Dijkstra's algorithm:
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.

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 neighbours 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.

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

- **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.
- **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.

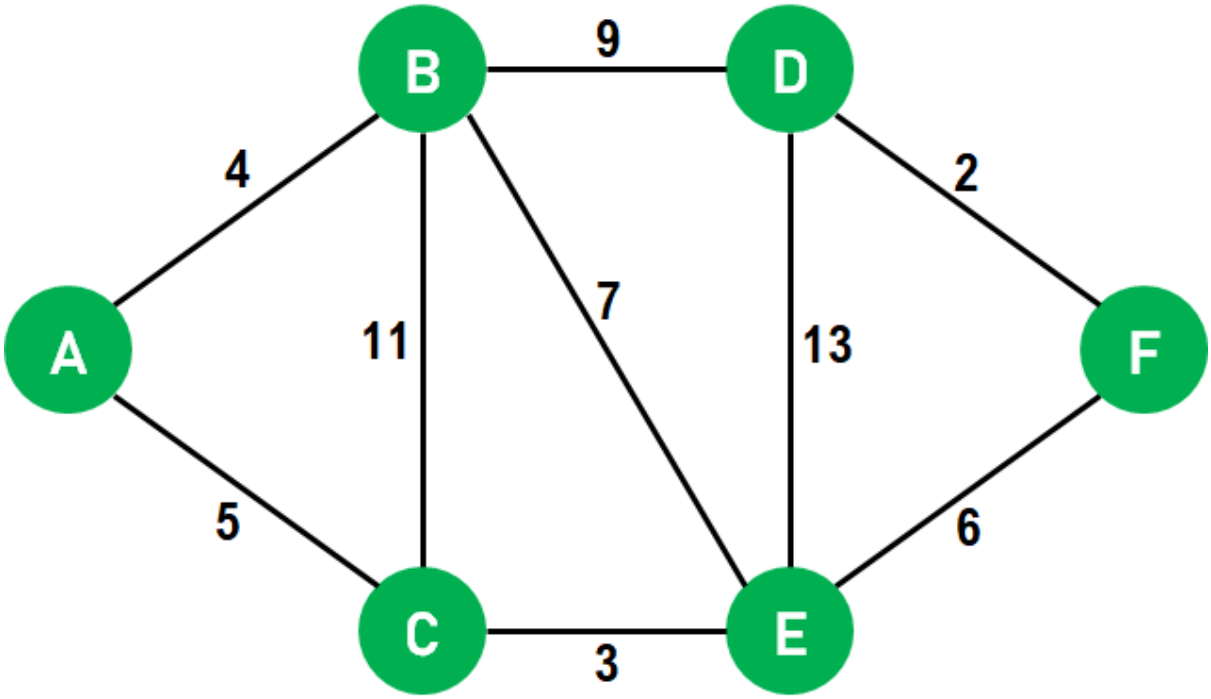
Ever wondered how does Google Maps finds the shortest and fastest route between two places?

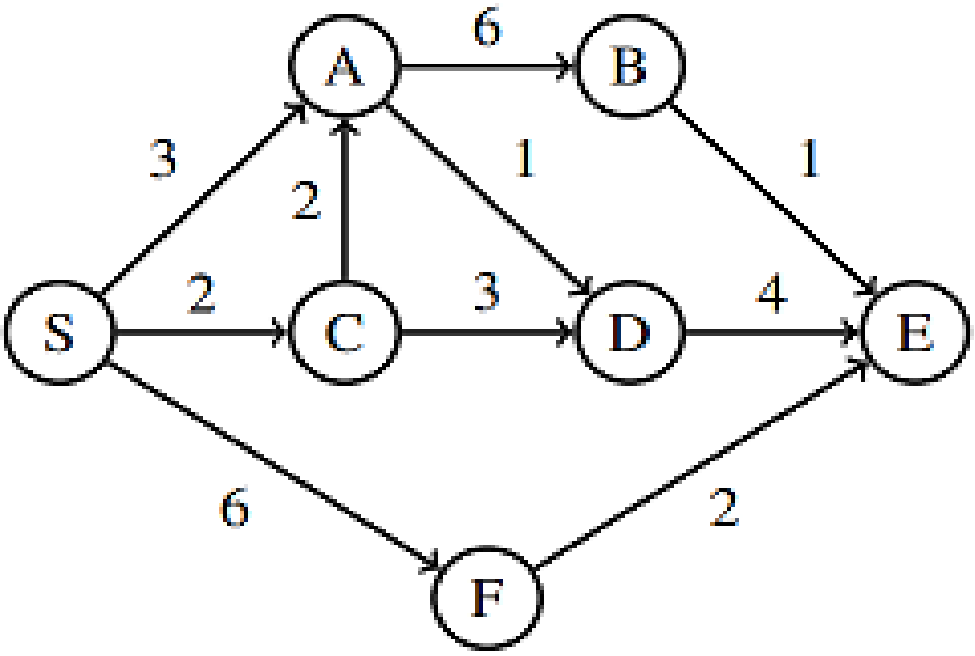
Well, the answer is **Dijkstra's Algorithm**. **Dijkstra's Algorithm** is a Graph algorithm **that finds the shortest path** from a source vertex to all other vertices in the Graph (single source shortest path). It is a type of Greedy Algorithm that only works on Weighted Graphs having positive weights. The time complexity of Dijkstra's Algorithm

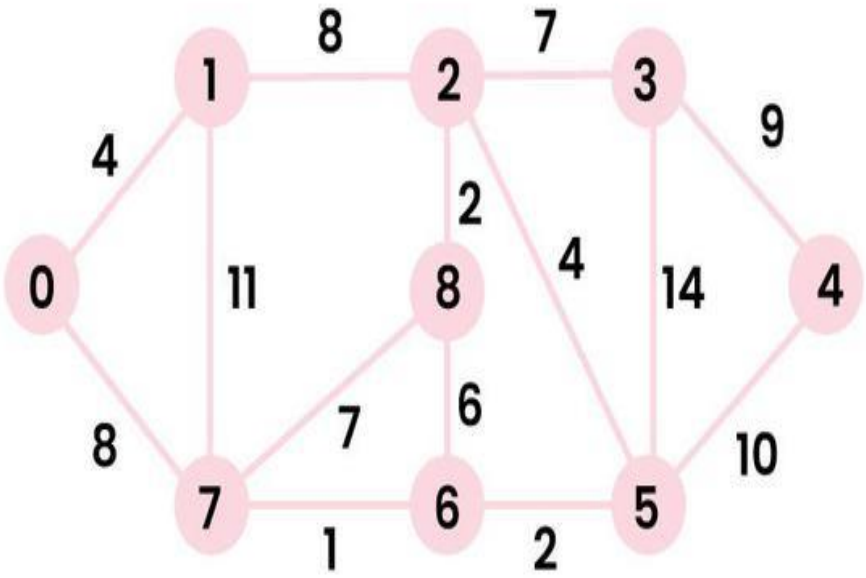
is **O(V^2)** with the help of the adjacency matrix representation of the graph. This time complexity can be reduced to **O((V + E) log V)** with the help of an adjacency list representation of the graph, where **V** is the number of vertices and **E** is the number of edges in the graph

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**.


B
D
A
F
C
E
9
4
2
11
7
13
5
6
3


A
B
S
C
D
E
F
6
3
1
1
2
2
3
4
6
2



Working of Dijkstra's Algorithm

Applications of Dijkstra's Algorithm:

- **Google maps** uses Dijkstra algorithm to show shortest distance between source and destination.
- In **computer networking**, Dijkstra's algorithm forms the basis for various routing protocols, such as OSPF (Open Shortest Path First) and IS-IS (Intermediate System to Intermediate System).
- Transportation and traffic management systems use Dijkstra's algorithm to optimize traffic flow, minimize congestion, and plan the most efficient routes for vehicles.
- Airlines use Dijkstra's algorithm to plan flight paths that minimize fuel consumption, reduce travel time.
- Dijkstra's algorithm is applied in electronic design automation for routing connections on integrated circuits and very-large-scale integration (VLSI) chips.