimum distance[vx]
18. delete vx from Q
19. }
20. // where the neighbor ux has not been deleted yet from Q.
21. **for** each neighbor ux of vx:
22. alt = distance[vx] + length(vx, ux)
23. // A path with lesser weight (shorter path), to ux is found
24. **if** alt < distance[ux]:
25. distance[ux] = alt // updating the distance of ux
27. **return** dist[]
28. end Method

## Implementation of Dijkstra Algorithm

The following code implements the Dijkstra Algorithm using the diagram mentioned below.

