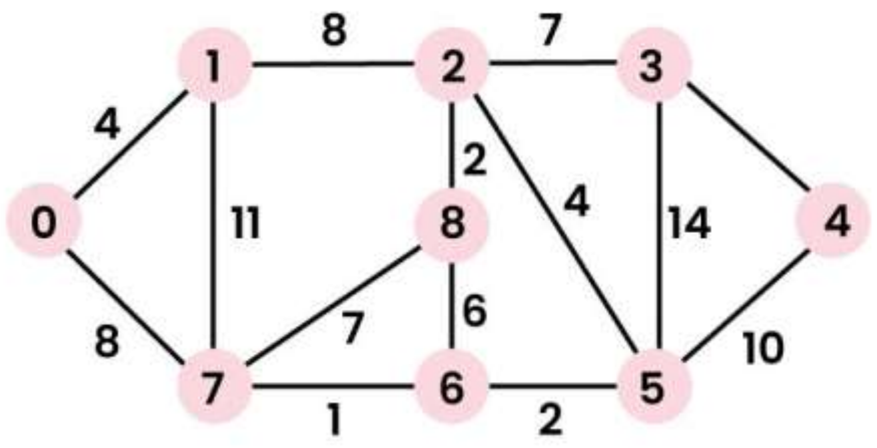
Dijkstra’s Algorithm

Given a weighted graph and a source vertex in the graph, find the **shortest paths** from the source to all the other vertices in the given graph.
Note: The given graph does not contain any negative edge.
Examples:
Input: *src = 0, the graph is shown below.*



Working of Dijkstra's Algorithm

Output: *0 4 12 19 21 11 9 8 14*
Explanation: *The distance from 0 to 1 = 4.*
The minimum distance from 0 to 2 = 12. 0->1->2
The minimum distance from 0 to 3 = 19. 0->1->2->3
The minimum distance from 0 to 4 = 21. 0->7->6->5->4
The minimum distance from 0 to 5 = 11. 0->7->6->5
The minimum distance from 0 to 6 = 9. 0->7->6
The minimum distance from 0 to 7 = 8. 0->7
The minimum distance from 0 to 8 = 14. 0->1->2->8

Dijkstra’s Algorithm using Adjacency Matrix:

*The idea is to generate a **SPT (shortest path tree)** with a given source as a root. Maintain an Adjacency Matrix with two sets,*

- *one set contains vertices included in the shortest-path tree,*
- *other set includes vertices not yet included in the shortest-path tree.*

At every step of the algorithm, find a vertex that is in the other set (set not yet included) and has a minimum distance from the source.

Algorithm:

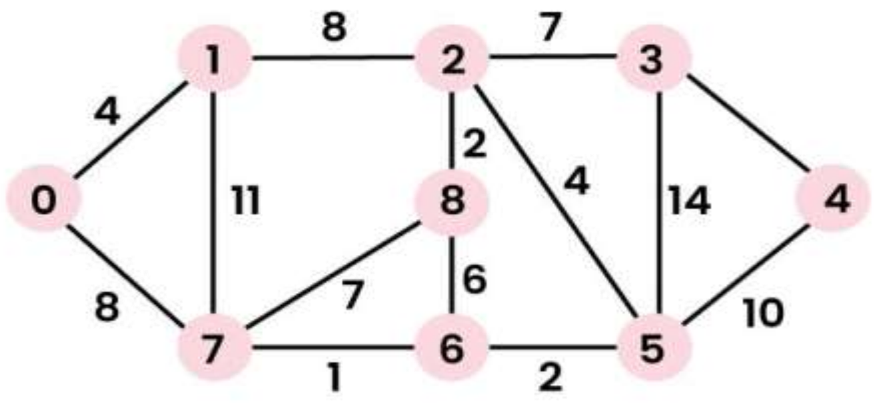
- Create a set **sptSet** (shortest path tree set) that keeps track of vertices included in the shortest path tree, i.e., whose minimum distance from the source is calculated and finalized. Initially, this set is empty.
- Assign a distance value to all vertices in the input graph. Initialize all distance values as **INFINITE**. Assign the distance value as 0 for the source vertex so that it is picked first.
- While **sptSet** doesn't include all vertices
    - Pick a vertex **u** that is not there in **sptSet** and has a minimum distance value.
    - Include u to **sptSet**.
    - Then update the distance value of all adjacent vertices of **u**.
        - To update the distance values, iterate through all adjacent vertices.
        - For every adjacent vertex **v,** if the sum of the distance value of **u** (from source) and weight of edge **u-v**, is less than the distance value of **v**, then update the distance value of **v**.

Note: We use a boolean array **sptSet[]** to represent the set of vertices included in **SPT**. If a value **sptSet[v]** is true, then vertex v is included in **SPT**, otherwise not. Array **dist[]** is used to store the shortest distance values of all vertices.

Illustration of Dijkstra Algorithm:

To understand the Dijkstra's Algorithm lets take a graph and find the shortest path from source to all nodes.

Consider below graph and ***src = 0***

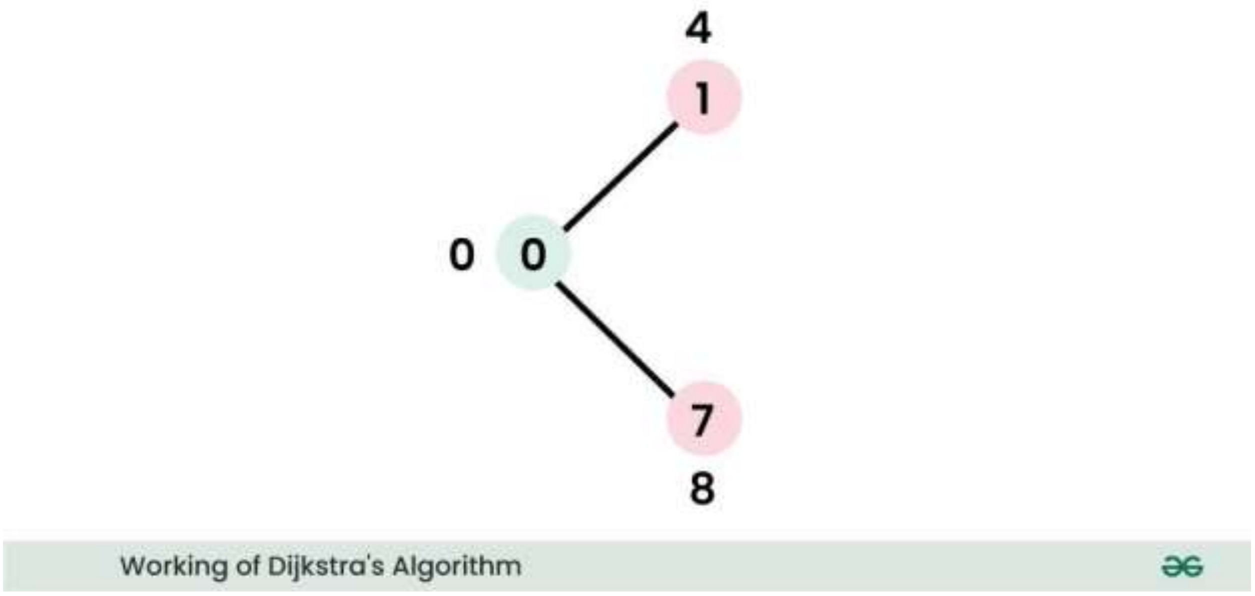


Working of Dijkstra's Algorithm

Step 1:

- *The set **sptSet** is initially empty and distances assigned to vertices are {0, INF, INF, INF, INF, INF, INF, INF} where **INF** indicates infinite.*
- *Now pick the vertex with a minimum distance value. The vertex 0 is picked, include it in **sptSet**. So **sptSet** becomes {0}. After including 0 to **sptSet**, update distance values of its adjacent vertices.*
- *Adjacent vertices of 0 are 1 and 7. The distance values of 1 and 7 are updated as 4 and 8.*

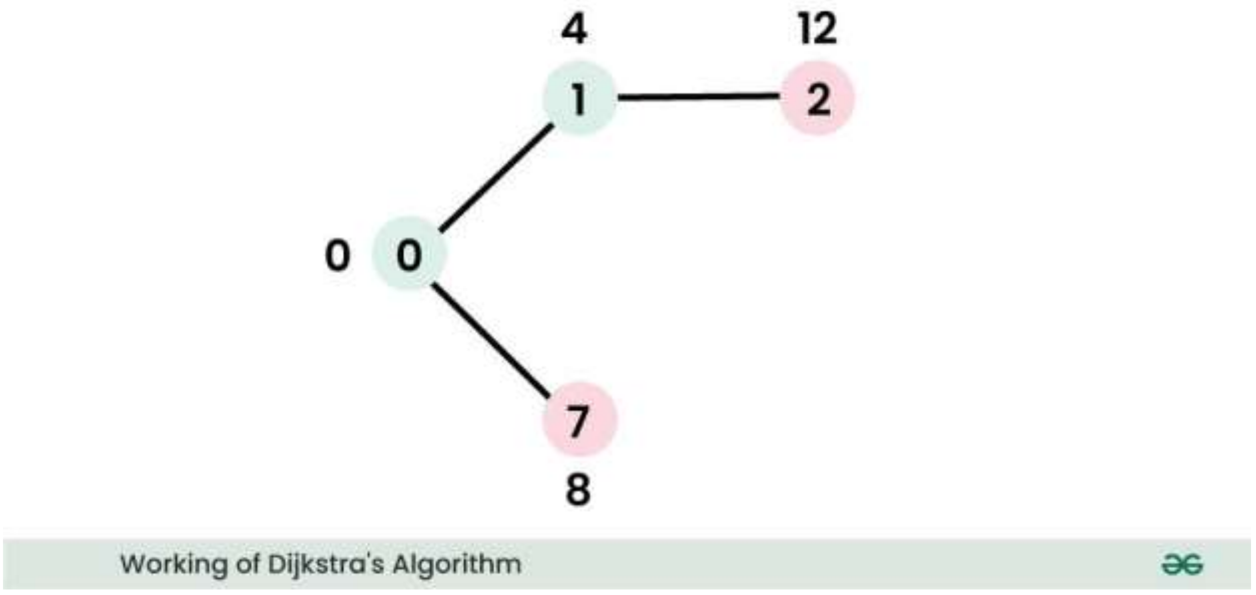
*The following subgraph shows vertices and their distance values, only the vertices with finite distance values are shown. The vertices included in **SPT** are shown in **green** colour.*



Step 2:

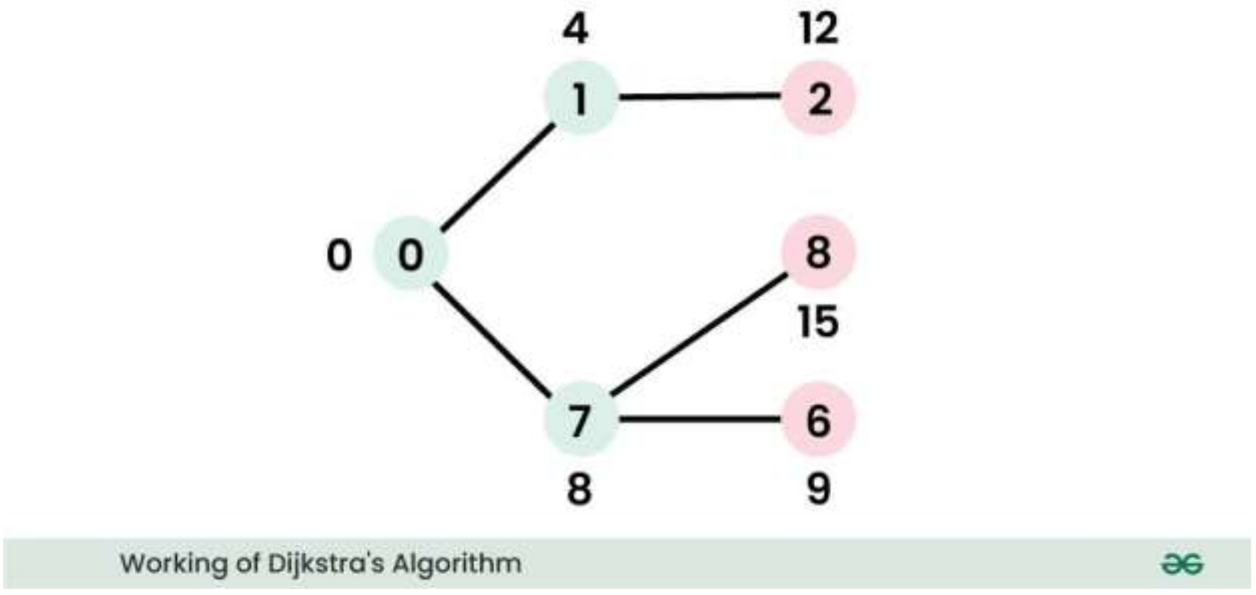
- *Pick the vertex with minimum distance value and not already included in **SPT** (not in **sptSET**). The vertex 1 is picked and added to **sptSet**.*
- *So **sptSet** now becomes {0, 1}. Update the distance values of adjacent vertices of 1.*

- *The distance value of vertex 2 becomes* ***12****.*



Step 3:

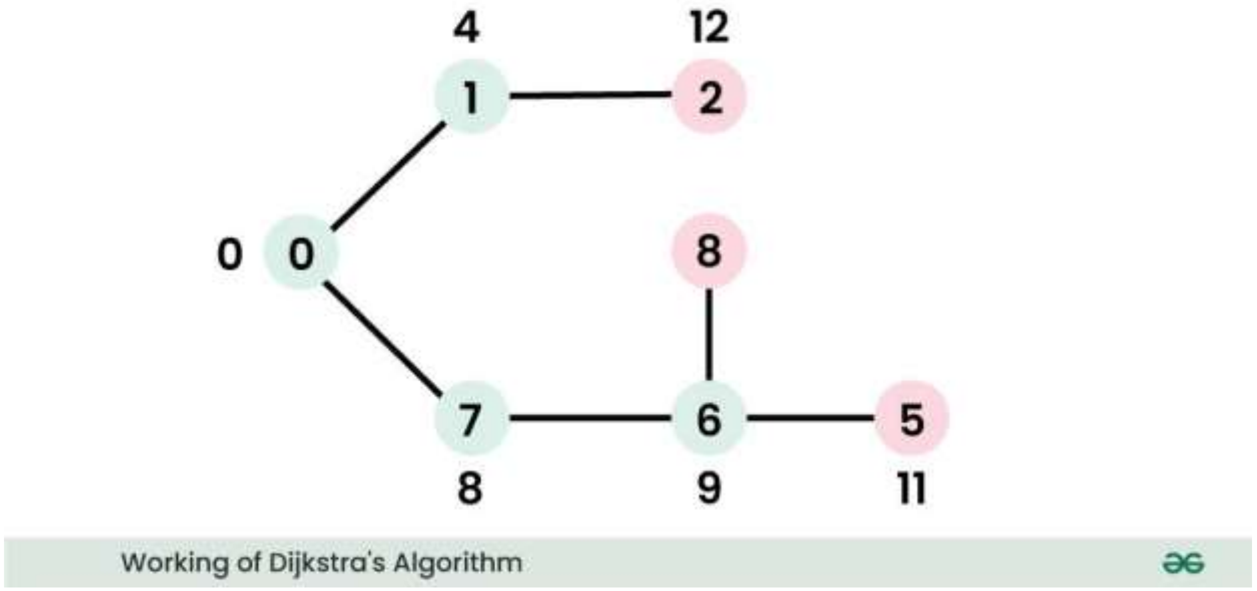
- *Pick the vertex with minimum distance value and not already included in* ***SPT*** *(not in* ***sptSET****). Vertex 7 is picked. So* ***sptSet*** *now becomes* ***{0, 1, 7}.***
- *Update the distance values of adjacent vertices of 7. The distance value of vertex 6 and 8 becomes finite (****15 and 9*** *respectively).*



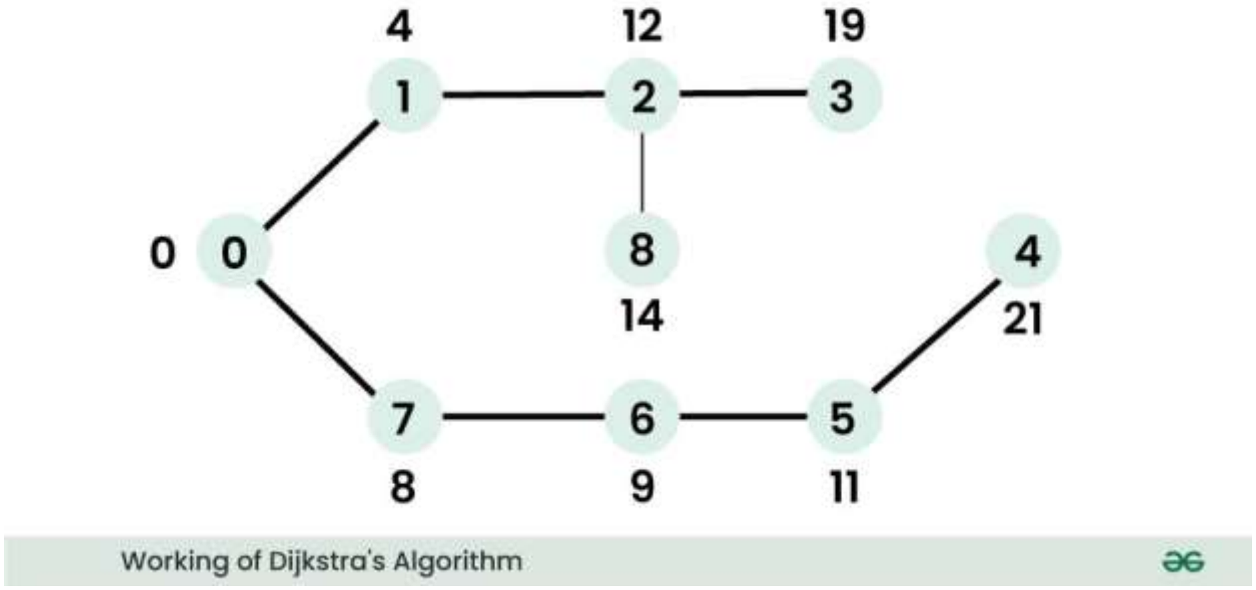
Step 4:

- *Pick the vertex with minimum distance value and not already included in* ***SPT*** *(not in* ***sptSET****). Vertex 6 is picked. So* ***sptSet*** *now becomes* ***{0, 1, 7, 6}****.*

- *Update the distance values of adjacent vertices of 6. The distance value of vertex 5 and 8 are updated.*



We repeat the above steps until ***sptSet*** *includes all vertices of the given graph. Finally, we get the following S****hortest Path Tree (SPT).***


4
12
19
1
2
3
0
0
8
4
14
21
7
6
5
8
9
11
Working of Dijkstra's Algorithm