Lecture Eight

Dijkstra's Algorithm

MSc Karar Sadiq Mohsin

Email: karar.sadeq@uowa.edu.iq

DEFINITIONS

- Dijkstra's algorithm is a popular algorithm for solving many single-source shortest path problems having non-negative edge weight in the graphs
- i.e., it is to find the shortest distance between two vertices on a graph.
- It was conceived by Dutch computer scientist Edsger W. Dijkstra in 1956.

Dijkstra's Algorithm

- The algorithm maintains a set of visited vertices and a set of unvisited vertices.
- It starts at the source vertex and iteratively selects the unvisited vertex with the smallest tentative distance from the source.
- It then visits the neighbors of this vertex and updates their tentative distances if a shorter path is found.
- This process continues until the destination vertex is reached, or all reachable vertices have been visited.

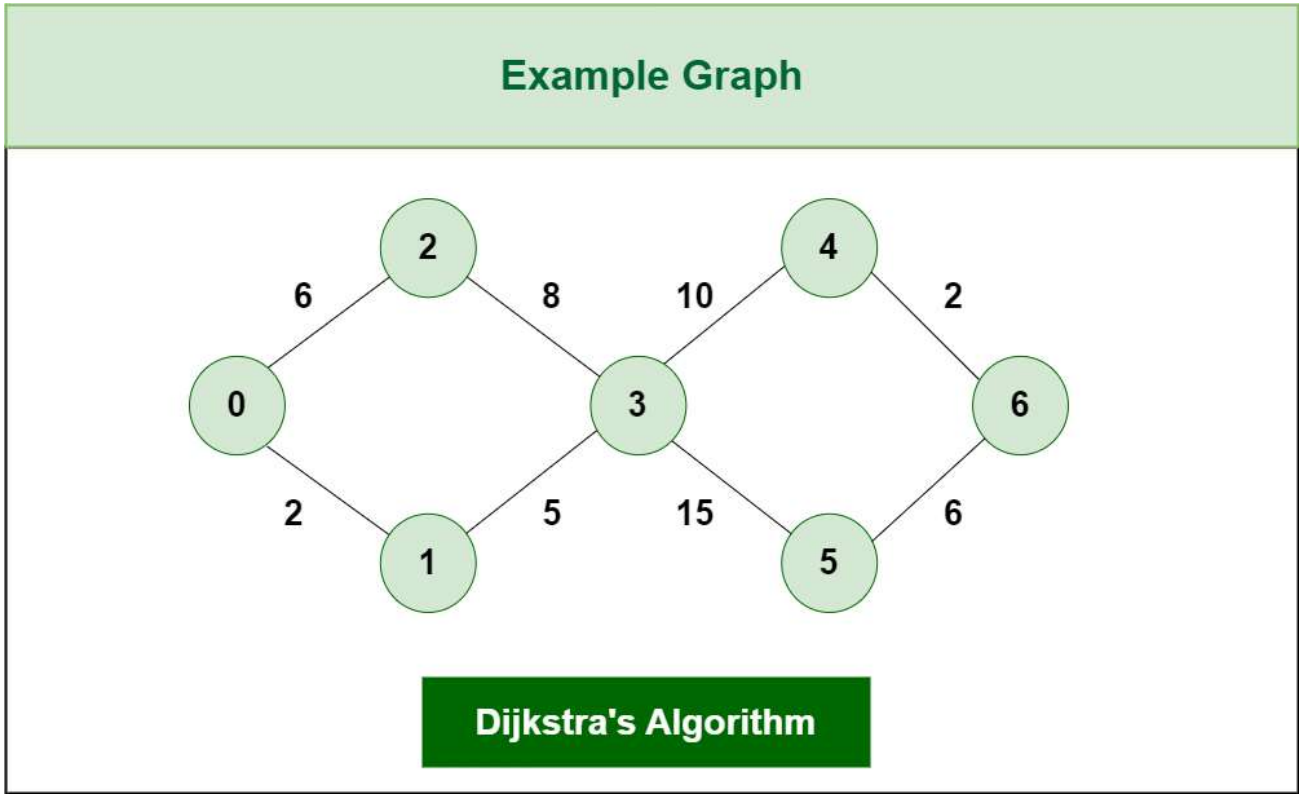
Algorithm for Dijkstra's Algorithm

1. Mark the source node with a current distance of 0 and the rest with infinity.
2. Set the non-visited node with the smallest current distance as the current node.
3. For each neighbor, N of the current node adds the current distance of the adjacent node with the weight of the edge connecting 0->1. If it is smaller than the current distance of Node, set it as the new current distance of N.
4. Mark the current node 1 as visited.
5. Go to step 2 if there are any nodes are unvisited.

How does Dijkstra's Algorithm work?

- Let's see how Dijkstra's Algorithm works with an example given below:
- Dijkstra's Algorithm will generate the shortest path from Node 0 to all other Nodes in the graph.

Consider the below graph

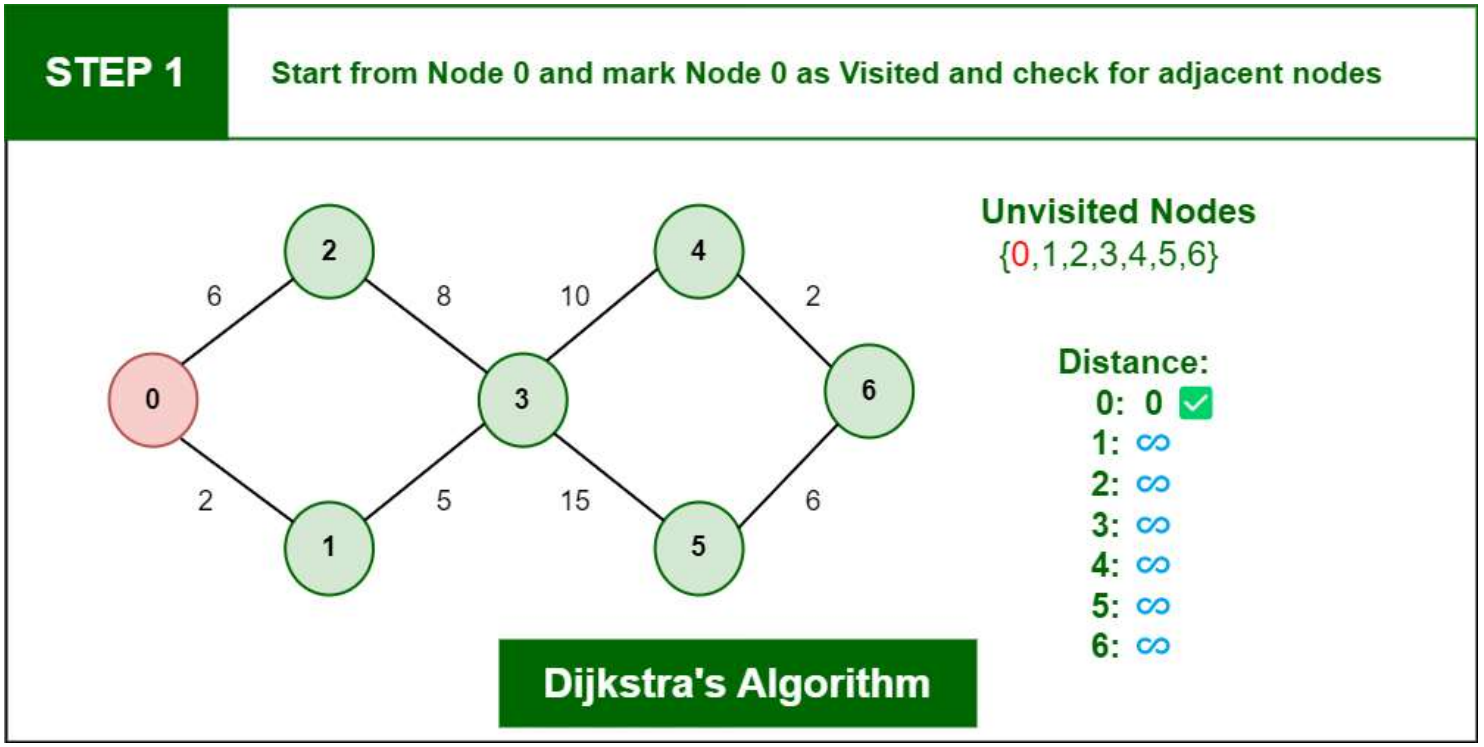


- The algorithm will generate the shortest path from node 0 to all the other nodes in the graph.
- For this graph, we will assume that the weight of the edges represents the distance between two nodes.
- As, we can see we have the shortest path from,
- Node 0 to Node 1, from
- Node 0 to Node 2, from
- Node 0 to Node 3, from
- Node 0 to Node 4, from
- Node 0 to Node 6.

Initially we have a set of resources given below :

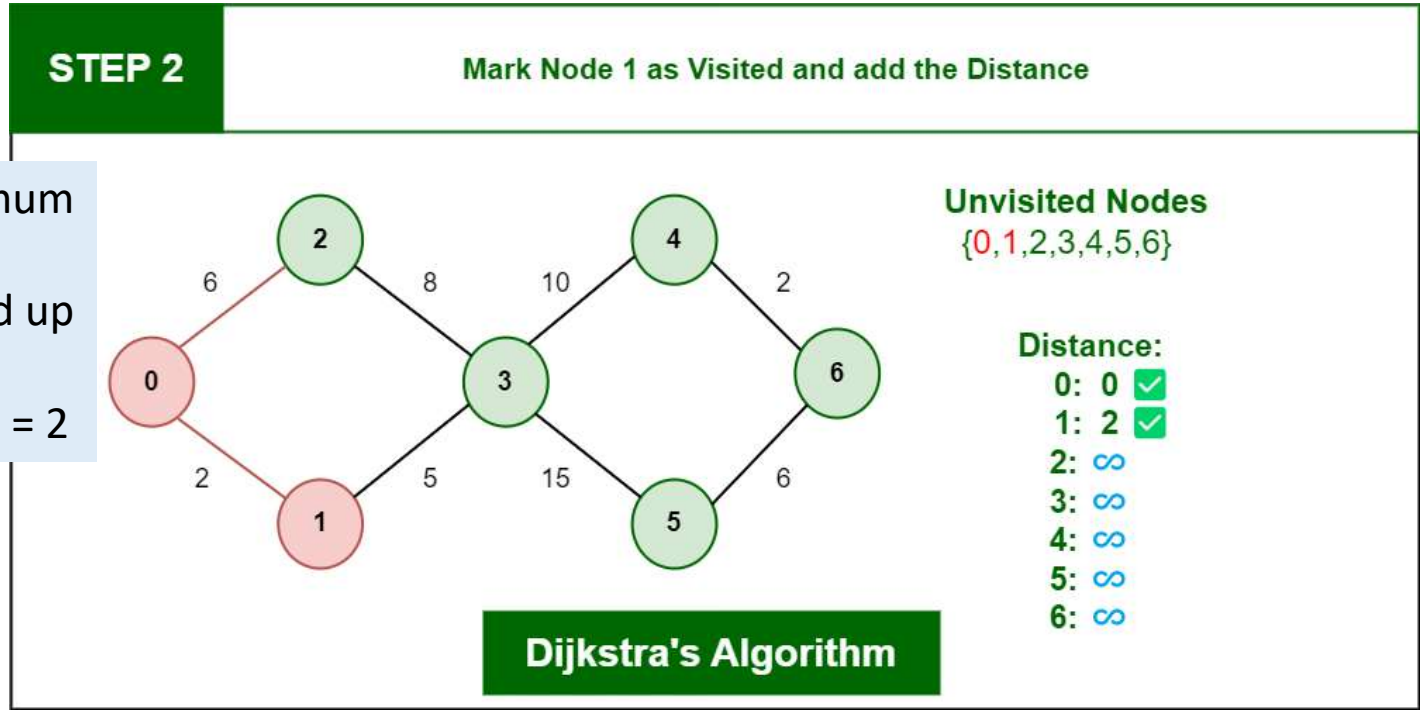
- The Distance from the source node to itself is 0. In this example the source node is 0.
- The distance from the source node to all other node is unknown so we mark all of them as infinity.
- Example: 0 -> 0, 1-> ∞,2-> ∞,3-> ∞,4-> ∞,5-> ∞,6-> ∞.
- we'll also have an array of unvisited elements that will keep track of unvisited or unmarked Nodes.
- Algorithm will complete when all the nodes marked as visited and the distance between them added to the path.
- Unvisited Nodes:- 0 1 2 3 4 5 6.

Step 1: Start from Node 0 and mark Node as visited as you can check in below image visited Node is marked red.



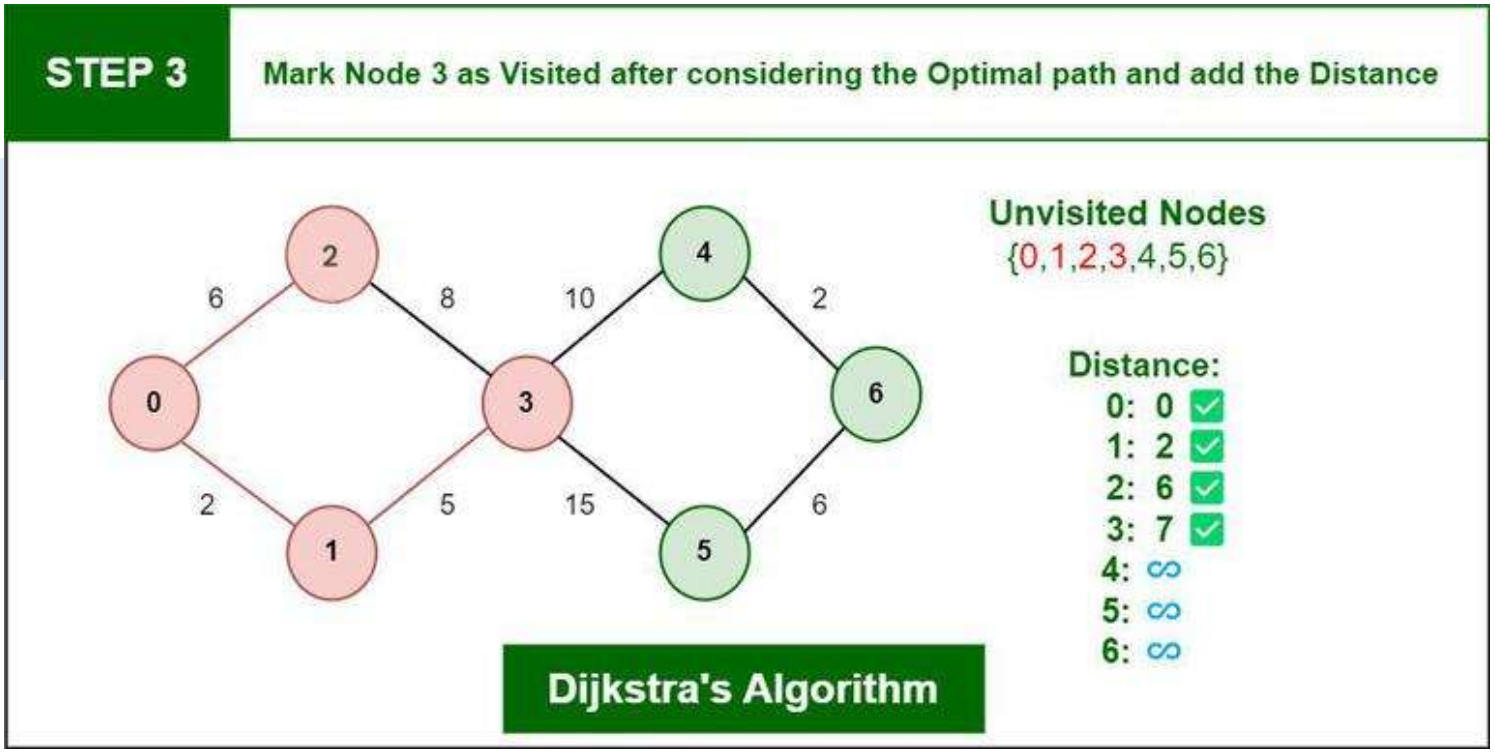
Step 2: Check for adjacent Nodes, Now we have two choices (Either choose Node1 with distance 2 or either choose Node 2 with distance 6) and choose Node with minimum distance.

In this step Node 1 is Minimum distance adjacent Node, so marked it as visited and add up the distance.
Distance: Node 0 -> Node 1 = 2



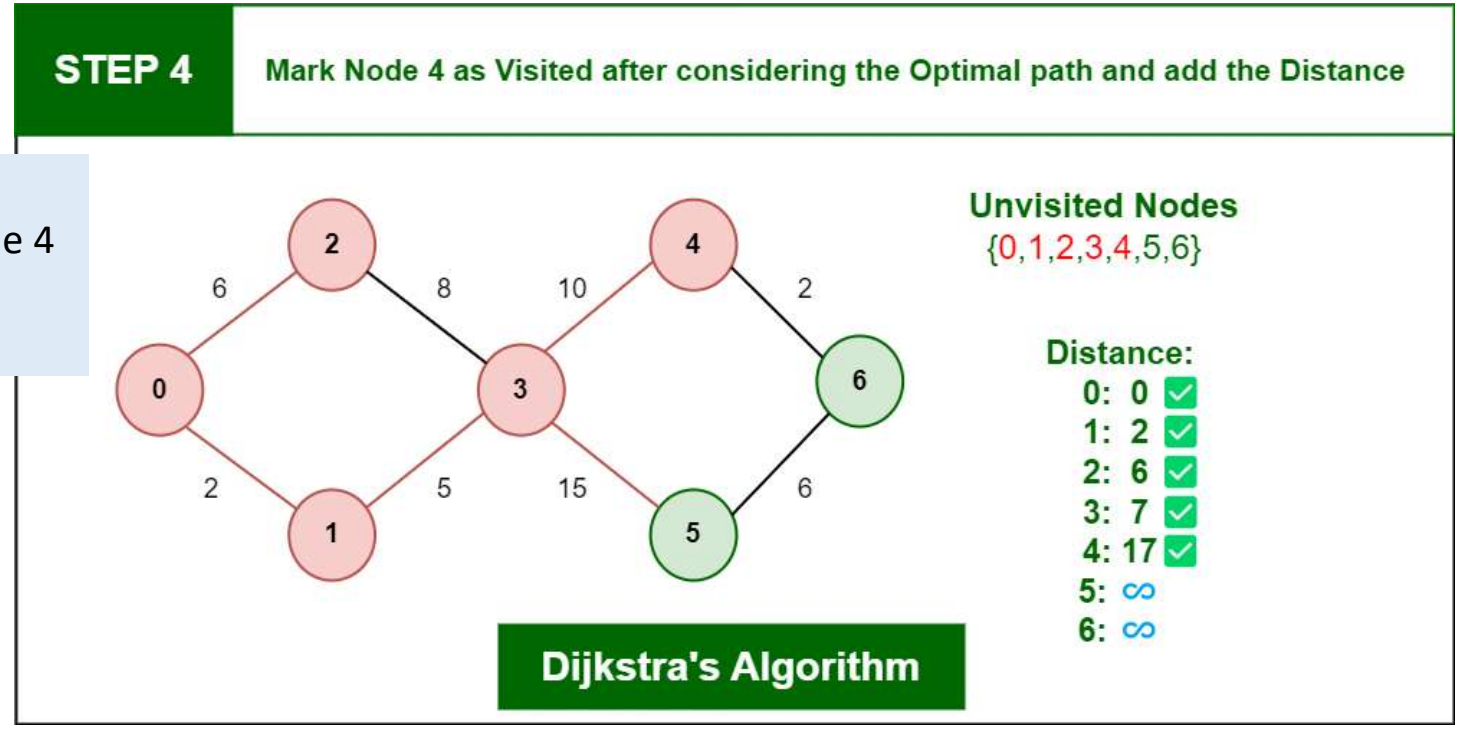
Step 3: Then Move Forward and check for adjacent Node which is Node 3, so marked it as visited and add up the distance, Now the distance will be:

Distance:
Node 0 -> Node 1 -> Node 3
= 2 + 5 = 7



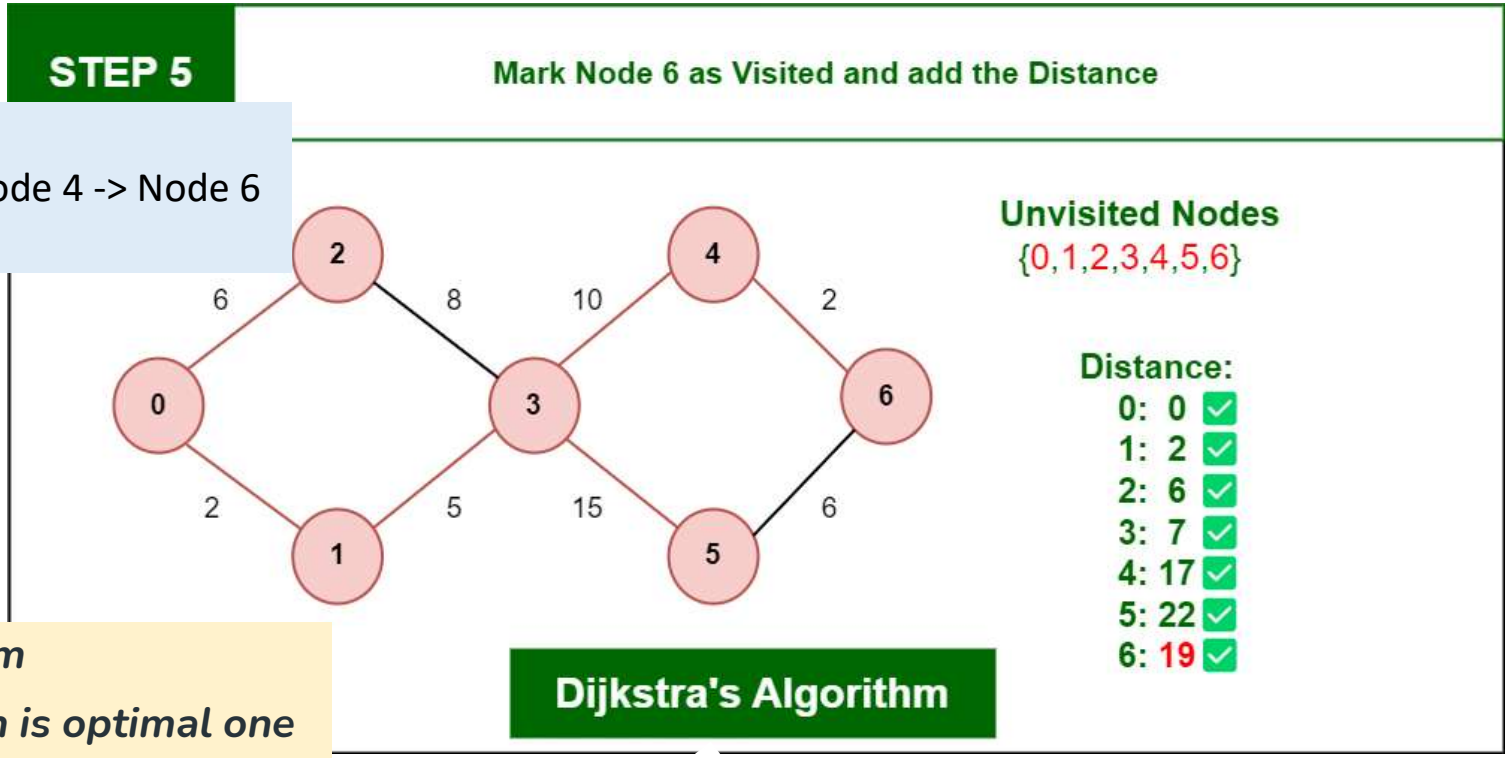
Step 4: Again, we have two choices for adjacent Nodes (Either we can choose Node 4 with distance 10 or we can choose Node 5 with distance 15) so choose Node with minimum distance. In this step Node 4 is Minimum distance adjacent Node, so marked it as visited and add up the distance.

Distance:
Node 0 -> Node 1 -> Node 3 -> Node 4
= 2 + 5 + 10 = 17



Step 5: Again, Move Forward and check for adjacent Node which is Node 6, so marked it as visited and add up the distance, Now the distance will be:

Distance:
Node 0 -> Node 1 -> Node 3 -> Node 4 -> Node 6
= 2 + 5 + 10 + 2 = 19



So, the Shortest Distance from the Source Vertex is 19 which is optimal one

Homework

- Implement Dijkstra's Algorithm in any programming language of your choice (e.g., Python, Java, C++). Your program should: Accept a graph as input (preferably as an adjacency list or matrix).
- Take the starting node as input from the user.
- Output the shortest distances from the starting node to all other nodes

Need for Dijkstra's Algorithm (Purpose and Use-Cases)

- The need for Dijkstra's algorithm arises in many applications were finding the shortest path between two points is crucial.
- For example, It can be used in the routing protocols for computer networks and used by map systems to find the shortest path between starting point and the Destination.

Can Dijkstra's Algorithm work on both Directed and Undirected graphs?

- Yes, Dijkstra's algorithm can work on both directed graphs and undirected graphs as this algorithm is designed to work on any type of graph if it meets the requirements of having non-negative edge weights and being connected.

Can Dijkstra's Algorithm work on both Directed and Undirected graphs?

- In a **directed graph**, each edge has a direction, indicating the direction of travel between the vertices connected by the edge.
- In this case, the algorithm follows the direction of the edges when searching for the shortest path.

Can Dijkstra's Algorithm work on both Directed and Undirected graphs?

- In an undirected graph, the edges have no direction, and the algorithm can traverse both forward and backward along the edges when searching for the shortest path.

See You Next Lecture