DATA STRUCTURES (ITPC-203)

Trees and Graphs - Conclusion

Dr. Sanga Chaki

Department of Information Technology

Dr. B. R. Ambedkar National Institute of Technology, Jalandhar

Contents

1. Shortest Path algorithm:
   a) Warshal Algorithm
   b) Dijikstra Algorithm,

Shortest Path Algorithms

Shortest Path Algorithms

1. The shortest path problem involves finding the shortest path between two vertices (or nodes) A and B in a graph.
2. Two points to consider:
   a) Is there a path from A to B?
   b) If there is more than one path from A to B, which path is the shortest?
3. Can be applied to
   a) Unweighted graphs - Simply perform BFS
   b) Weighted graphs:
      i. Single Source All Destinations – Dijkstra’s Algorithm
      ii. All Pairs Shortest Paths – Floyd-Warshal Algorithm

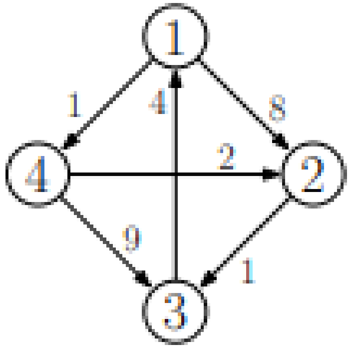
Floyd-Warshal Algorithm

1. The Floyd-Warshall algorithm finds out all the shortest paths from each vertex to all other vertices

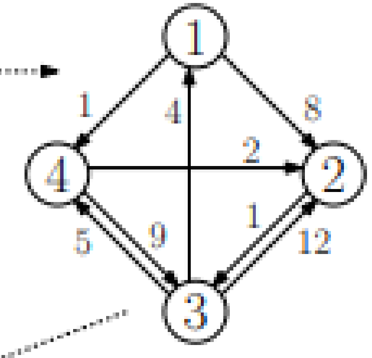
Algorithm 1: Pseudocode of Floyd-Warshall Algorithm

```
Data: A directed weighted graph G(V, E)
Result: Shortest path between each pair of vertices in G
for each d ∈ V do
  | distance[d][d] ← 0;
end
for each edge (s, p) ∈ E do
  | distance[s][p] ← weight(s, p);
end
n = cardinality(V);
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            if distance[i][j] > distance[i][k] + distance[k][j] then
              | distance[i][j] ← distance[i][k] + distance[k][j];
            end
        end
    end
end
```

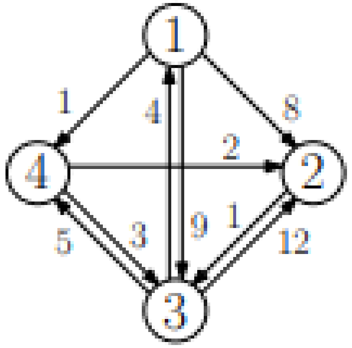
Floyd-Warshal Algorithm



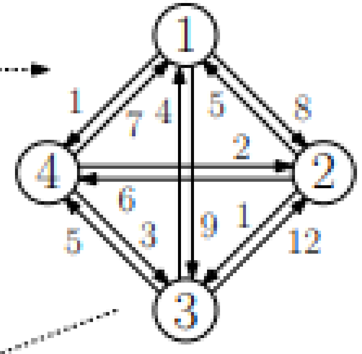
$$d^{(0)} = \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & \infty & 0 & \infty \\ \infty & 2 & 9 & 0 \end{bmatrix}$$



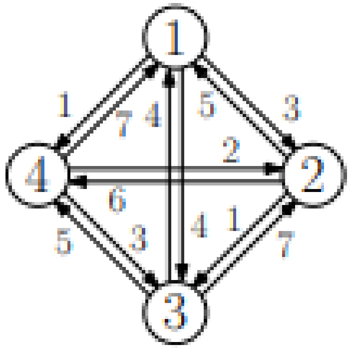
$$d^{(1)} = \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 9 & 0 \end{bmatrix}$$



$$d^{(2)} = \begin{bmatrix} 0 & 8 & 9 & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 3 & 0 \end{bmatrix}$$



$$d^{(3)} = \begin{bmatrix} 0 & 8 & 9 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 12 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix}$$

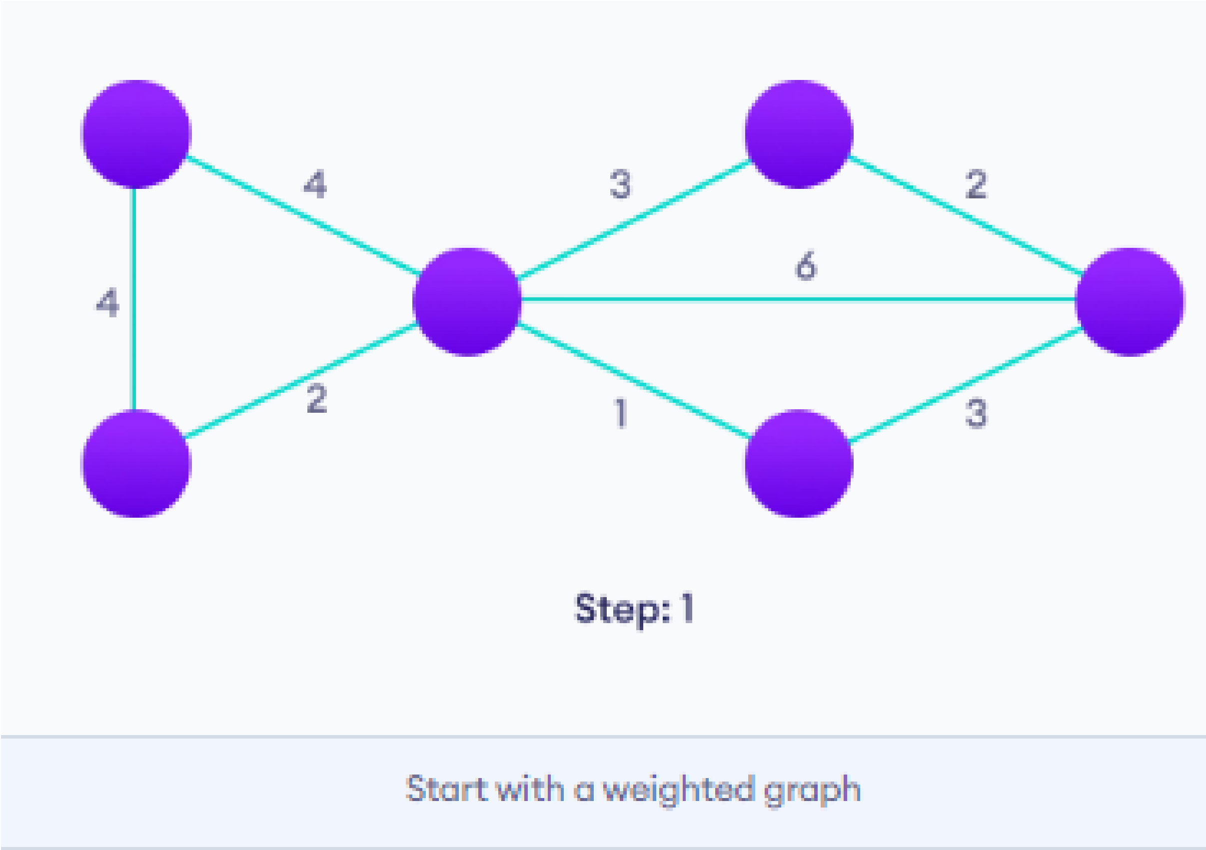


$$d^{(4)} = \begin{bmatrix} 0 & 3 & 4 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 7 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix}$$

$$\text{final} = \begin{bmatrix} 0 & 3 & 4 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 7 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix}$$

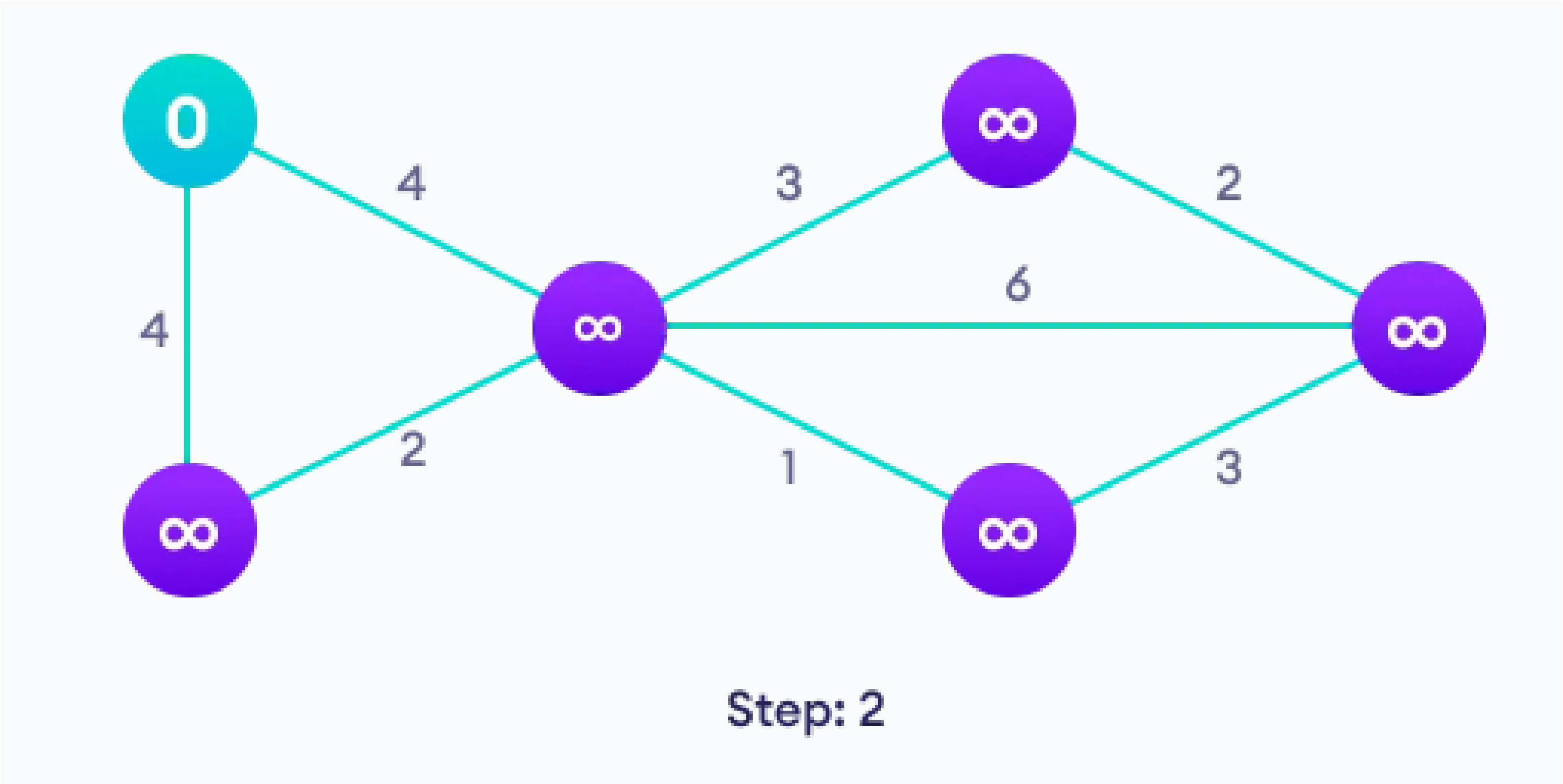
Dijkstra Algorithm

1. Determine a shortest path from a source vertex to each of the remaining vertices of G.



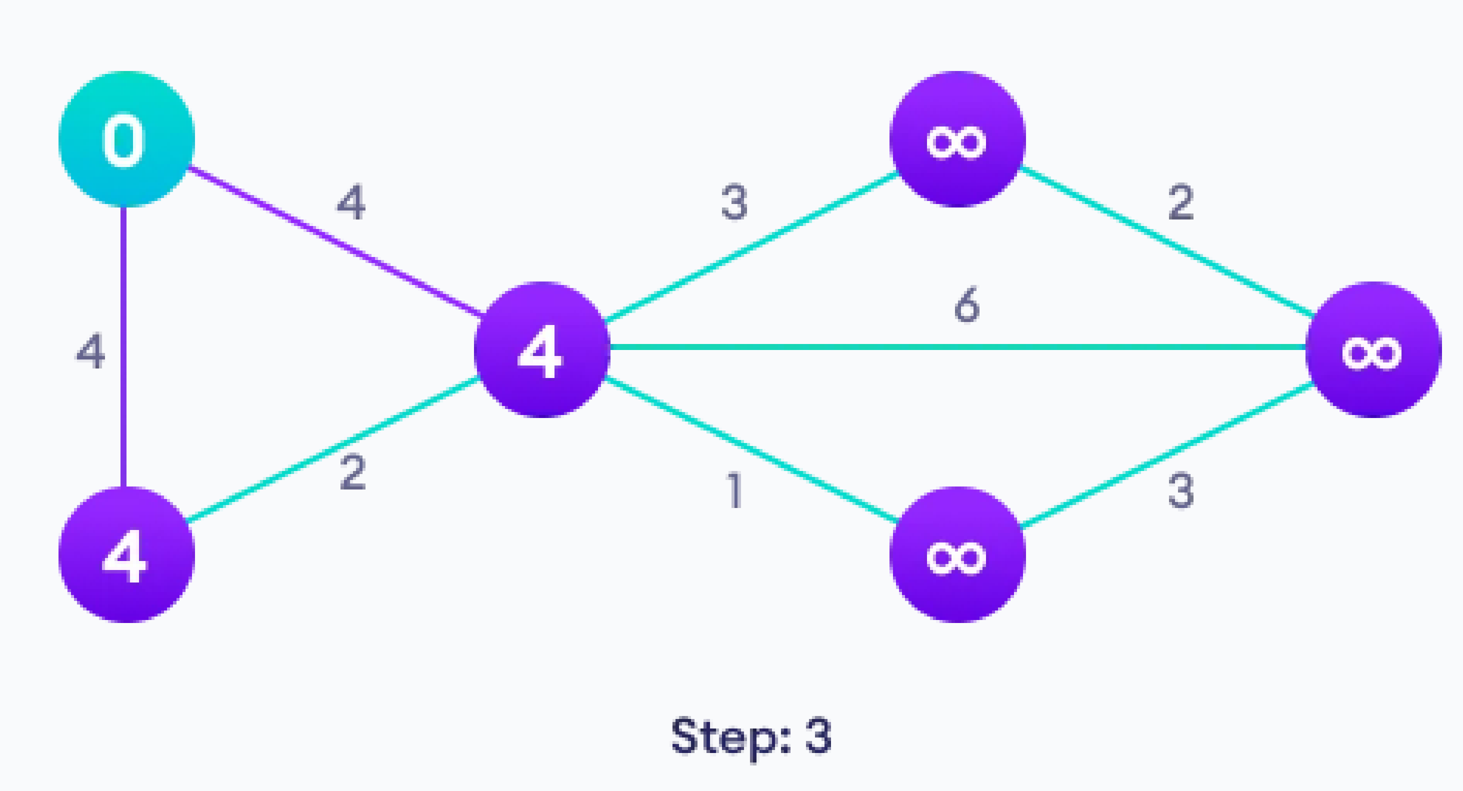
Dijkstra Algorithm

1. Choose a starting vertex and assign infinity path values to all other vertices – these will be determined.



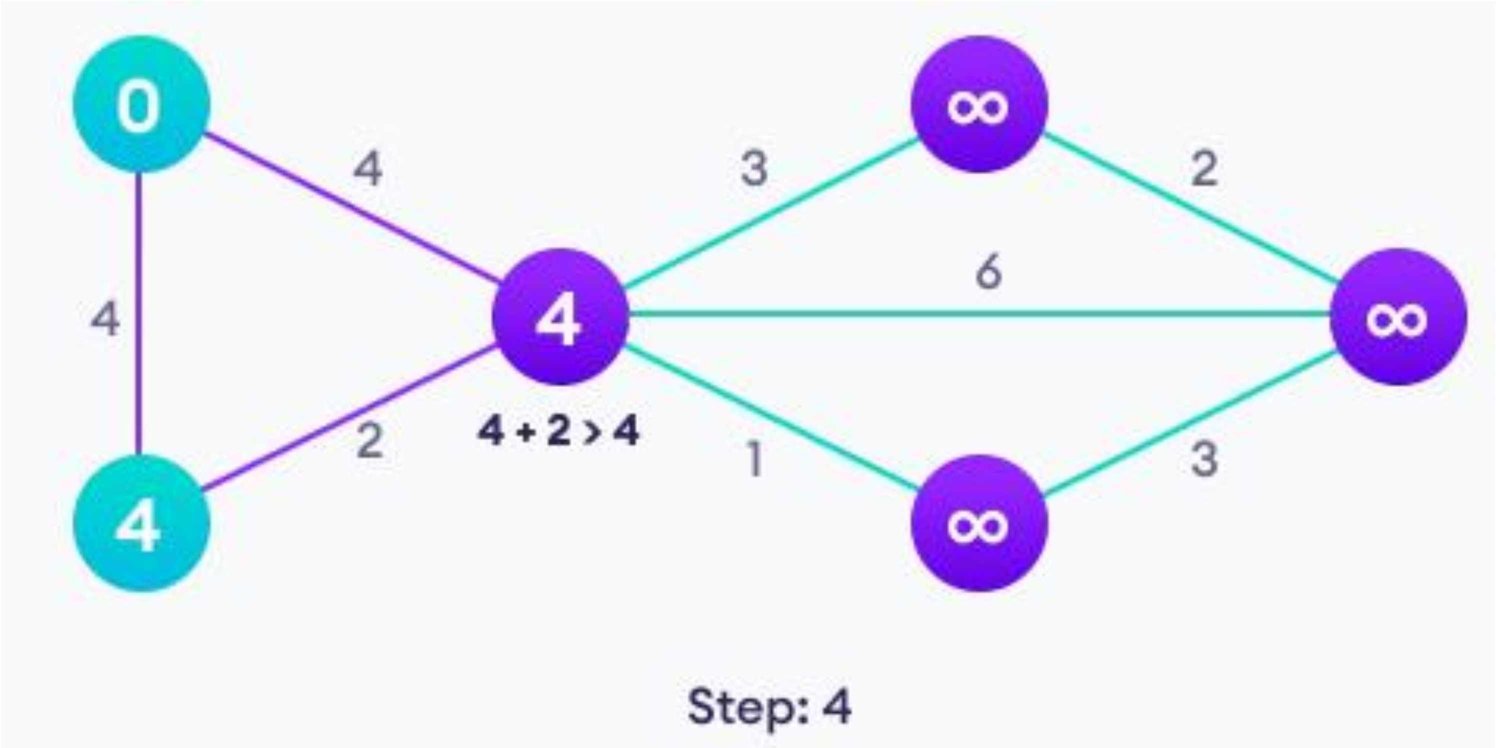
Dijkstra Algorithm

1. Go to each adjacent vertex and update its path length from the source.



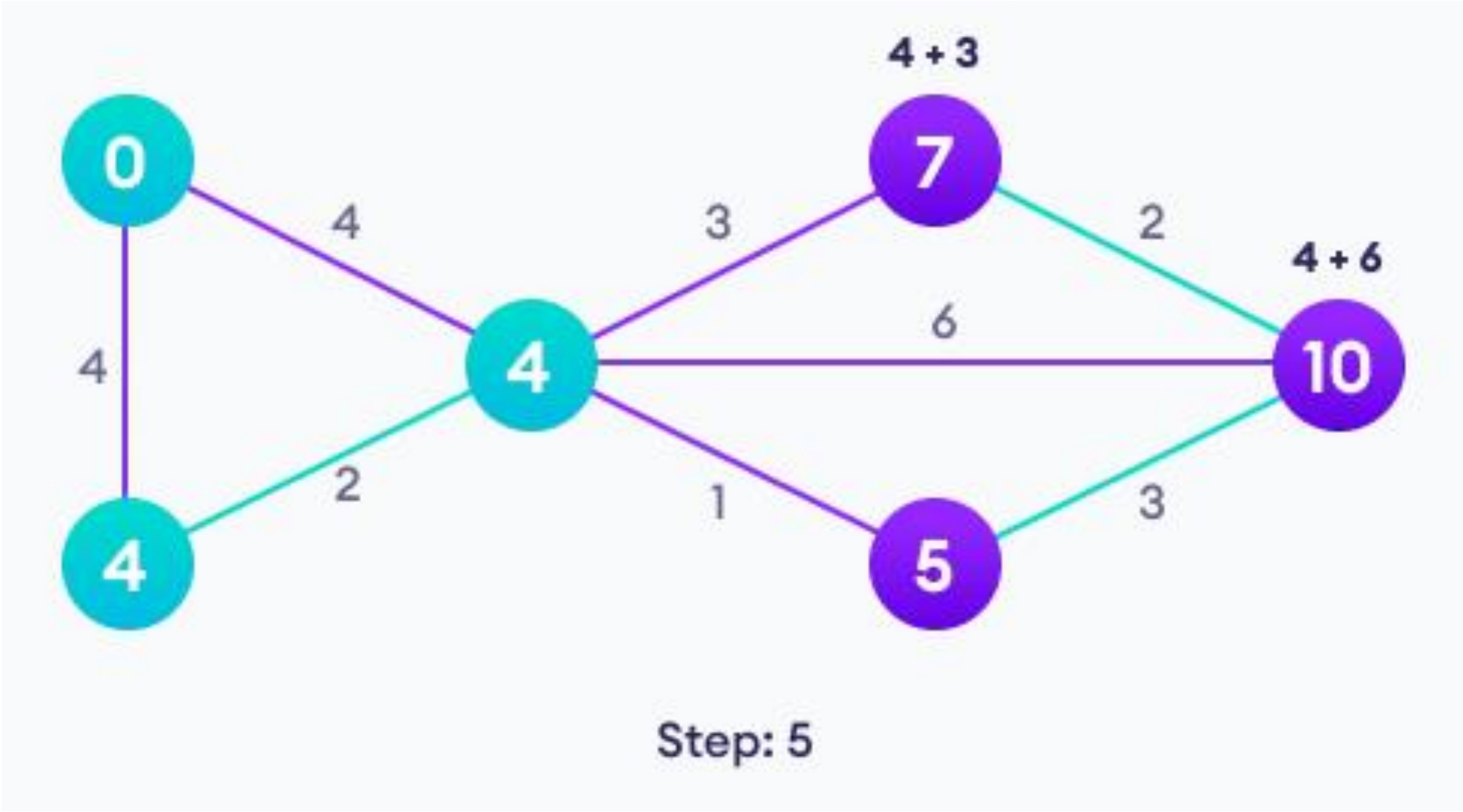
Dijkstra Algorithm

1. If the path length of the adjacent vertex is lesser than new path length, don't update it. Otherwise, update it.



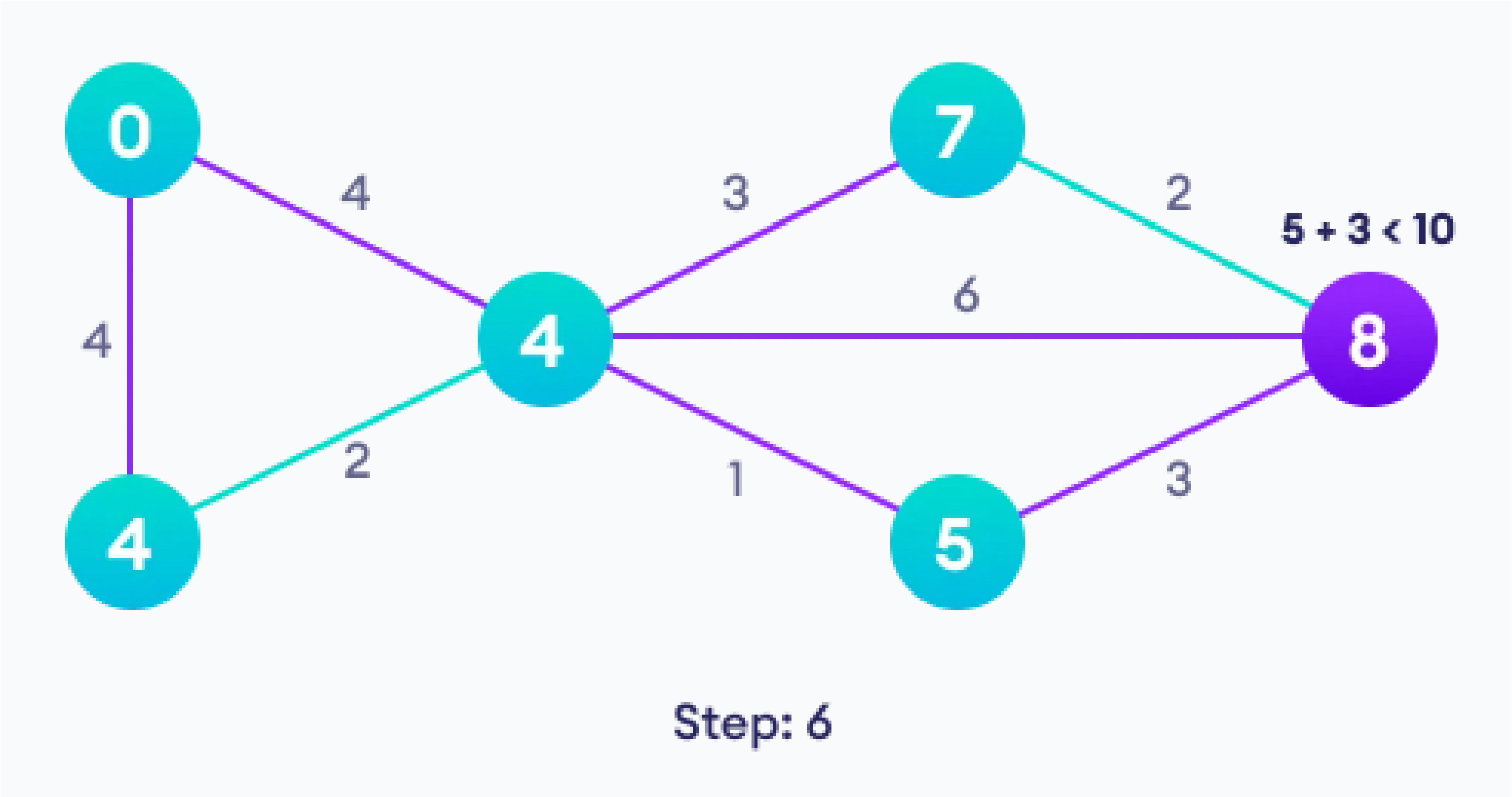
Dijkstra Algorithm

1. Avoid updating path lengths of already visited vertices



Dijkstra Algorithm

1. Pick the unvisited vertex with the least path length. So choose 5 before 7



Dijkstra Algorithm

1. The rightmost vertex has its path length updated twice