Shortest path in Directed Acyclic Graph

Given a Directed Acyclic Graph of N vertices from 0 to N-1 and a 2D Integer array(or vector) edges[ ][ ] of length M, where there is a directed edge from edge[i][0] to edge[i][1] with a distance of edge[i][2] for all i, 0<=i

Find the **shortest** path from **src(0)** vertex to all the vertices and if it is impossible to reach any vertex, then return **-1** for that vertex.

Example:

```
Input:
n = 6, m= 7
edge=[[0,1,2],[0,4,1],[4,5,4]
,[4,2,2],[1,2,3],[2,3,6],[5,3,1]]

Output:
0 2 3 6 1 5
```

Your Task:

You don't need to print or input anything. Complete the function **shortest path()** which takes an integer N as number of vertices, an integer M as number of edges and a 2D Integer array(or vector) edges as the input parameters and returns an **integer array(or vector)**, denoting the list of distance from src to all nodes.

Constraint:

- 1 <= n,m <= 100

- $0 <= edge_{i,0}, edge_{i,1} < n$

Expected Time Complexity: O(N+E), where N is the number of nodes and E is edges

Expected Space Complexity: O(N)