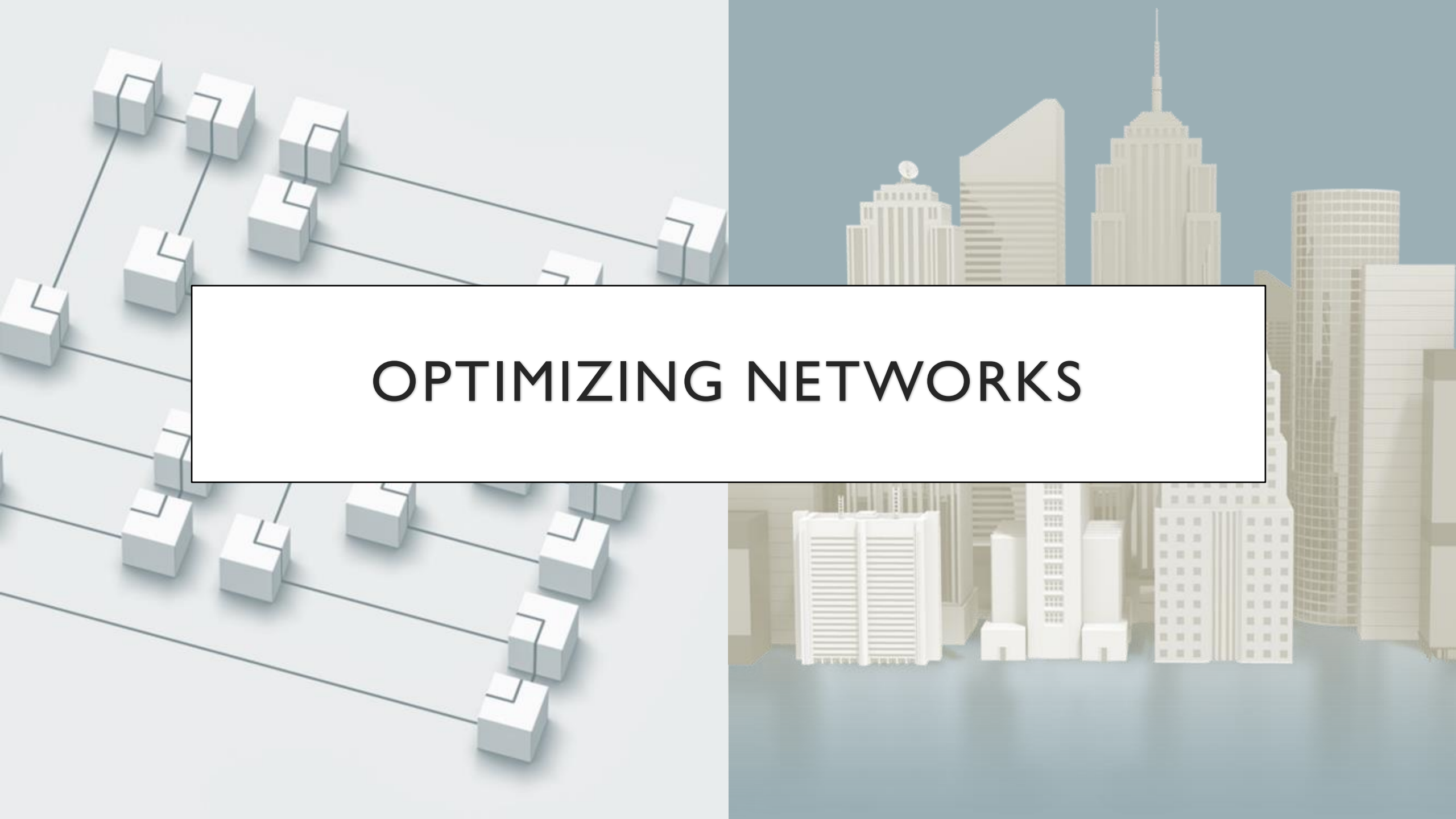


OPTIMIZING NETWORKS

NETWORK ALGORITHMS BACKGROUND

- Network algorithms are computational procedures designed to solve problems related to networks or graphs.
- A network or graph consists of nodes (vertices) and edges connecting these nodes, representing relationships or connections between entities. In the context of transportation networks, nodes represent locations (such as stations or intersections), and edges represent connections between them (roads, rails, or other links).

- **Importance of Network Algorithms:**
- Network algorithms play a crucial role in various domains such as computer science, engineering, biology, and social sciences.
- They enable efficient problem-solving in tasks like routing, optimization, clustering, and analysis of complex networks.

- **Scope of Network Algorithms:**
- Shortest Path algorithms, Minimum Spanning Trees, Graph Traversal, Flow Networks…etc

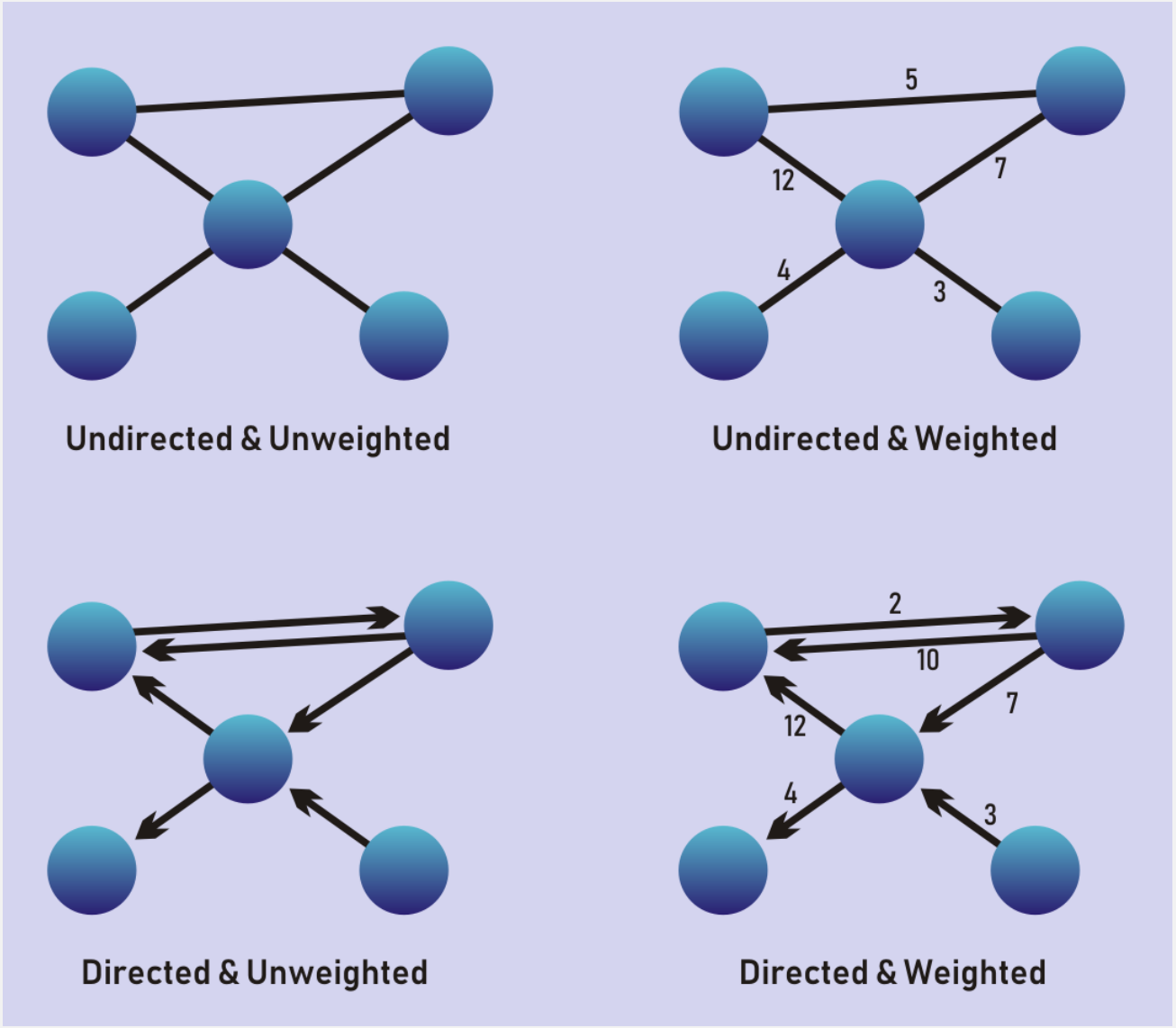
GRAPH THEORY BASICS

Definition:

- A graph is a mathematical structure consisting of a set of vertices (nodes) and a set of edges (connections) that establish relationships between pairs of vertices.
- Directed graphs have edges with a specific direction, while undirected graphs do not.

Types of Graphs:

- Simple, weighted/unweighted, directed/undirected
- Other categories also include cyclic/acyclic, connected/disconnected … etc



MORE BACKGROUND

- **Greedy Algorithms**:
  - Greedy algorithms make locally optimal choices at each step with the hope of finding a global optimum.
  - Pros: efficiency, (when greedy approach works), simplicity, optimality in some cases, space efficiency
  - Cons: lack of global optimality, dependent on problem structure, difficulty in Identifying Greedy Choices, can't backtrack
- **Non-Greedy Algorithms**
  - Non greedy algorithms consider a broader range of possibilities to find the globally optimal solution.
  - Pros: Can guarantee optimal solutions, flexibility, backtracking.
  - Cons: often higher time/space complexity, may not be as efficient, can be more complex.

DIJKSTRA'S ALGORITHM

Steps:

1) Mark all destinations as unvisited
2) Assume all nodes have a tentative value of infinity before visiting
3) Determine the current node
4) At the current node, compare the distance to any unvisited neighbor nodes
5) Update the shortest distance if applicable
6) Mark current node as visited
7) Choose the next current node as unvisited node with shortest distance

Objective: Find the shortest path from intersection A to intersection F, considering the travel times on each road.

Node	Shortest Distance	Prior Node
A	0	-
B	5	A
C	~~10~~ 7	~~A~~ B
D	~~13~~ 12	~~B~~ C
E	11	C
F	17	E

MINIMUM SPANNING TREES

A minimum spanning tree (MST) is a graph theory concept utilized in algorithms and optimization. It is a subset of the edges of a connected, undirected graph that connects all the vertices together excluding cycles designed for the minimum possible total edge weight.

A minimum spanning tree identifies the most efficient way to connect all the nodes (or locations) in a network while minimizing the total cost (distance, time, or other metric). In urban design, this could represent the optimal layout of roads, pathways, or public transportation routes to ensure efficient connectivity between different areas within a city.

By identifying the minimum spanning tree of such networks, urban planners can reduce travel distances, improve connectivity, decrease expenses, and enhance the overall efficiency of these systems.

Applicable to Road Networks, Railway Networks, Telecommunication Networks, Supply Chain Logistics…etc

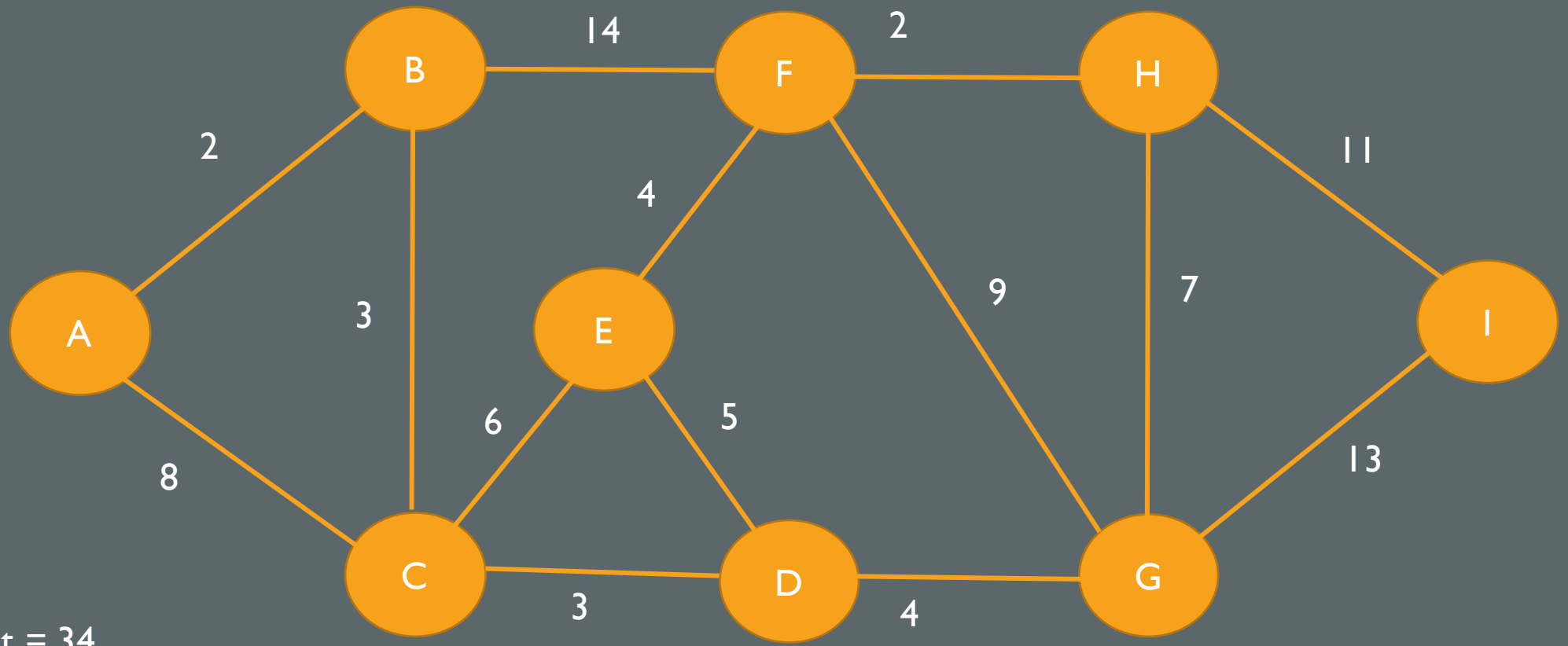
Usage: Prim's algorithm is suitable for scenarios where the objective is to find the minimum spanning tree of a connected, undirected graph with weighted edges. It's commonly used in network design, circuit design, and clustering algorithms.

In terms of urban planning it can be utilized for network optimization techniques to solve facility location problems, such as determining the optimal locations for public services (e.g., bus stops, fire stations) to maximize accessibility and coverage within an urban area.

Prims Algorithm for Minimum Spanning Trees

- **Step 1**: Initialize an empty set to represent the MST and select an arbitrary vertex as the starting point. Add this vertex to the MST set.
- **Step 2**: Repeat the following steps until all vertices are included in the MST:
  - **Step 2a**: Identify all edges that connect vertices in the MST set to vertices outside the MST set.
  - **Step 2b**: Select the edge with the minimum weight among these edges.
  - **Step 2c**: Add the vertex at the other end of this selected edge to the MST set.
  - **Step 2d**: Add this selected edge to the MST.
- **Step 3**: End the algorithm when all vertices are included in the MST.

PRIMS ALGORITHM FOR MINIMUM SPANNING TREES



=2
+3
+3
+4
+5
+4
+2
+11

Total weight = 34

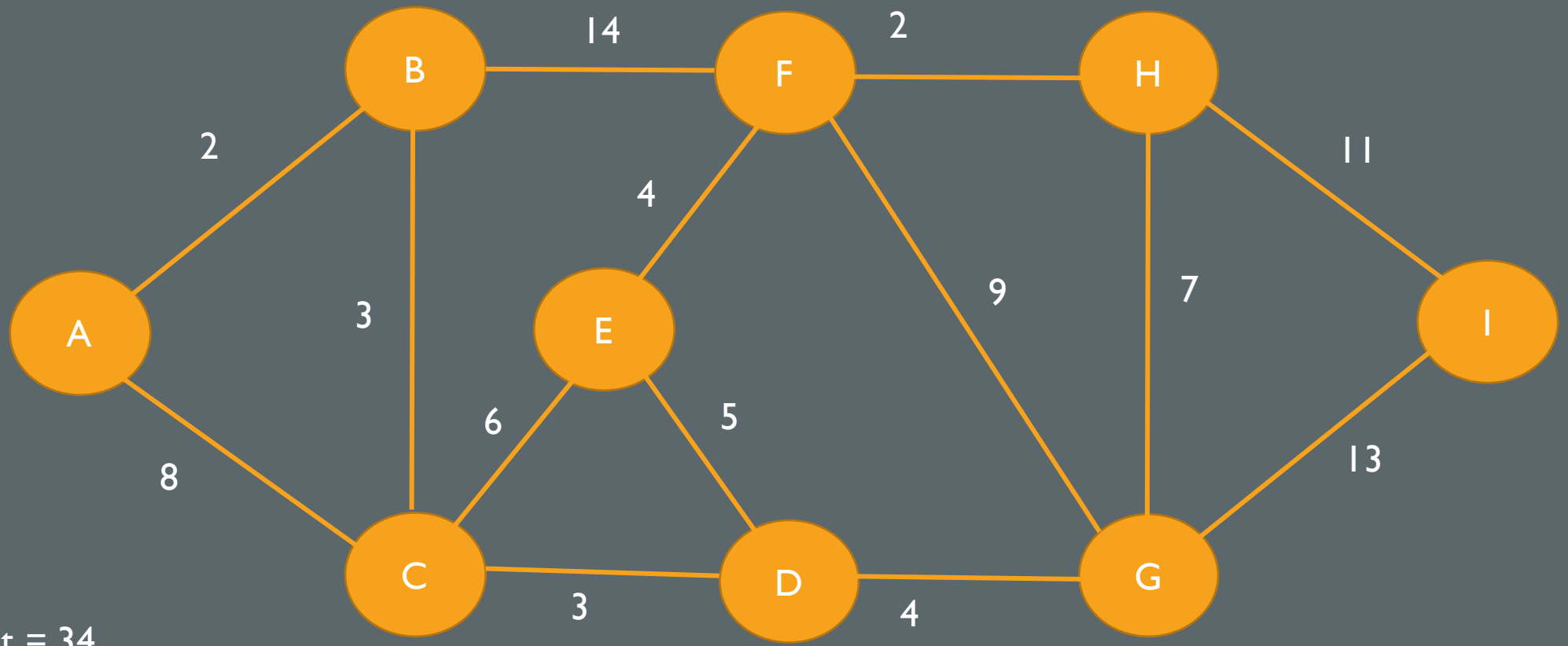
Kruskal's Algorithm for Minimum Spanning Trees

- Kruskal's algorithm can be relevant where the focus is on optimizing the overall infrastructure layout rather than ensuring specific connectivity paths. Kruskal's algorithm can be used to determine the minimal set of connections required to ensure coverage while minimizing costs or resource usage.
- Additionally, in transportation planning, Kruskal's algorithm can help in optimizing the layout of public transportation routes or railway lines within a city, considering factors such as passenger demand, geographic features, and construction costs.

- **Step 1**: Sort all edges of the graph in order of their weights.
- **Step 2**: Initialize an empty set to represent the MST.
- **Step 3**: Iterate through the sorted edges:
  - **Step 3a**: Pick the edge with the smallest weight.
  - **Step 3b**: Check if adding this edge to the MST set creates a cycle. If not, include the edge in the MST set.
  - **Step 3c**: If adding the edge creates a cycle, discard the edge.
- **Step 4**: Repeat this process until the MST set contains $V-1$ edges, where V is the number of vertices in the graph.
- **Step 5**: Terminate the algorithm when the MST set contains $V-1$ edges.

KRUSKALS ALGORITHM FOR MINIMUM SPANNING TREES

=2
+3
+3
+4
+5
+4
+2
+11



Total weight = 34

INTERPRETING RESULTS

- **Prim's Algorithm Weight (34)**: The total weight of the minimum spanning tree generated by Prim's algorithm is 34 units.

- **Kruskal's Algorithm Weight (34)**: The total weight of the minimum spanning tree generated by Kruskal's algorithm is 34 units.

1. Weight of the MST:

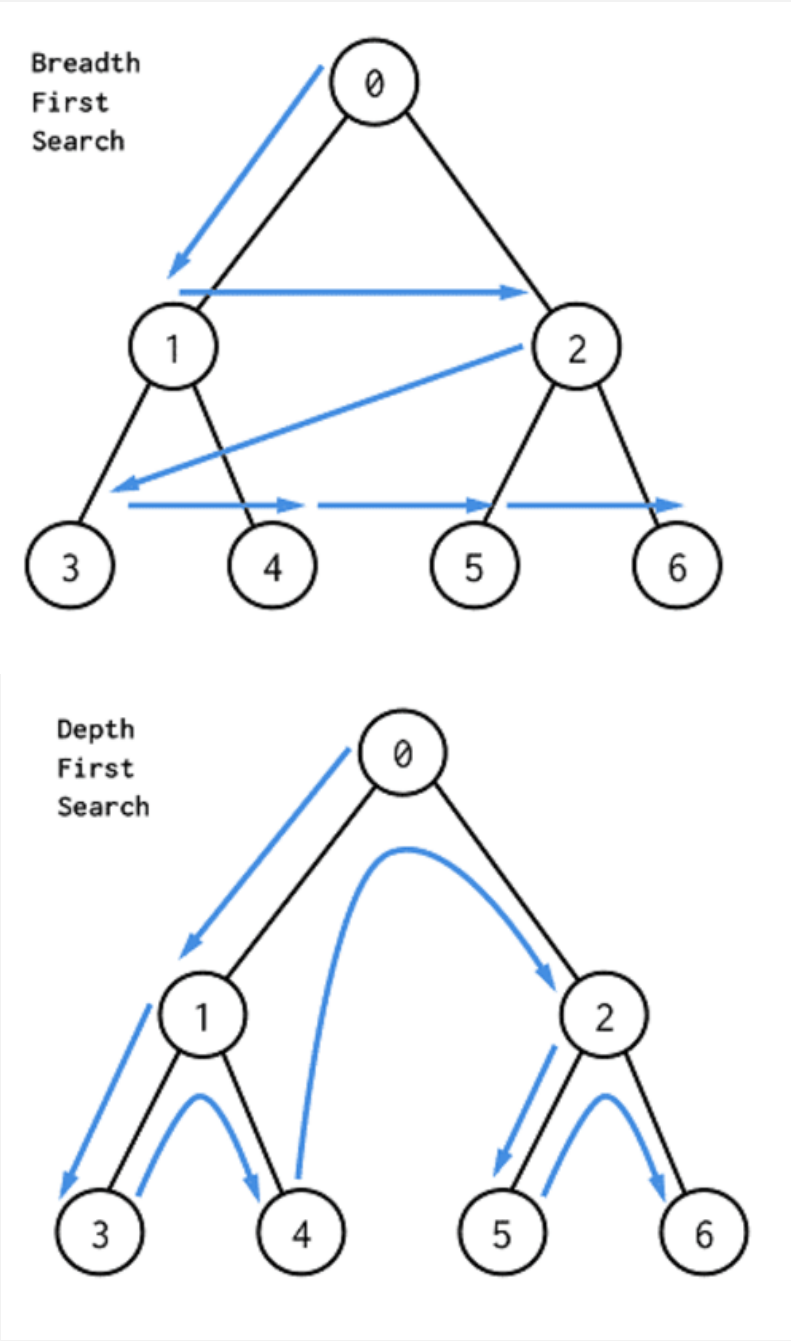
1. The weight of the MST indicates the total cost or length required to span all vertices of the graph while forming a tree with the minimum total weight. A lower weight implies a more efficient or cheaper spanning tree in terms of the given weights associated with the edges.

2. Impact of Graph Structure and Edge Weights:

1. The structure of the graph and the distribution of edge weights can influence the results obtained by both algorithms. If there are multiple edges with the same weight or if the graph has certain properties (e.g., dense vs. sparse), it may affect the choice of edges made by the algorithms and thus lead to different MSTs.

MORE ALGORITHMS

- **Breadth-First Search (BFS)**:
  - BFS is a graph traversal algorithm that explores all the vertices in the graph level by level.
  - It starts at a specified vertex and explores its neighbors before moving to the next level.
  - Useful for finding the shortest path in an unweighted graph and for exploring connected components.
- **Depth-First Search (DFS)**:
  - DFS is a graph traversal algorithm that explores as far as possible along each branch before backtracking.
  - It starts at a specified vertex and explores as deeply as possible along each branch before backtracking.
  - Useful for topological sorting, cycle detection, and maze solving.

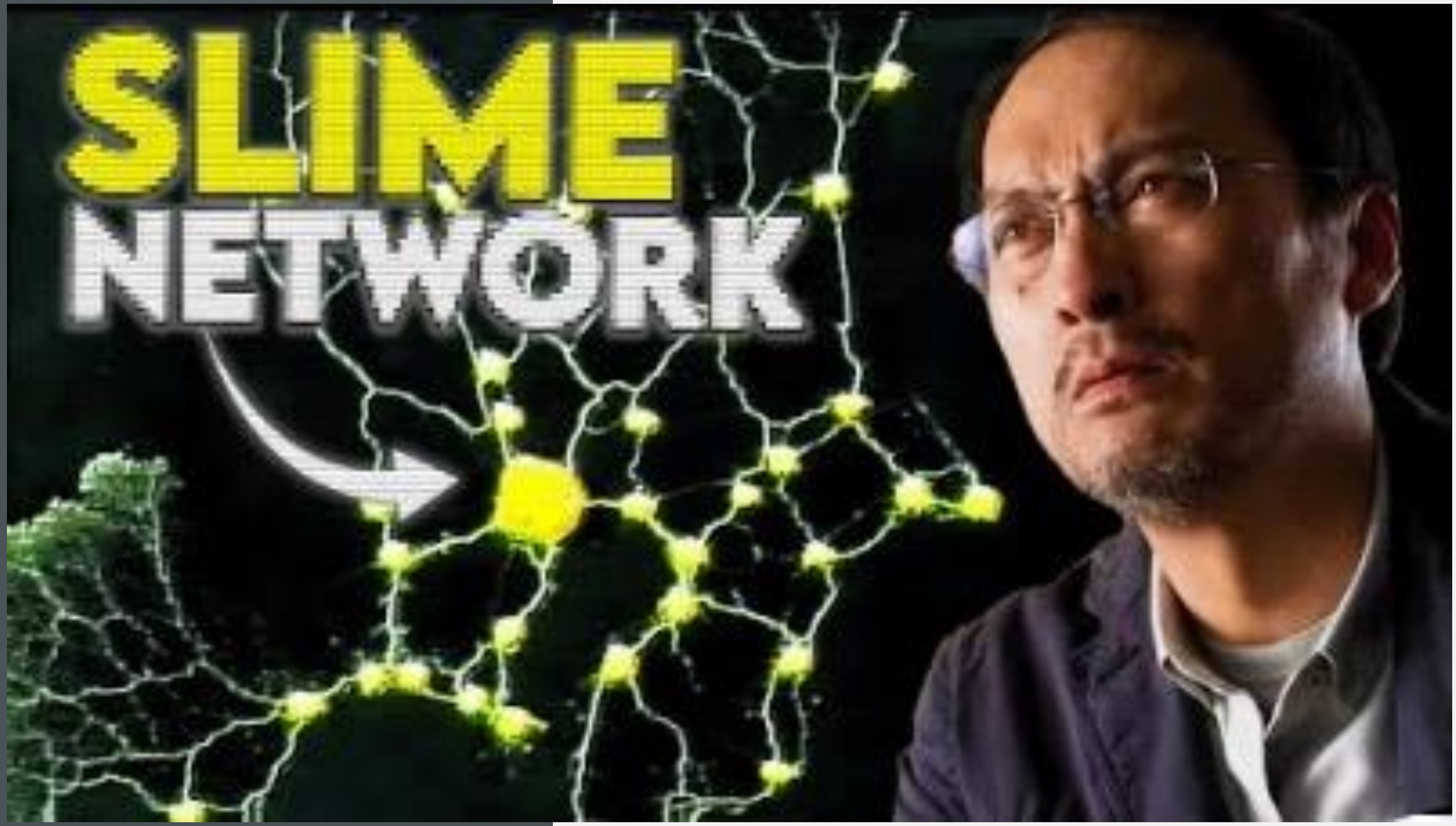


Ant Colony Optimization

Ant Colony Optimization (ACO) is a metaheuristic optimization algorithm inspired by the foraging behavior of ants. It was introduced by Marco Dorigo in the 1990s. The basic idea behind ACO is to mimic the way ants find the shortest paths between their nest and food sources.

ACO applications in combinatorial optimization problems including the traveling salesman problem and the minimum spanning tree problem.

1. **Initialization**:
    1. Initialize a population of artificial ants.
    2. Place them randomly on the problem space.
2. **Construct Solutions**:
    1. Each ant constructs a solution by iteratively selecting edges (or components) based on certain criteria, typically guided by both heuristic information and pheromone trails.
    2. The choice of the next component depends on factors like the amount of pheromone on the edge and its desirability (e.g., distance, cost, etc.).
3. **Update Pheromones**:
    1. After all ants have constructed solutions, the amount of pheromone on each edge is updated based on the quality of the solutions found.
    2. Better solutions typically contribute more pheromone to the edges they contain.
4. **Evaporation**:
    1. Pheromone trails are subject to evaporation to prevent stagnation and encourage exploration.
    2. Over time, pheromone trails naturally decay, simulating the fading of scent in the environment.
5. **Termination Criterion**:
    1. ACO iterates through the construction-update cycle until a termination criterion is met, such as a maximum number of iterations or reaching a satisfactory solution.


SLIME
NETWORK

QUEUING THEORY

Steps:

1.Arrival Process: The arrival process describes how entities (e.g., vehicles) arrive at the service facility (e.g., intersection). It is often modeled as a Poisson process, where arrivals occur randomly over time according to a Poisson distribution.
2.Service Process: The service process describes how entities are served by one or more servers (e.g., traffic signals). It is often modeled as an exponential distribution, where service times follow an exponential distribution with a constant rate parameter.
3.Queueing Models: Queuing models are mathematical representations of the queuing system, which specify the characteristics of the arrival process, service process, number of servers, queue discipline (e.g., first-come-first-served or priority), and other relevant parameters.

Performance Metrics: Queuing theory provides several performance metrics to evaluate the performance of the queuing system, including utilization (the proportion of time servers are busy), average queue length, average waiting time, and system throughput.
Analytical Solutions: Queuing theory offers analytical solutions to determine key performance metrics and optimize system parameters. These solutions include Little's Law, Erlang's formulas, and various queuing models such as M/M/1, M/M/C, and M/G/1.

TRAFFIC QUEUE EXAMPLE

Background: Objective is to analyze the queuing behavior of vehicles waiting at traffic signals and determine optimal signal timings to minimize delays and improve overall traffic flow.

Traffic Configuration:

•The intersection experiences a steady flow of vehicles throughout the day, with an average arrival rate of 800 vehicles per hour.

•Vehicles arrive at the intersection following a Poisson distribution, representing random arrivals over time.

•Service times at the intersection follow an exponential distribution, with an average service rate of 900 vehicles per hour (corresponding to the capacity of the intersection to process vehicles).

1.Queuing Model:

We will use the M/M/1 queuing model, where:

1. M represents a Markovian arrival process (Poisson distribution).
2. M represents a Markovian service process (exponential distribution).
3. 1 represents a single server (traffic signal) controlling the flow of vehicles.

2.Key Metrics:

1. Utilization (ρ): The proportion of time the intersection is occupied by vehicles.
2. Average Number of Vehicles in the Queue (L): The average number of vehicles waiting at the intersection.
3. Average Waiting Time (W): The average time a vehicle spends waiting at the intersection before being able to proceed.

TRAFFIC QUEUE EXAMPLE

Utilization (ρ):

•Utilization is calculated as the ratio of the average arrival rate to the average service rate.

•Utilization (ρ) = λ / μ, where λ is the arrival rate and μ is the service rate.

•In this example, with λ = 800 vehicles/hour and μ = 900 vehicles/hour:

•We can simplify

λ = 800/60 = 13.33 vehicles per minute

μ = 900/60 = 15 vehicles per minute

•Utilization (ρ) = 13.33 / 15 = 0.89 (or 89%).

1.Average Waiting Time (W):

1. The average waiting time is calculated as the average time a vehicle spends waiting at the intersection.
2. W = 1 / (μ - λ), where μ is the service rate and λ is the arrival rate.
3. In this example, with μ = 15 vehicles/minute and λ = 13 vehicles/minute:
4. W = 1 / (15 – 13.33) = .6 minutes = 36 seconds.

1.Average Number of Vehicles in the Queue (L):

1. The average number of vehicles in the queue is calculated using Little's Law: L = λ * W, where L is the average number of vehicles in the queue, λ is the arrival rate, and W is the average waiting time.
2. In this example, with λ = 13.33 vehicles/minute and W =.6 minutes, so the average number of vehicles in the queue is:
3. L = 13.33 * .6 = 8 vehicles.

Future Advancements/ Challenges

•**Smart City Integration**: As cities become increasingly connected /"smart," network algorithms will likely evolve to integrate with various data sources such sensors, and real-time data streams. This integration can provide more accurate and dynamic information for optimizing traffic flow, resource allocation, and infrastructure planning.

•**Machine Learning and AI**: Advancements in machine learning and AI can enable network algorithms to adapt and learn from historical data, predict future traffic patterns, and optimize network designs.

Environmental Considerations: With growing concerns about climate change and environmental sustainability, future network algorithms may prioritize solutions that minimize carbon emissions, promote energy-efficient transportation modes, and incorporate green infrastructure into urban planning initiatives.

•**Multi-modal Transportation**: Future network algorithms may need to support multi-modal transportation systems, including not only cars but also public transit, bicycles, pedestrians, and emerging modes such as electric scooters and drones.

THANK YOU

QUESTIONS?