```
!pip install osmnx
```

```
Collecting osmnx
  Downloading osmnx-2.0.2-py3-none-any.whl.metadata (4.9 kB)
Requirement already satisfied: geopandas>=1.0 in
/usr/local/lib/python3.11/dist-packages (from osmnx) (1.0.1)
Requirement already satisfied: networkx>=2.5 in
/usr/local/lib/python3.11/dist-packages (from osmnx) (3.4.2)
Requirement already satisfied: numpy>=1.22 in
/usr/local/lib/python3.11/dist-packages (from osmnx) (2.0.2)
Requirement already satisfied: pandas>=1.4 in
/usr/local/lib/python3.11/dist-packages (from osmnx) (2.2.2)
Requirement already satisfied: requests>=2.27 in
/usr/local/lib/python3.11/dist-packages (from osmnx) (2.32.3)
Requirement already satisfied: shapely>=2.0 in
/usr/local/lib/python3.11/dist-packages (from osmnx) (2.1.0)
Requirement already satisfied: pyogrio>=0.7.2 in
/usr/local/lib/python3.11/dist-packages (from geopandas>=1.0->osmnx)
(0.10.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.11/dist-packages (from geopandas>=1.0->osmnx)
(24.2)
Requirement already satisfied: pyproj>=3.3.0 in
/usr/local/lib/python3.11/dist-packages (from geopandas>=1.0->osmnx)
(3.7.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas>=1.4->osmnx)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas>=1.4->osmnx)
(2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.11/dist-packages (from pandas>=1.4->osmnx)
(2025.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.27->osmnx)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.27->osmnx)
(3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.27->osmnx)
(2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.27->osmnx)
(2025.1.31)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2-
>pandas>=1.4->osmnx) (1.17.0)
Downloading osmnx-2.0.2-py3-none-any.whl (99 kB)
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.9/99.9 kB 3.3 MB/s eta
0:00:00
nx
Successfully installed osmnx-2.0.2
```

```python
# Importing all necessary libraries
import osmnx as ox
import networkx as nx
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import matplotlib.cm as cm
import random
import numpy as np
import pandas as pd
from collections import defaultdict

class RoadNetwork:
    """
    Represents a road network loaded from OpenStreetMap using OSMnx.
    """
    def __init__(self, address, distance=1000, network_type='drive'):
        """
        Initializes the RoadNetwork with data from OSMnx.

        Args:
            address (str): Address to center the network around.
            distance (int, optional): Radius (in meters) around the
address to load. Defaults to 1000.
            network_type (str, optional): Type of network to load
(e.g., 'drive', 'walk'). Defaults to 'drive'.

        Raises:
            ValueError: If network cannot be loaded for the given
address.
        """
        try:
            self.graph = ox.graph_from_address(address, dist=distance,
network_type=network_type)

            # Check if graph is empty
            if len(self.graph.nodes) == 0:
                raise ValueError("Loaded network contains no nodes")
            if len(self.graph.edges) == 0:
                raise ValueError("Loaded network contains no edges")

            self._add_travel_time_attribute()
        except Exception as e:
            raise ValueError(f"Failed to load network for address
'{address}': {str(e)}")
```

```
    def _add_travel_time_attribute(self):
        """
        Adds a 'travel_time' attribute to each edge based on speed
limit and length.
        Handles different speed formats and units.

        Formula: travel_time (minutes) = length (meters) / (speed
(km/h) * 1000/60)
        Where:
            - length is in meters
            - speed is in kilometers per hour
            - 1000/60 converts km/h to m/min
        """
        for u, v, k, data in self.graph.edges(data=True, keys=True):
            # Get maxspeed or default to 30 km/h
            speed = data.get("maxspeed", "30")

            # Handle list of speeds
            if isinstance(speed, list):
                speed = speed[0]

            # Clean and convert speed value
            speed_value = 30.0  # Default value

            if isinstance(speed, str):
                # Extract numeric part and unit
                speed_parts = speed.strip().split()
                speed_num = speed_parts[0]
                try:
                    speed_value = float(speed_num)

                    # Convert mph to km/h if necessary
                    if len(speed_parts) > 1 and "mph" in
speed_parts[1].lower():
                        speed_value *= 1.60934  # Convert mph to km/h
                except (ValueError, TypeError):
                    print(f"Warning: Could not parse speed value
'{speed}', using default of 30 km/h")
            elif isinstance(speed, (int, float)):
                speed_value = float(speed)

            # Calculate travel time (in minutes) with explicit unit
conversion
            length_m = data["length"]  # length in meters
            speed_kmh = speed_value    # speed in km/h
            meters_per_minute = (speed_kmh * 1000) / 60  # Convert
km/h to m/min
            data["travel_time"] = length_m / meters_per_minute  #
Result in minutes
```

```
            # Calculate road capacity based on road properties
            lanes = data.get("lanes", 1)
            if isinstance(lanes, list):
                lanes = lanes[0]
            try:
                lanes = float(lanes)
            except (ValueError, TypeError):
                lanes = 1.0

            road_type = data.get("highway", "residential")
            # Base capacity depends on road type
            if road_type in ["motorway", "trunk", "primary"]:
                base_capacity = 25  # Higher capacity for major roads
            elif road_type in ["secondary", "tertiary"]:
                base_capacity = 15  # Medium capacity
            else:
                base_capacity = 8   # Lower capacity for minor roads

            data["capacity"] = base_capacity * lanes

    def plot(self):
        """
        Plots the road network using OSMnx.

        Returns:
            tuple: Figure and Axes objects of the plot.
        """
        fig, ax = ox.plot_graph(self.graph, figsize=(10,10))
        return fig, ax

class Car:
    """
    Represents a car in the traffic simulation.
    """
    def __init__(self, start, destination, graph=None):
        """
        Initializes the Car with a start and destination node.

        Args:
            start (int): Node ID of the starting location.
            destination (int): Node ID of the destination location.
            graph (networkx.MultiDiGraph, optional): The road network
graph.

        Attributes:
            current_location (int): Current node ID where the car is
located.
            destination (int): Target node ID where the car is
heading.
            path (list): Ordered list of node IDs representing the
```

```
planned route,
                        excluding the current location (first node is
the next node to visit).
            graph (networkx.MultiDiGraph, optional): Reference to the
road network graph.
        """
        if graph is not None:
        # Validate nodes exist in graph
          if start not in graph:
              raise ValueError(f"Start node {start} does not exist in
the graph")
          if destination not in graph:
              raise ValueError(f"Destination node {destination} does
not exist in the graph")

        self.current_location = start
        self.destination = destination
        self.path = []
        self.graph = graph # Reference to the graph

        # Initialize path if graph is provided
        if graph is not None:
            self.calculate_path(graph)

    def calculate_path(self, graph):
        """
        Calculates the shortest path from current location to
destination.

        Args:
            graph (networkx.MultiDiGraph): The road network graph.

        Returns:
            bool: True if path was found, False otherwise.
        """
        try:
            import networkx as nx
            full_path = nx.shortest_path(graph, self.current_location,
self.destination, weight="travel_time")
            self.path = full_path[1:]  # Exclude current location
            return True
        except (nx.NetworkXNoPath, nx.NodeNotFound):
            return False

    def recalculate_path_with_congestion(self, graph, congestion_map,
congestion_threshold=0.7): # Corrected indentation
      """
      Recalculates path avoiding congested roads.
      """
      import networkx as nx
```

```
        # Create a copy of the graph with congestion weights
        G_temp = graph.copy()

        # Apply congestion penalties to edge weights
        for u, v, k, data in G_temp.edges(data=True, keys=True):
            congestion = congestion_map.get((u, v, k), 0)
            if congestion > congestion_threshold:
                # Penalize congested roads by increasing travel time
                congestion_penalty = 1 + (congestion * 5)  # Up to 6x
longer
                G_temp[u][v][k]["travel_time_adjusted"] =
data["travel_time"] * congestion_penalty
            else:
                G_temp[u][v][k]["travel_time_adjusted"] =
data["travel_time"]

        try:
            # Find path with adjusted weights
            return nx.shortest_path(G_temp, self.current_location,
self.destination,
                                   weight="travel_time_adjusted")
        except (nx.NetworkXNoPath, nx.NodeNotFound):
            # Fall back to original path if no alternative found
            return nx.shortest_path(graph, self.current_location,
self.destination,
                                   weight="travel_time")

class TrafficSimulator:
    """
    Simulates traffic flow on a road network.
    """
    def __init__(self, road_network, num_cars=100, start_hour = 8):
        """
        Initializes the TrafficSimulator with a road network and
number of cars.

        Args:
            road_network (RoadNetwork): The road network to simulate
on.
            num_cars (int, optional): Number of cars in the
simulation. Defaults to 100.
        """
        self.road_network = road_network
        self.num_cars = num_cars
        self.cars = self._create_cars()
        self.current_hour = start_hour
        self.minutes = 0
        self.minutes_per_step = 1 # Define minutes_per_step (I decided
to set at 1 as it gave interesting results)
```

```
        self.congestion_map = {}

    def _create_cars(self):
      """
      Creates cars with random starting and destination locations.
      Initializes their paths using the road network.
      """
      nodes = list(self.road_network.graph.nodes())
      cars = []

      for _ in range(self.num_cars):
          start = random.choice(nodes)
          destination = random.choice([n for n in nodes if n !=
start])
          car = Car(start, destination, self.road_network.graph)
        # Ensure car has a valid path
          if not car.path and car.current_location != car.destination:
              try:
                  car.path = nx.shortest_path(self.road_network.graph,
                                              car.current_location,
                                              car.destination,
                                              weight="travel_time")
[1:]  # Exclude current location
              except nx.NetworkXNoPath:
                  # Try a different destination
                  continue
          cars.append(car)

      return cars

    def move_cars(self):
      """
      Moves cars in the simulation based on their paths and traffic
conditions.
      """
    # Update congestion map periodically
      if not hasattr(self, 'step_counter'):
          self.step_counter = 0
      self.step_counter += 1

      if self.step_counter % 5 == 0:
          self.update_congestion_map()

      # Track if any car moved for debugging
      any_car_moved = False

      for car in self.cars:
        # Check if car has reached its destination
        if car.current_location == car.destination:
            # Generate a new destination
```

```
            nodes = list(self.road_network.graph.nodes())
            car.destination = random.choice([node for node in nodes if
node != car.current_location])
            try:
                car.path = nx.shortest_path(self.road_network.graph,
                                          car.current_location,
                                          car.destination,
                                          weight="travel_time")[1:]  #
Exclude current location
            except nx.NetworkXNoPath:
                continue  # Skip this car if no path found
           # print(f"Car reached destination, new path length:
{len(car.path)}")
            continue

        # Skip cars with empty paths
        if not car.path:
            # Try to recalculate path
            try:
                car.path = nx.shortest_path(self.road_network.graph,
                                          car.current_location,
                                          car.destination,
                                          weight="travel_time")[1:]
                #print(f"Recalculated path, new length:
{len(car.path)}")
            except nx.NetworkXNoPath:
                # If no path, assign new destination
                nodes = list(self.road_network.graph.nodes())
                car.destination = random.choice([node for node in
nodes if node != car.current_location])
                try:
                    car.path =
nx.shortest_path(self.road_network.graph,
                                              car.current_location,
                                              car.destination,
                                              weight="travel_time")
[1:]
                   # print(f"New destination, path length:
{len(car.path)}")
                except nx.NetworkXNoPath:
                    continue

            if not car.path:
                continue

        next_node = car.path[0]

        # Check if the edge exists (in case the graph changed)
        if next_node not in
self.road_network.graph[car.current_location]:
```

```
            # Path is invalid, recalculate
            try:
                car.path = nx.shortest_path(self.road_network.graph,
                                          car.current_location,
                                          car.destination,
                                          weight="travel_time")[1:]
            except nx.NetworkXNoPath:
                continue
            if not car.path:
                continue
            next_node = car.path[0]

        # Find the edge data
        edge_data = None
        for k in self.road_network.graph[car.current_location]
[next_node]:
            edge_data = self.road_network.graph[car.current_location]
[next_node][k]
            break

        # Count cars on edge
        cars_on_edge = sum(1 for c in self.cars if c.current_location
== car.current_location and c.path and c.path[0] == next_node)

        # Calculate movement probability based on congestion
        capacity = edge_data.get('capacity', 8)

        # Apply time factors
        time_factor = 1.0
        if 7 <= self.current_hour <= 9 or 16 <= self.current_hour <=
18: # RUsh hour
            time_factor = 0.6

        effective_capacity = capacity * time_factor
        congestion_factor = min(1.0, cars_on_edge /
effective_capacity)
        speed_reduction = 1.0 - (congestion_factor * 0.8)

        # ONLY ONE movement decision per car per step
        if random.random() < speed_reduction:
            car.current_location = next_node
            car.path.pop(0)
            any_car_moved = True

        # Dynamic rerouting (separate from movement)
        reroute_probability = 0.05
        if congestion_factor > 0.7:
            reroute_probability = 0.3

        if random.random() < reroute_probability and hasattr(car,
```

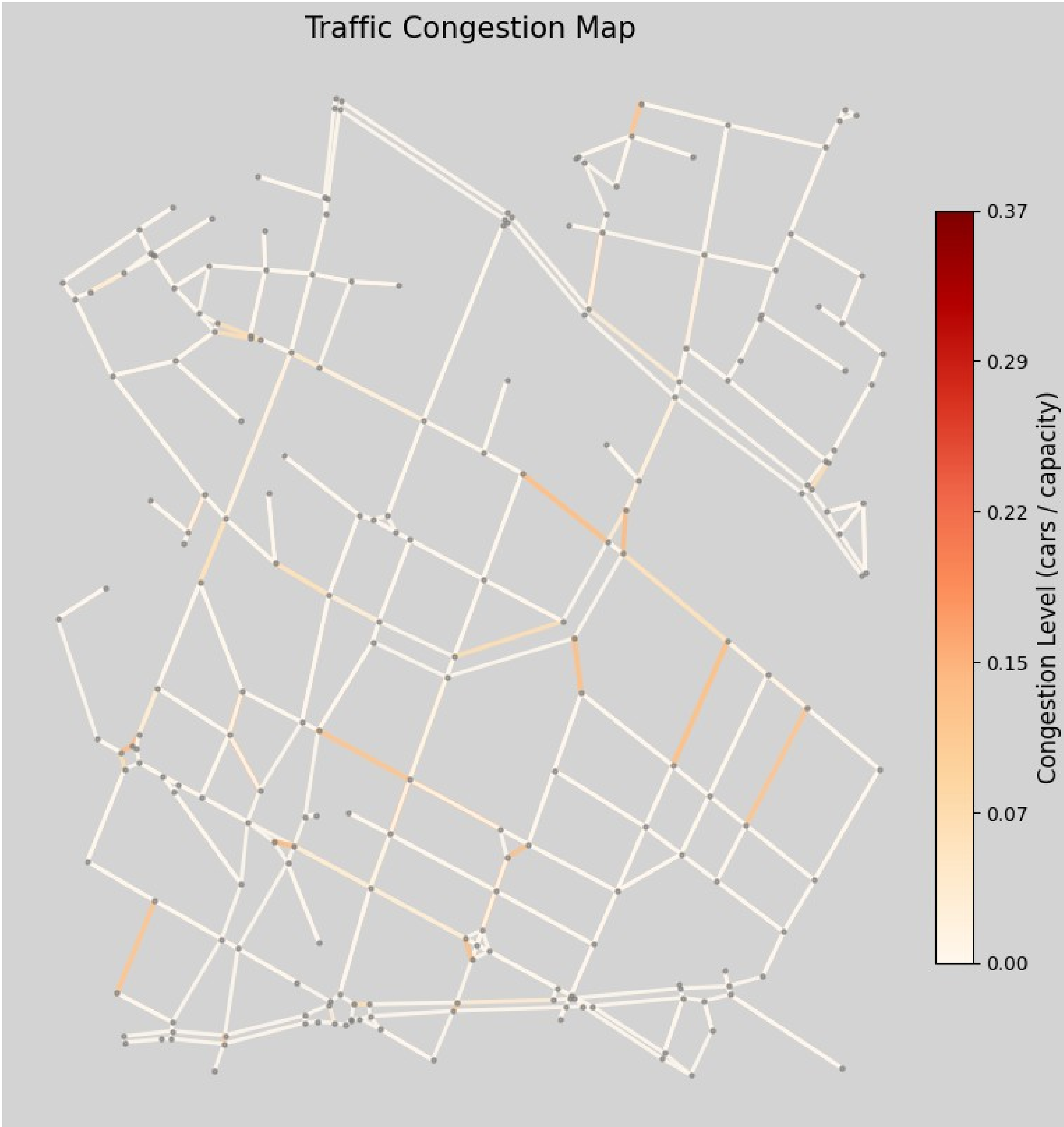
```
'recalculate_path_with_congestion'):
            try:
                new_path = car.recalculate_path_with_congestion(
                    self.road_network.graph,
                    self.congestion_map
                )
                if new_path and len(new_path) > 1:
                    car.path = new_path[1:]
            except Exception as e:
                print(f"Error in rerouting: {e}")

      if not any_car_moved:
        print("WARNING: No cars moved this step!")

    def simulate(self, num_steps=10):
        """
        Runs the traffic simulation for a given number of steps.

        Args:
            num_steps (int, optional): Number of simulation steps.
Defaults to 10.
        """
        for step in range(num_steps):
          # Update simulation time
          self.minutes += self.minutes_per_step
          if self.minutes >= 60:
            self.current_hour += (self.current_hour + 1) % 24
            self.minutes %= 60

          # Move cars - this MUST be properly indented to run every
step
          self.move_cars()
          #print(f"Step {step + 1} - Time: {self.current_hour:02d}:
{self.minutes:02d}")

          # Check if anything is happening
          car_moved = False
          for car in self.cars:
              if car.path:
                  car_moved = True
                  break

          if not car_moved:
              print("WARNING: No cars are moving in the simulation!")

            # For demonstration, print the number of cars at each node
          """
          node_counts = {node: 0 for node in
self.road_network.graph.nodes()}
          for car in self.cars:
```

```
                node_counts[car.current_location] += 1
            print(f"Step {step + 1}:")
            for node, count in node_counts.items():
                print(f"Node {node}: {count} cars")
            print()
            """

    def plot_traffic(self):
        """
        Improved traffic visualization using congestion (cars /
capacity).
        Adds colorbar and better styling.
        """
        # Update congestion map before plotting
        self.update_congestion_map()
        G = self.road_network.graph

        # Get node positions for layout
        node_positions = {node: (data['x'], data['y']) for node, data
in G.nodes(data=True)}

        # Boost low congestion values
        congestion_boost = {k: v**0.5 for k, v in
self.congestion_map.items()}  # gamma correction
        max_congestion = max(congestion_boost.values()) if
congestion_boost else 1
        norm = mcolors.Normalize(vmin=0,
vmax=max(congestion_boost.values()))
        cmap = cm.get_cmap("OrRd")

        fig, ax = plt.subplots(figsize=(10, 10))
        fig.patch.set_facecolor('lightgray')     # Set figure
background
        ax.set_facecolor('lightgray')            # Set plot (axes)
background

        for (u, v, k), congestion in self.congestion_map.items():
            color = cmap(norm(congestion))
            width = 2 + 4 * congestion  # line thickness increases
with congestion

            x_vals = [node_positions[u][0], node_positions[v][0]]
            y_vals = [node_positions[u][1], node_positions[v][1]]
            ax.plot(x_vals, y_vals, color=color, linewidth=width,
alpha=0.9)

        # Draw nodes
        nx.draw_networkx_nodes(G, node_positions, ax=ax, node_size=5,
node_color="gray", alpha = 0.7)
```

```
        # Colorbar legend
        sm = plt.cm.ScalarMappable(cmap=cmap, norm=norm)
        sm.set_array([])
        cbar = plt.colorbar(sm, ax=ax, shrink=0.7, pad=0.01)
        cbar.set_label("Congestion Level (cars / capacity)",
fontsize=12)
        # Add ticks with readable values
        cbar_ticks = np.linspace(0, max(congestion_boost.values()),
num=6)
        cbar.set_ticks(cbar_ticks)
        cbar.set_ticklabels([f"{tick:.2f}" for tick in cbar_ticks])
        cbar.ax.yaxis.set_tick_params(color='black',
labelcolor='black')
        plt.setp(plt.getp(cbar.ax.axes, 'yticklabels'), color='black')

        ax.set_title("Traffic Congestion Map", fontsize=15,
color="black")
        ax.tick_params(left=False, bottom=False, labelleft=False,
labelbottom=False, colors = 'white')
        plt.axis("off")
        plt.show()

        return fig, ax

    def _edge_car_counts(self):
        """
        Calculates the number of cars on each edge of the road
network.

        Returns:
            dict: A dictionary mapping edge keys (u, v, k) to car
counts.
        """
        counts = {}
        for u, v, k in self.road_network.graph.edges(keys=True):
            counts[(u, v, k)] = 0 # Using u, v, k as key

        for car in self.cars:
            if car.path:
                next_node = car.path[0]
                # Find the correct edge key
                for k in self.road_network.graph[car.current_location]
[next_node].keys():
                    counts[(car.current_location, next_node, k)] += 1
                    break  # Assuming we only want to count one edge
between the nodes
        return counts

    def update_congestion_map(self):
      """
```

```python
        Creates a map of congested edges in the network.
        """
        self.congestion_map = {}

    # Calculate congestion for each edge
        for u, v, k, data in self.road_network.graph.edges(data=True,
keys=True):
            cars_on_edge = sum(1 for car in self.cars
                              if car.current_location == u
                              and car.path
                              and car.path[0] == v)
            capacity = data.get('capacity', 8) # Get capacity from edge
data
            congestion_level = cars_on_edge / capacity
            self.congestion_map[(u, v, k)] = congestion_level

berlin_network = RoadNetwork('Adalbertstraße 58, Berlin, Germany')
simulator = TrafficSimulator(berlin_network)
simulator.simulate()
simulator.plot_traffic()
```

```
<ipython-input-6-5b8c658043ff>:220: MatplotlibDeprecationWarning: The
get_cmap function was deprecated in Matplotlib 3.7 and will be removed
in 3.11. Use ``matplotlib.colormaps[name]`` or
``matplotlib.colormaps.get_cmap()`` or ``pyplot.get_cmap()`` instead.
  cmap = cm.get_cmap("OrRd")
```




```
(<Figure size 1000x1000 with 2 Axes>,
 <Axes: title={'center': 'Traffic Congestion Map'}>)

## Simulating Berlin network for 10 times
# a. Empirical Analysis
num_simulations = 10  # Number of simulations to run
congestion_data = []  # Store congestion levels for each simulation

for _ in range(num_simulations):
    simulator = TrafficSimulator(berlin_network)
```

```
    simulator.simulate(num_steps=100) # Run simulation for 100 steps
in each loop
    congestion_data.append(simulator.congestion_map)

# Analyze congestion_data to identify consistently congested edges
# For example, calculate average congestion for each edge:
edge_congestion_avg = {}
for edge in berlin_network.graph.edges(keys=True):
    edge_congestion_avg[edge] = np.mean([d.get(edge, 0) for d in
congestion_data])

# Print the most congested edges
# This will show which edges are most congested across simulations
import pprint
pprint.pprint(edge_congestion_avg)

simulator.plot_traffic()
```

```
{(21487224, 29215073, 0): np.float64(0.006666666666666666),
 (21487224, 196725581, 0): np.float64(0.0033333333333333333),
 (21487230, 26960762, 0): np.float64(0.003333333333333333),
 (21487230, 28794539, 0): np.float64(0.0),
 (21487230, 196724115, 0): np.float64(0.006666666666666666),
 (21487230, 1491865195, 0): np.float64(0.0),
 (21487231, 29215047, 0): np.float64(0.013333333333333332),
 (21487231, 5984003368, 0): np.float64(0.0),
 (21487232, 26960762, 0): np.float64(0.006666666666666666),
 (21487232, 29218293, 0): np.float64(0.02),
 (21487232, 29218315, 0): np.float64(0.01),
 (21487232, 269476782, 0): np.float64(0.0044444444444444444),
 (26517409, 287447548, 0): np.float64(0.0022222222222222222),
 (26960758, 29271261, 0): np.float64(0.006666666666666666),
 (26960761, 28794542, 0): np.float64(0.019999999999999997),
 (26960761, 29218316, 0): np.float64(0.0125),
 (26960761, 2625899861, 0): np.float64(0.02),
 (26960761, 3411530669, 0): np.float64(0.006666666666666666),
 (26960762, 21487230, 0): np.float64(0.006666666666666666),
 (26960762, 21487232, 0): np.float64(0.015555555555555555),
 (26960762, 28373656, 0): np.float64(0.02),
 (26960762, 28794542, 0): np.float64(0.017777777777777778),
 (27555211, 27555221, 0): np.float64(0.0),
 (27555211, 173953089, 0): np.float64(0.0),
 (27555214, 27555215, 0): np.float64(0.0),
 (27555215, 27555227, 0): np.float64(0.006666666666666666),
 (27555221, 29276746, 0): np.float64(0.0),
 (27555221, 29276959, 0): np.float64(0.003333333333333333),
 (27555224, 27555225, 0): np.float64(0.01),
 (27555224, 27555227, 0): np.float64(0.0),
 (27555224, 318549151, 0): np.float64(0.0),
 (27555225, 27555226, 0): np.float64(0.025),
```

```
 (27555225, 122087860, 0): np.float64(0.0),
 (27555226, 27555215, 0): np.float64(0.0125),
 (27555226, 27555225, 0): np.float64(0.0),
 (27555227, 27555224, 0): np.float64(0.0),
 (27555227, 29276961, 0): np.float64(0.0),
 (28373648, 29273079, 0): np.float64(0.0022222222222222222),
 (28373648, 10254148068, 0): np.float64(0.008888888888888889),
 (28373656, 26960762, 0): np.float64(0.008888888888888889),
 (28373656, 3352479275, 0): np.float64(0.02),
 (28794517, 28794518, 0): np.float64(0.0375),
 (28794517, 29215060, 0): np.float64(0.0),
 (28794517, 848523542, 0): np.float64(0.0),
 (28794518, 28794517, 0): np.float64(0.0125),
 (28794518, 3352479286, 0): np.float64(0.006666666666666666),
 (28794518, 3463840860, 0): np.float64(0.0),
 (28794519, 338894718, 0): np.float64(0.0),
 (28794519, 611862495, 0): np.float64(0.003333333333333333),
 (28794519, 3352479290, 0): np.float64(0.003333333333333333),
 (28794539, 21487230, 0): np.float64(0.0),
 (28794539, 28794542, 0): np.float64(0.0),
 (28794539, 612417324, 0): np.float64(0.0125),
 (28794542, 26960761, 0): np.float64(0.008888888888888889),
 (28794542, 26960762, 0): np.float64(0.015555555555555555),
 (29215046, 29215049, 0): np.float64(0.013333333333333332),
 (29215046, 287447547, 0): np.float64(0.04666666666666667),
 (29215047, 21487231, 0): np.float64(0.026666666666666665),
 (29215047, 287447549, 0): np.float64(0.006666666666666666),
 (29215049, 29215046, 0): np.float64(0.013333333333333332),
 (29215049, 29218289, 0): np.float64(0.0),
 (29215049, 29218293, 0): np.float64(0.026666666666666665),
 (29215057, 29215058, 0): np.float64(0.006666666666666666),
 (29215057, 29215067, 0): np.float64(0.025),
 (29215057, 269476782, 0): np.float64(0.0),
 (29215058, 29215057, 0): np.float64(0.006666666666666666),
 (29215058, 29215060, 0): np.float64(0.0),
 (29215058, 262484667, 0): np.float64(0.0),
 (29215060, 28794517, 0): np.float64(0.0),
 (29215060, 29215058, 0): np.float64(0.0),
 (29215063, 29215067, 0): np.float64(0.0),
 (29215067, 29215057, 0): np.float64(0.0),
 (29215067, 29215063, 0): np.float64(0.0),
 (29215067, 292682410, 0): np.float64(0.0125),
 (29215071, 29215073, 0): np.float64(0.0),
 (29215073, 29215071, 0): np.float64(0.0),
 (29215073, 196724115, 0): np.float64(0.006666666666666666),
 (29215073, 660778774, 0): np.float64(0.0),
 (29216571, 29216572, 0): np.float64(0.003333333333333333),
 (29216571, 29216575, 0): np.float64(0.0),
 (29216572, 1969136398, 0): np.float64(0.006666666666666666),
```

```
(29216575, 29216571, 0): np.float64(0.0),
(29217269, 5984003368, 0): np.float64(0.0),
(29217269, 5984003376, 0): np.float64(0.025),
(29217276, 287447549, 0): np.float64(0.02),
(29217276, 1285251472, 0): np.float64(0.02),
(29217276, 1285251502, 0): np.float64(0.0),
(29217277, 1285251502, 0): np.float64(0.0),
(29217277, 5984003369, 0): np.float64(0.0),
(29217280, 29217277, 0): np.float64(0.0),
(29217280, 29218294, 0): np.float64(0.03333333333333333),
(29217280, 283251475, 0): np.float64(0.02),
(29217293, 29217280, 0): np.float64(0.04666666666666667),
(29217293, 29217317, 0): np.float64(0.013333333333333332),
(29217293, 29217348, 0): np.float64(0.0375),
(29217303, 29218288, 0): np.float64(0.0),
(29217317, 29217293, 0): np.float64(0.016666666666666667),
(29217317, 29217322, 0): np.float64(0.013333333333333332),
(29217317, 29218327, 0): np.float64(0.02),
(29217317, 88071069, 0): np.float64(0.0033333333333333333),
(29217319, 164641499, 0): np.float64(0.0),
(29217321, 29217332, 0): np.float64(0.015555555555555555),
(29217321, 267214998, 0): np.float64(0.025),
(29217322, 29217317, 0): np.float64(0.006666666666666666),
(29217322, 29217321, 0): np.float64(0.015555555555555555),
(29217325, 29217322, 0): np.float64(0.015555555555555555),
(29217327, 29217325, 0): np.float64(0.01),
(29217327, 442649842, 0): np.float64(0.013333333333333332),
(29217332, 29217340, 0): np.float64(0.013333333333333332),
(29217332, 8078047384, 0): np.float64(0.0),
(29217340, 268466879, 0): np.float64(0.015555555555555555),
(29217342, 29217280, 0): np.float64(0.0),
(29217342, 29218289, 0): np.float64(0.0),
(29217348, 29218296, 0): np.float64(0.0125),
(29217348, 359541354, 0): np.float64(0.0125),
(29218287, 29218288, 0): np.float64(0.0),
(29218287, 29788955, 0): np.float64(0.0),
(29218287, 5984003377, 0): np.float64(0.0033333333333333333),
(29218288, 29218287, 0): np.float64(0.0),
(29218288, 442649846, 0): np.float64(0.006666666666666666),
(29218289, 29215049, 0): np.float64(0.0),
(29218289, 29217342, 0): np.float64(0.0),
(29218289, 29218291, 0): np.float64(0.025),
(29218289, 29218294, 0): np.float64(0.0),
(29218291, 29218289, 0): np.float64(0.0),
(29218291, 29218293, 0): np.float64(0.0),
(29218291, 29218295, 0): np.float64(0.0),
(29218293, 21487232, 0): np.float64(0.023333333333333333),
(29218293, 29215049, 0): np.float64(0.023333333333333334),
(29218293, 29218291, 0): np.float64(0.0),
```

```
 (29218294, 29217280, 0): np.float64(0.013333333333333332),
 (29218294, 29218289, 0): np.float64(0.0125),
 (29218294, 1285251472, 0): np.float64(0.006666666666666666),
 (29218295, 29217342, 0): np.float64(0.0125),
 (29218295, 29218291, 0): np.float64(0.0),
 (29218295, 29218296, 0): np.float64(0.025),
 (29218295, 29218299, 0): np.float64(0.0),
 (29218296, 29218295, 0): np.float64(0.0),
 (29218296, 29218325, 0): np.float64(0.0),
 (29218296, 29218326, 0): np.float64(0.025),
 (29218299, 29218295, 0): np.float64(0.0),
 (29218299, 29218302, 0): np.float64(0.013333333333333332),
 (29218299, 29218315, 0): np.float64(0.013333333333333332),
 (29218299, 5450916417, 0): np.float64(0.0125),
 (29218302, 29218299, 0): np.float64(0.013333333333333332),
 (29218302, 29218305, 0): np.float64(0.0),
 (29218302, 29218325, 0): np.float64(0.04),
 (29218305, 29218302, 0): np.float64(0.0125),
 (29218305, 29218318, 0): np.float64(0.0),
 (29218305, 29218322, 0): np.float64(0.0),
 (29218313, 29218315, 0): np.float64(0.0125),
 (29218315, 21487232, 0): np.float64(0.01),
 (29218315, 29218299, 0): np.float64(0.026666666666666665),
 (29218315, 29218313, 0): np.float64(0.0),
 (29218316, 26960761, 0): np.float64(0.0125),
 (29218316, 29218319, 0): np.float64(0.0),
 (29218318, 29218305, 0): np.float64(0.0),
 (29218318, 29218316, 0): np.float64(0.0),
 (29218318, 29218319, 0): np.float64(0.0),
 (29218319, 29218318, 0): np.float64(0.0),
 (29218319, 5450916417, 0): np.float64(0.0),
 (29218321, 29218322, 0): np.float64(0.0125),
 (29218321, 29276217, 0): np.float64(0.01),
 (29218321, 2625899861, 0): np.float64(0.02),
 (29218322, 29218305, 0): np.float64(0.0),
 (29218322, 29218321, 0): np.float64(0.0),
 (29218322, 29218323, 0): np.float64(0.0),
 (29218322, 29276222, 0): np.float64(0.0),
 (29218323, 29218302, 0): np.float64(0.02),
 (29218323, 29218322, 0): np.float64(0.0),
 (29218323, 29218324, 0): np.float64(0.03333333333333333),
 (29218324, 29218323, 0): np.float64(0.006666666666666666),
 (29218324, 29218326, 0): np.float64(0.04666666666666667),
 (29218324, 29276211, 0): np.float64(0.03333333333333333),
 (29218325, 29218296, 0): np.float64(0.0),
 (29218325, 29218302, 0): np.float64(0.0),
 (29218325, 29218324, 0): np.float64(0.04),
 (29218326, 29218296, 0): np.float64(0.0),
 (29218326, 29218324, 0): np.float64(0.05333333333333333),
```

```
(29218326, 29218327, 0): np.float64(0.04),
(29218326, 2143522428, 0): np.float64(0.0125),
(29218327, 29217317, 0): np.float64(0.026666666666666665),
(29218327, 29218326, 0): np.float64(0.03333333333333333),
(29218327, 29218328, 0): np.float64(0.0),
(29218327, 29276743, 0): np.float64(0.0),
(29218328, 29218327, 0): np.float64(0.0),
(29218328, 29218328, 0): np.float64(0.0),
(29218328, 29218328, 1): np.float64(0.0),
(29219437, 21487224, 0): np.float64(0.01),
(29219438, 29219437, 0): np.float64(0.006666666666666666),
(29271261, 29219438, 0): np.float64(0.0022222222222222222),
(29271261, 29273080, 0): np.float64(0.0),
(29273048, 29273049, 0): np.float64(0.0125),
(29273048, 29273057, 0): np.float64(0.0033333333333333333),
(29273048, 29273071, 0): np.float64(0.0033333333333333333),
(29273049, 29273048, 0): np.float64(0.0625),
(29273049, 271872510, 0): np.float64(0.0),
(29273050, 29273056, 0): np.float64(0.0),
(29273050, 9500676798, 0): np.float64(0.025),
(29273056, 29273050, 0): np.float64(0.0375),
(29273056, 29273058, 0): np.float64(0.0125),
(29273056, 271872441, 0): np.float64(0.0125),
(29273057, 29273048, 0): np.float64(0.005),
(29273057, 29273056, 0): np.float64(0.0),
(29273057, 29273068, 0): np.float64(0.0375),
(29273057, 29273069, 0): np.float64(0.011111111111111112),
(29273058, 26960758, 0): np.float64(0.0033333333333333333),
(29273058, 29273056, 0): np.float64(0.025),
(29273058, 1984006630, 0): np.float64(0.0),
(29273064, 29273065, 0): np.float64(0.008333333333333333),
(29273064, 29276209, 0): np.float64(0.011111111111111112),
(29273064, 3242035262, 0): np.float64(0.006666666666666666),
(29273065, 29273058, 0): np.float64(0.013333333333333332),
(29273065, 29273064, 0): np.float64(0.02),
(29273065, 29273069, 0): np.float64(0.011111111111111112),
(29273067, 2476048346, 0): np.float64(0.0125),
(29273067, 2479455794, 0): np.float64(0.0),
(29273068, 29273057, 0): np.float64(0.0125),
(29273068, 29273076, 0): np.float64(0.0125),
(29273068, 2476048346, 0): np.float64(0.0),
(29273069, 29273057, 0): np.float64(0.006666666666666666),
(29273069, 29273065, 0): np.float64(0.01111111111111111),
(29273069, 2814705317, 0): np.float64(0.0375),
(29273071, 29273048, 0): np.float64(0.0033333333333333333),
(29273071, 29273076, 0): np.float64(0.0),
(29273071, 2757332333, 0): np.float64(0.0033333333333333333),
(29273075, 29273076, 0): np.float64(0.0),
(29273075, 29273077, 0): np.float64(0.025),
```

```
(29273076, 29273068, 0): np.float64(0.0125),
(29273076, 29273071, 0): np.float64(0.0),
(29273076, 29273075, 0): np.float64(0.0),
(29273077, 29273075, 0): np.float64(0.0),
(29273077, 29273078, 0): np.float64(0.0),
(29273077, 29273079, 0): np.float64(0.0125),
(29273078, 29273077, 0): np.float64(0.0),
(29273079, 28373648, 0): np.float64(0.0),
(29273079, 29273077, 0): np.float64(0.0),
(29273080, 3411530669, 0): np.float64(0.013333333333333332),
(29273080, 10312836355, 0): np.float64(0.006666666666666666),
(29275853, 29273065, 0): np.float64(0.015555555555555555),
(29275853, 3242035262, 0): np.float64(0.0),
(29275983, 29785137, 0): np.float64(0.0),
(29275983, 60237352, 0): np.float64(0.0022222222222222222),
(29276209, 29273064, 0): np.float64(0.008888888888888889),
(29276209, 29276210, 0): np.float64(0.0),
(29276209, 29276213, 0): np.float64(0.05000000000000001),
(29276210, 29276209, 0): np.float64(0.0),
(29276211, 29276726, 0): np.float64(0.0125),
(29276211, 3243910627, 0): np.float64(0.05333333333333333),
(29276213, 29276209, 0): np.float64(0.016666666666666666),
(29276213, 29276217, 0): np.float64(0.06333333333333332),
(29276217, 29218321, 0): np.float64(0.023333333333333334),
(29276217, 29276222, 0): np.float64(0.04666666666666666),
(29276217, 3243910627, 0): np.float64(0.02),
(29276222, 29218322, 0): np.float64(0.0125),
(29276222, 29218323, 0): np.float64(0.026666666666666665),
(29276687, 29276688, 0): np.float64(0.003333333333333333),
(29276687, 29276755, 0): np.float64(0.05),
(29276687, 3243910627, 0): np.float64(0.013333333333333332),
(29276688, 29276687, 0): np.float64(0.006666666666666666),
(29276688, 29276689, 0): np.float64(0.016666666666666666),
(29276688, 29276757, 0): np.float64(0.0),
(29276689, 29276688, 0): np.float64(0.006666666666666666),
(29276689, 29276690, 0): np.float64(0.003333333333333333),
(29276689, 29276762, 0): np.float64(0.0),
(29276690, 29276689, 0): np.float64(0.003333333333333333),
(29276690, 29276778, 0): np.float64(0.025),
(29276726, 29276211, 0): np.float64(0.0125),
(29276726, 29276728, 0): np.float64(0.0),
(29276726, 29276755, 0): np.float64(0.0125),
(29276728, 29276726, 0): np.float64(0.0375),
(29276728, 29276730, 0): np.float64(0.0125),
(29276728, 29276753, 0): np.float64(0.0),
(29276730, 29276728, 0): np.float64(0.025),
(29276730, 29276752, 0): np.float64(0.0),
(29276730, 2143522428, 0): np.float64(0.025),
(29276737, 29276730, 0): np.float64(0.0),
```

```
(29276737, 29276743, 0): np.float64(0.0375),
(29276743, 29218327, 0): np.float64(0.0375),
(29276743, 29276737, 0): np.float64(0.0),
(29276743, 88071065, 0): np.float64(0.025),
(29276743, 11391147437, 0): np.float64(0.0),
(29276745, 88070313, 0): np.float64(0.0125),
(29276745, 88071064, 0): np.float64(0.006666666666666666),
(29276745, 88071065, 0): np.float64(0.025),
(29276745, 88071069, 0): np.float64(0.003333333333333333),
(29276746, 27555221, 0): np.float64(0.0125),
(29276746, 2677649274, 0): np.float64(0.013333333333333332),
(29276750, 318545593, 0): np.float64(0.01),
(29276750, 536286819, 0): np.float64(0.01),
(29276750, 3900230865, 0): np.float64(0.006666666666666666),
(29276752, 29276730, 0): np.float64(0.0),
(29276752, 29276753, 0): np.float64(0.0),
(29276752, 29276759, 0): np.float64(0.0125),
(29276752, 11391147437, 0): np.float64(0.0375),
(29276753, 29276728, 0): np.float64(0.0),
(29276753, 29276752, 0): np.float64(0.0625),
(29276753, 29276755, 0): np.float64(0.075),
(29276753, 29276759, 0): np.float64(0.0),
(29276755, 29276687, 0): np.float64(0.025),
(29276755, 29276726, 0): np.float64(0.0125),
(29276755, 29276753, 0): np.float64(0.05),
(29276755, 29276757, 0): np.float64(0.0),
(29276757, 29276688, 0): np.float64(0.0),
(29276757, 29276755, 0): np.float64(0.0),
(29276757, 29276759, 0): np.float64(0.0),
(29276757, 29276762, 0): np.float64(0.0),
(29276759, 29276752, 0): np.float64(0.0),
(29276759, 29276753, 0): np.float64(0.0),
(29276759, 29276757, 0): np.float64(0.0),
(29276759, 29276764, 0): np.float64(0.0125),
(29276762, 29276689, 0): np.float64(0.0),
(29276762, 29276757, 0): np.float64(0.0),
(29276762, 29276764, 0): np.float64(0.0),
(29276762, 29276778, 0): np.float64(0.0),
(29276764, 29276759, 0): np.float64(0.0125),
(29276764, 29276762, 0): np.float64(0.0),
(29276764, 29276777, 0): np.float64(0.0),
(29276773, 32248754, 0): np.float64(0.0),
(29276773, 122087860, 0): np.float64(0.0),
(29276773, 318549151, 0): np.float64(0.0),
(29276775, 29276773, 0): np.float64(0.006666666666666666),
(29276775, 29276777, 0): np.float64(0.0),
(29276777, 29276764, 0): np.float64(0.0),
(29276777, 29276775, 0): np.float64(0.0375),
(29276777, 29276778, 0): np.float64(0.0),
```

```
(29276778, 29276690, 0): np.float64(0.0),
(29276778, 29276762, 0): np.float64(0.0),
(29276778, 29276777, 0): np.float64(0.05),
(29276959, 318545593, 0): np.float64(0.0),
(29276959, 318545595, 0): np.float64(0.0125),
(29276959, 3900230865, 0): np.float64(0.0),
(29276961, 27555224, 0): np.float64(0.006666666666666666),
(29276961, 29276750, 0): np.float64(0.003333333333333333),
(29785137, 29275983, 0): np.float64(0.0),
(29785137, 29785151, 0): np.float64(0.0),
(29785137, 60237352, 0): np.float64(0.0),
(29785151, 29275983, 0): np.float64(0.01),
(29785151, 29785137, 0): np.float64(0.0),
(29785151, 2028472686, 0): np.float64(0.0),
(29787489, 122087860, 0): np.float64(0.0),
(29788946, 29217340, 0): np.float64(0.003333333333333333),
(29788955, 164641499, 0): np.float64(0.025),
(29788955, 267214998, 0): np.float64(0.0),
(29788955, 5984003369, 0): np.float64(0.0125),
(32248754, 29276773, 0): np.float64(0.0),
(60237352, 29785137, 0): np.float64(0.0),
(60237352, 3242035261, 0): np.float64(0.0022222222222222222),
(88070313, 29276745, 0): np.float64(0.0),
(88070313, 29276746, 0): np.float64(0.025),
(88070313, 88071064, 0): np.float64(0.025),
(88071064, 29276745, 0): np.float64(0.0),
(88071064, 88071065, 0): np.float64(0.025),
(88071064, 318545591, 0): np.float64(0.003333333333333333),
(88071065, 29276743, 0): np.float64(0.025),
(88071065, 29276745, 0): np.float64(0.0),
(88071065, 88071069, 0): np.float64(0.0),
(88071069, 29217317, 0): np.float64(0.003333333333333333),
(88071069, 29276745, 0): np.float64(0.0),
(88071069, 88070313, 0): np.float64(0.025),
(122087860, 29276773, 0): np.float64(0.006666666666666666),
(122087860, 29787489, 0): np.float64(0.0),
(164641499, 29217293, 0): np.float64(0.025),
(164641499, 29217319, 0): np.float64(0.0),
(173953089, 27555211, 0): np.float64(0.025),
(173953089, 1539712867, 0): np.float64(0.0125),
(196724115, 21487230, 0): np.float64(0.003333333333333333),
(196724115, 660778774, 0): np.float64(0.006666666666666666),
(196725581, 29219438, 0): np.float64(0.0),
(196725581, 29273080, 0): np.float64(0.0016666666666666666),
(196774436, 9500676798, 0): np.float64(0.0),
(252778554, 2476048346, 0): np.float64(0.025),
(262484667, 29215058, 0): np.float64(0.0),
(262484667, 3352479286, 0): np.float64(0.0),
(267214998, 29788955, 0): np.float64(0.0),
```

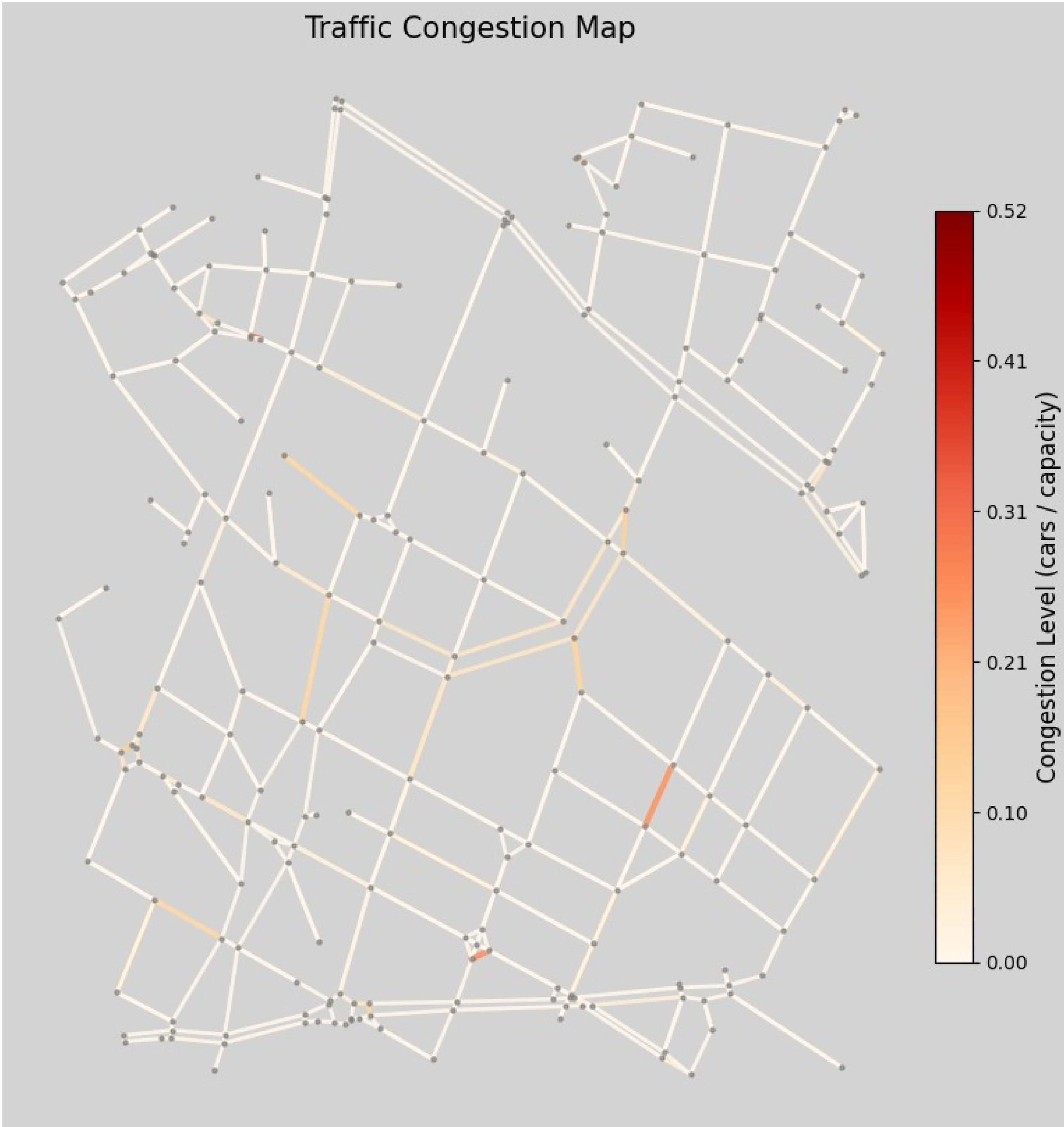
```
(268466879, 11806567841, 0): np.float64(0.017777777777777778),
(269476781, 269476859, 0): np.float64(0.0),
(269476782, 21487232, 0): np.float64(0.0022222222222222222),
(269476782, 29215057, 0): np.float64(0.006666666666666666),
(269476782, 269476859, 0): np.float64(0.0125),
(269476859, 269476781, 0): np.float64(0.0),
(269476859, 269476782, 0): np.float64(0.025),
(269476859, 9847831441, 0): np.float64(0.0),
(271872441, 29273056, 0): np.float64(0.0),
(271872510, 29273049, 0): np.float64(0.025),
(271872510, 1289066093, 0): np.float64(0.0),
(271872510, 9787291407, 0): np.float64(0.0),
(271872510, 10593327815, 0): np.float64(0.0),
(283251475, 29217293, 0): np.float64(0.0),
(283251475, 164641499, 0): np.float64(0.0),
(287447524, 29215046, 0): np.float64(0.04),
(287447524, 287447547, 0): np.float64(0.0),
(287447547, 287447548, 0): np.float64(0.06),
(287447548, 26517409, 0): np.float64(0.0),
(287447548, 29215047, 0): np.float64(0.03333333333333334),
(287447549, 29217276, 0): np.float64(0.0033333333333333335),
(287447549, 287447524, 0): np.float64(0.06),
(292682410, 29215067, 0): np.float64(0.0),
(292682410, 3352479272, 0): np.float64(0.03333333333333333),
(318545589, 29276746, 0): np.float64(0.0033333333333333335),
(318545591, 88071064, 0): np.float64(0.013333333333333332),
(318545591, 318545589, 0): np.float64(0.0),
(318545591, 536286819, 0): np.float64(0.013333333333333332),
(318545593, 27555214, 0): np.float64(0.01),
(318545593, 29276961, 0): np.float64(0.0),
(318545595, 29276959, 0): np.float64(0.0125),
(318549151, 27555224, 0): np.float64(0.0),
(318549151, 29276750, 0): np.float64(0.006666666666666666),
(318549151, 353457764, 0): np.float64(0.0125),
(338894718, 28794519, 0): np.float64(0.0),
(338894718, 1491865195, 0): np.float64(0.0),
(353457764, 318549151, 0): np.float64(0.0),
(357477858, 29275853, 0): np.float64(0.0022222222222222222),
(357477858, 2814705317, 0): np.float64(0.0),
(357477858, 3242035266, 0): np.float64(0.0),
(359541354, 29217348, 0): np.float64(0.0),
(412194367, 2757332331, 0): np.float64(0.0),
(442649842, 173953089, 0): np.float64(0.0),
(442649842, 1539712867, 0): np.float64(0.0033333333333333335),
(442649846, 29788946, 0): np.float64(0.0033333333333333335),
(442649846, 8078047384, 0): np.float64(0.0),
(536286819, 29276750, 0): np.float64(0.013333333333333332),
(536286819, 318545591, 0): np.float64(0.0033333333333333335),
(536286819, 3900230865, 0): np.float64(0.025),
```

```
(536286819, 11391147437, 0): np.float64(0.0625),
(611862495, 28794519, 0): np.float64(0.01),
(611862495, 292682410, 0): np.float64(0.006666666666666666),
(611862495, 338894718, 0): np.float64(0.0125),
(612417324, 28794539, 0): np.float64(0.0),
(660778774, 29215073, 0): np.float64(0.0125),
(660778774, 196725581, 0): np.float64(0.0044444444444444444),
(848523542, 28794517, 0): np.float64(0.0125),
(1237777024, 5450916417, 0): np.float64(0.0),
(1285251472, 29217276, 0): np.float64(0.02),
(1285251472, 29218294, 0): np.float64(0.013333333333333332),
(1285251472, 1285251502, 0): np.float64(0.0),
(1285251502, 29217276, 0): np.float64(0.0125),
(1285251502, 29217277, 0): np.float64(0.0),
(1285251502, 1285251472, 0): np.float64(0.0),
(1289066093, 271872510, 0): np.float64(0.0),
(1491865195, 21487230, 0): np.float64(0.0125),
(1491865195, 338894718, 0): np.float64(0.0),
(1491865195, 3352479275, 0): np.float64(0.0),
(1491865195, 12036921938, 0): np.float64(0.0125),
(1539712867, 27555221, 0): np.float64(0.006666666666666666),
(1539712867, 2677649274, 0): np.float64(0.0375),
(1814668312, 26517409, 0): np.float64(0.0),
(1814668312, 1814668313, 0): np.float64(0.0),
(1814668313, 1814668312, 0): np.float64(0.0),
(1814668480, 292682410, 0): np.float64(0.0),
(1814668480, 611862495, 0): np.float64(0.04),
(1969136398, 29218288, 0): np.float64(0.006666666666666666),
(1969136398, 5984003377, 0): np.float64(0.0),
(1984006630, 29273058, 0): np.float64(0.0),
(1984006630, 29273064, 0): np.float64(0.006666666666666666),
(2028472686, 29785151, 0): np.float64(0.0),
(2143522428, 29218326, 0): np.float64(0.0125),
(2143522428, 29276730, 0): np.float64(0.0),
(2143522428, 29276737, 0): np.float64(0.0375),
(2476048346, 29273067, 0): np.float64(0.0125),
(2476048346, 29273068, 0): np.float64(0.0),
(2476048346, 252778554, 0): np.float64(0.0),
(2479455794, 29273067, 0): np.float64(0.0),
(2479455794, 2814705317, 0): np.float64(0.0375),
(2625899861, 26960761, 0): np.float64(0.015555555555555555),
(2625899861, 29218321, 0): np.float64(0.01),
(2625899861, 2891066381, 0): np.float64(0.0),
(2677649274, 29217325, 0): np.float64(0.016666666666666666),
(2757332331, 2757332333, 0): np.float64(0.0),
(2757332333, 29273071, 0): np.float64(0.0033333333333333333),
(2757332333, 412194367, 0): np.float64(0.013333333333333332),
(2757332333, 2757332331, 0): np.float64(0.0),
(2814705317, 29273069, 0): np.float64(0.05),
```

```
 (2814705317, 357477858, 0): np.float64(0.0),
 (2814705317, 2479455794, 0): np.float64(0.05),
 (2891066381, 2625899861, 0): np.float64(0.0125),
 (3242035261, 29275853, 0): np.float64(0.0033333333333333335),
 (3242035261, 3242035266, 0): np.float64(0.006666666666666667),
 (3242035262, 2028472686, 0): np.float64(0.0022222222222222222),
 (3242035262, 3242035261, 0): np.float64(0.03333333333333333),
 (3242035266, 357477858, 0): np.float64(0.0125),
 (3242035266, 10254148068, 0): np.float64(0.006666666666666667),
 (3243910627, 29276213, 0): np.float64(0.10000000000000002),
 (3243910627, 29276217, 0): np.float64(0.008333333333333333),
 (3243910627, 29276687, 0): np.float64(0.013333333333333332),
 (3352479272, 28373656, 0): np.float64(0.02),
 (3352479272, 3352479275, 0): np.float64(0.0),
 (3352479275, 1491865195, 0): np.float64(0.025),
 (3352479275, 1814668480, 0): np.float64(0.039999999999999994),
 (3352479275, 3352479272, 0): np.float64(0.025),
 (3352479286, 262484667, 0): np.float64(0.0),
 (3352479286, 3352479290, 0): np.float64(0.0),
 (3352479290, 28794519, 0): np.float64(0.01),
 (3352479290, 3463840860, 0): np.float64(0.006666666666666667),
 (3411530669, 26960761, 0): np.float64(0.0022222222222222222),
 (3411530669, 10312836355, 0): np.float64(0.0),
 (3463840860, 28794518, 0): np.float64(0.006666666666666667),
 (3463840860, 3352479290, 0): np.float64(0.006666666666666667),
 (3463840860, 11759269145, 0): np.float64(0.0),
 (3900230865, 29276959, 0): np.float64(0.0),
 (3900230865, 318545589, 0): np.float64(0.006666666666666667),
 (3900230865, 536286819, 0): np.float64(0.0),
 (5450916417, 29218299, 0): np.float64(0.025),
 (5450916417, 29218319, 0): np.float64(0.0),
 (5450916417, 1237777024, 0): np.float64(0.0),
 (5984003368, 21487231, 0): np.float64(0.0125),
 (5984003368, 29217269, 0): np.float64(0.025),
 (5984003368, 5984003369, 0): np.float64(0.0),
 (5984003369, 29788955, 0): np.float64(0.0),
 (5984003369, 5984003368, 0): np.float64(0.0),
 (5984003369, 5984003376, 0): np.float64(0.0),
 (5984003376, 29217269, 0): np.float64(0.0125),
 (5984003376, 5984003377, 0): np.float64(0.0125),
 (5984003377, 29216575, 0): np.float64(0.006666666666666667),
 (5984003377, 1969136398, 0): np.float64(0.025),
 (5984003377, 5984003376, 0): np.float64(0.0125),
 (8078047384, 29218287, 0): np.float64(0.0033333333333333335),
 (8078047384, 442649846, 0): np.float64(0.0125),
 (9500676798, 196774436, 0): np.float64(0.0),
 (9500676798, 10593327815, 0): np.float64(0.0),
 (9787291407, 271872510, 0): np.float64(0.0125),
 (9847831441, 269476859, 0): np.float64(0.0125),
 (10254148068, 28373648, 0): np.float64(0.0044444444444444444),
```

```
 (10254148068, 357477858, 0): np.float64(0.02),
 (10312836355, 26960758, 0): np.float64(0.0),
 (10312836355, 1984006630, 0): np.float64(0.0),
 (10593327815, 271872510, 0): np.float64(0.0),
 (11391147437, 29276743, 0): np.float64(0.0),
 (11391147437, 29276752, 0): np.float64(0.1125),
 (11391147437, 536286819, 0): np.float64(0.025),
 (11759269145, 3463840860, 0): np.float64(0.0),
 (11806567841, 29217327, 0): np.float64(0.02),
 (12036921938, 1491865195, 0): np.float64(0.0125)}
```

```
<ipython-input-6-5b8c658043ff>:220: MatplotlibDeprecationWarning: The
get_cmap function was deprecated in Matplotlib 3.7 and will be removed
in 3.11. Use ``matplotlib.colormaps[name]`` or
``matplotlib.colormaps.get_cmap()`` or ``pyplot.get_cmap()`` instead.
  cmap = cm.get_cmap("OrRd")
```




```
(<Figure size 1000x1000 with 2 Axes>,
 <Axes: title={'center': 'Traffic Congestion Map'}>)
```

```
## Theoretical Analysis
betweenness_centrality =
nx.betweenness_centrality(berlin_network.graph, weight="travel_time")
edge_betweenness_centrality =
nx.edge_betweenness_centrality(berlin_network.graph,
weight="travel_time")
closeness_centrality = nx.closeness_centrality(berlin_network.graph,
```

```python
distance="travel_time")
degree_centrality = nx.degree_centrality(berlin_network.graph)
print(f"Between centrality:", betweenness_centrality)
print(f"Edge between centrality:", edge_betweenness_centrality)
print(f"Closeness centrality:", closeness_centrality)
print(f"Degree centrality:", degree_centrality)
```

```
Between centrality: {21487224: 0.020275319567354965, 21487230:
0.0535889872173058, 21487231: 0.04232055063913471, 21487232:
0.23089478859390364, 26517409: 0.008770894788593903, 26960758:
0.03710914454277286, 26960761: 0.20214355948869223, 26960762:
0.27058013765978367, 27555211: 0.0, 27555214: 0.008180924287118977,
27555215: 0.012684365781710914, 27555221: 0.030285152409046213,
27555224: 0.022241887905604718, 27555225: 0.013254670599803344,
27555226: 0.0003343166175024582, 27555227: 0.012920353982300885,
28373648: 0.025034414945919372, 28373656: 0.1377384464110128,
28794517: 0.008810226155358898, 28794518: 0.025152409046214356,
28794519: 0.0696558505408063, 28794539: 0.008790560471976402,
28794542: 0.18200589970501474, 29215046: 0.13390363815142575,
29215047: 0.09634218289085546, 29215049: 0.14271386430678465,
29215057: 0.025801376597836774, 29215058: 0.017718780727630286,
29215060: 0.00031465093411996067, 29215063: 0.0, 29215067:
0.012546705998033432, 29215071: 0.0, 29215073: 0.02623402163225172,
29216571: 0.008770894788593903, 29216572: 0.008692232055063913,
29216575: 0.008829891838741397, 29217269: 0.021002949852507374,
29217276: 0.07706981317600786, 29217277: 0.004464110127826942,
29217280: 0.08932153392330383, 29217293: 0.1032251720747296, 29217303:
0.0, 29217317: 0.18674532940019667, 29217319: 0.0, 29217321:
0.08792527040314652, 29217322: 0.14761061946902654, 29217325:
0.08945919370698131, 29217327: 0.0799803343166175, 29217332:
0.07447394296951819, 29217340: 0.08015732546705998, 29217342:
0.005152409046214356, 29217348: 0.013549655850540807, 29218287:
0.025388397246804327, 29218288: 0.031366764995083576, 29218289:
0.007905604719764012, 29218291: 3.9331366764995083e-05, 29218293:
0.14843657817109143, 29218294: 0.0743559488692232, 29218295:
0.010835791543756145, 29218296: 0.020117994100294984, 29218299:
0.09223205506391347, 29218302: 0.0912487708947886, 29218305:
0.008121927236971485, 29218313: 0.0, 29218315: 0.083952802359882,
29218316: 0.01559488692232055, 29218318: 0.009203539823008849,
29218319: 0.010894788593903639, 29218321: 0.17010816125860373,
29218322: 0.020491642084562438, 29218323: 0.10021632251720747,
29218324: 0.18861356932153392, 29218325: 0.05726647000983284,
29218326: 0.1809439528023599, 29218327: 0.14357915437561455, 29218328:
0.0, 29219437: 0.020334316617502457, 29219438: 0.020393313667649952,
29271261: 0.03705014749262537, 29273048: 0.07238938053097345,
29273049: 0.034198623402163224, 29273050: 0.01960668633235005,
29273056: 0.036440511307767945, 29273057: 0.10125860373647984,
29273058: 0.06827925270403147, 29273064: 0.28947885939036383,
29273065: 0.22930186823992132, 29273067: 0.016656833824975417,
29273068: 0.01815142576204523, 29273069: 0.14269419862340216,
```

29273071: 0.027394296951819077, 29273075: 0.0029891838741396264,
29273076: 0.011543756145526057, 29273077: 0.010678466076696164,
29273078: 0.0, 29273079: 0.01718780727630285, 29273080:
0.035181907571288105, 29275853: 0.0568338249754179, 29275983:
0.02175024582104228, 29276209: 0.2855653883972468, 29276210: 0.0,
29276211: 0.06977384464110128, 29276213: 0.2876892822025565, 29276217:
0.26043264503441493, 29276222: 0.05465093411996067, 29276687:
0.15006882989183873, 29276688: 0.062477876106194694, 29276689:
0.04627335299901671, 29276690: 0.030658800393313666, 29276726:
0.02294985250737463, 29276728: 0.017148475909537858, 29276730:
0.025486725663716812, 29276737: 0.03929203539823009, 29276743:
0.058662733529990164, 29276745: 0.01974434611602753, 29276746:
0.04017699115044248, 29276750: 0.09299901671583087, 29276752:
0.10247787610619469, 29276753: 0.08444444444444445, 29276755:
0.08896755162241887, 29276757: 0.0066470009832841695, 29276759:
0.01781710914454277, 29276762: 0.0045427728613569324, 29276764:
0.011032448377581121, 29276773: 0.042418879056047194, 29276775:
0.023126843657817107, 29276777: 0.03008849557522124, 29276778:
0.027315634218289087, 29276959: 0.03911504424778761, 29276961:
0.03968534906588004, 29785137: 1.9665683382497542e-05, 29785151:
0.017482792527040315, 29787489: 0.0, 29788946: 0.03465093411996067,
29788955: 0.024503441494591938, 32248754: 0.0, 60237352:
0.017266470009832842, 88070313: 0.027079646017699115, 88071064:
0.06470009832841692, 88071065: 0.06308751229105211, 88071069:
0.04737463126843658, 122087860: 0.017325467059980334, 164641499:
0.01889872173058014, 173953089: 0.006843657817109145, 196724115:
0.04206489675516224, 196725581: 0.01864306784660767, 196774436: 0.0,
252778554: 0.0, 262484667: 0.012251720747295969, 267214998:
0.010501474926253687, 268466879: 0.08009832841691249, 269476781: 0.0,
269476782: 0.054985250737463125, 269476859: 0.017541789577187807,
271872441: 0.0, 271872510: 0.03246804326450344, 283251475:
0.03706981317600787, 287447524: 0.08647000983284168, 287447547:
0.0608062930186824, 287447548: 0.07390363815142577, 287447549:
0.12141592920353983, 292682410: 0.057954768928220254, 318545589:
0.026096361848574237, 318545591: 0.06096361848574238, 318545593:
0.03559488692232055, 318545595: 0.0, 318549151: 0.046568338249754176,
338894718: 0.005624385447394297, 353457764: 0.0, 357477858:
0.026705998033431662, 359541354: 0.0, 412194367: 0.0, 442649842:
0.03793510324483776, 442649846: 0.035221238938053095, 536286819:
0.14444444444444443, 611862495: 0.07551622418879056, 612417324: 0.0,
660778774: 0.020334316617502457, 848523542: 0.0, 1237777024: 0.0,
1285251472: 0.06904621435594886, 1285251502: 0.004759095378564405,
1289066093: 0.0, 1491865195: 0.01431661750245821, 1539712867:
0.03581120943952802, 1814668312: 0.004405113077679449, 1814668313:
0.0, 1814668480: 0.05172074729596853, 1969136398:
0.028102261553588987, 1984006630: 0.04456243854473943, 2028472686:
0.017502458210422813, 2143522428: 0.0352015732546706, 2476048346:
0.01032448377581121, 2479455794: 0.024169124877089478, 2625899861:
0.16448377581120943, 2677649274: 0.04753195673549656, 2757332331:

0.004365781710914454, 2757332333: 0.017522123893805308, 2814705317:
0.032094395280235985, 2891066381: 0.0, 3242035261:
0.05716814159292035, 3242035262: 0.05752212389380531, 3242035266:
0.0255850540806293, 3243910627: 0.2583284169124877, 3352479272:
0.06650934119960669, 3352479275: 0.07431661750245822, 3352479286:
0.012271386430678466, 3352479290: 0.058407079646017, 3411530669:
0.043264503441494594, 3463840860: 0.04206489675516224, 3900230865:
0.061730580137659787, 5450916417: 0.0152802359882005 9, 5984003368:
0.03895771878072763, 5984003369: 0.020688298918387416, 5984003376:
0.027964601769911505, 5984003377: 0.039764011799410026, 8078047384:
0.02936086529006883, 9500676798: 0.019783677482792528, 9787291407:
0.0, 9847831441: 0.0, 10254148068: 0.03307767944936087, 10312836355:
0.043775811209439526, 10593327815: 0.015535889872173058, 11391147437:
0.09604719764011799, 11759269145: 0.0, 11806567841: 0.080039331366765,
12036921938: 0.0}
Edge between centrality: {(21487224, 196725581, 0):
0.008225800163736308, (21487224, 29215073, 0): 0.01621769131807727,
(21487230, 26960762, 0): 0.021928969630813613, (21487230, 28794539,
0): 0.008771587852325445, (21487230, 196724115, 0):
0.0234493781918833 6, (21487230, 1491865195, 0): 0.003313710966434057 4,
(21487231, 29215047, 0): 0.026665627071069356, (21487231, 5984003368,
0): 0.019628864371759387, (21487232, 29218315, 0):
0.051011656465634866, (21487232, 269476782, 0): 0.031694670773069276,
(21487232, 26960762, 0): 0.08678024248567308, (21487232, 29218293, 0):
0.0637207126427819 7, (26517409, 287447548, 0): 0.013040427273790496,
(26960758, 29271261, 0): 0.04112900081868153, (26960761, 3411530669,
0): 0.028049588709991816, (26960761, 29218316, 0): 0.009824178394604 5,
(26960761, 28794542, 0): 0.0881252192896963 1, (26960761, 2625899861,
0): 0.07871038166153367, (26960762, 21487230, 0): 0.02781567970059647,
(26960762, 28373656, 0): 0.07258976258235547, (26960762, 28794542, 0):
0.08896339324002964, (26960762, 21487232, 0): 0.08317414525749484,
(27555211, 173953089, 0): 0.002417059763751900 7, (27555211, 27555221,
0): 0.001929749327511598, (27555214, 27555215, 0):
0.012455654750302133, (27555215, 27555227, 0): 0.016919418346263305,
(27555221, 29276746, 0): 0.002982339869790651 5, (27555221, 29276959,
0): 0.03138279209387548, (27555224, 318549151, 0):
0.00871311059997661, (27555224, 27555227, 0): 0.00029238626174418 15,
(27555224, 27555225, 0): 0.017387236365053994, (27555225, 27555226,
0): 0.004736657440255741, (27555225, 122087860, 0):
0.012748041012046315, (27555226, 27555215, 0): 0.004522240848310008,
(27555226, 27555225, 0): 0.00015593933959689 68, (27555227, 27555224,
0): 0.000526295271139526 7, (27555227, 29276961, 0):
0.016627032084519124, (28373648, 29273079, 0): 0.012631086507348642,
(28373648, 10254148068, 0): 0.016529569997271062, (28373656,
3352479275, 0): 0.0690421426065260 7, (28373656, 26960762, 0):
0.0718295583018206, (28794517, 848523542, 0): 0.004405286343612335,
(28794517, 28794518, 0): 0.008401231920782817, (28794517, 29215060,
0): 0.00027289384429456944, (28794518, 3463840860, 0):
0.012475147167751745, (28794518, 28794517, 0): 0.008654633347627774,

(28794518, 3352479286, 0): 0.008147830493937858, (28794519, 611862495, 0): 0.037444933920704845, (28794519, 338894718, 0): 0.0030018322872402635, (28794519, 3352479290, 0): 0.03294218548984445, (28794539, 612417324, 0): 0.004405286343612335, (28794539, 28794542, 0): 0.007718997310046392, (28794539, 21487230, 0): 0.0009356360375813808, (28794542, 26960762, 0): 0.0920626876145l795, (28794542, 26960761, 0): 0.09268644497290554, (29215046, 29215049, 0): 0.08151728977427782, (29215046, 287447547, 0): 0.05555338973139449, (29215047, 287447549, 0): 0.077638298701805, (29215047, 21487231, 0): 0.022201863475108184, (29215049, 29218293, 0): 0.085376788429301, (29215049, 29218289, 0): 0.004229854586565826, (29215049, 29215046, 0): 0.05619663950723169, (29215057, 29215058, 0): 0.00937585279326342, (29215057, 29215067, 0): 0.006257066001325484, (29215057, 269476782, 0): 0.01428794199056567, (29215058, 29215060, 0): 0.004444271178511559, (29215058, 262484667, 0): 0.008693618182526997, (29215058, 29215057, 0): 0.008771587852325445, (29215060, 29215058, 0): 0.004522240848310008, (29215060, 28794517, 0): 0.00013644692214728472, (29215063, 29215067, 0): 0.004346809091263499, (29215067, 29215057, 0): 0.005652801060387509, (29215067, 29215063, 0): 0.004405286343612335, (29215067, 292682410, 0): 0.006724884020116175, (29215071, 29215073, 0): 0.004346809091263499, (29215073, 660778774, 0): 0.0032942185489844454, (29215073, 196724115, 0): 0.02265018907644926, (29215073, 29215071, 0): 0.004405286343612335, (29216571, 29216575, 0): 1.94924174496121e-05, (29216571, 29216572, 0): 0.013020934856340883, (29216572, 1969136398, 0): 0.012962457603992047, (29216575, 29216571, 0): 0.013098904526139332, (29217269, 5984003376, 0): 0.01344976804023235, (29217269, 5984003368, 0): 0.011714942887216874, (29217276, 287447549, 0): 0.04711317297571245, (29217276, 1285251472, 0): 0.029160656504619704, (29217276, 1285251502, 0): 0.004463763595961171, (29217277, 5984003369, 0): 0.006101126661728588, (29217277, 1285251502, 0): 0.002670461190596858, (29217280, 283251475, 0): 0.041148493236131144, (29217280, 29217277, 0): 0.006529959845620054, (29217280, 29218294, 0): 0.04520291606565047, (29217293, 29217317, 0): 0.03937468324821645, (29217293, 29217280, 0): 0.0537990721609294, (29217293, 29217348, 0): 0.013488752875131575, (29217303, 29218288, 0): 0.004366301508713111, (29217317, 29217293, 0): 0.0531558223850922, (29217317, 29218327, 0): 0.05054383844684418, (29217317, 29217322, 0): 0.05769755565085182, (29217317, 88071069, 0): 0.028049588709991816, (29217319, 164641499, 0): 0.004346809091263499, (29217321, 29217332, 0): 0.07822307122529336, (29217321, 267214998, 0): 0.01327433628318584, (29217322, 29217321, 0): 0.09155588476082804, (29217322, 29217317, 0): 0.059101009707223894, (29217325, 29217322, 0): 0.09301781606954895, (29217327, 442649842, 0): 0.04200615960391408, (29217327, 29217325, 0): 0.04161631125492184, (29217332, 29217340, 0): 0.04516393123075124, (29217332, 8078047384, 0): 0.033000662742193286, (29217340, 268466879, 0): 0.08379790261588242, (29217342, 29218289, 0): 0.002085688667108495, (29217342, 29217280, 0): 0.007368133795953374, (29217348, 359541354,

0): 0.004405286343612335, (29217348, 29218296, 0):
0.013371798370433901, (29218287, 29218288, 0): 0.00510701337179837,
(29218287, 5984003377, 0): 0.016295660987875715, (29218287, 29788955,
0): 0.00810884565903863, (29218288, 29218287, 0):
0.00828427741608514З, (29218288, 442649846, 0): 0.027152937507309658,
(29218289, 29215049, 0): 0.004210362169116214, (29218289, 29218291,
0): 0.0020272114147596586, (29218289, 29217342, 0):
0.002728938442945694, (29218289, 29218294, 0): 0.0032162488791859967,
(29218291, 29218289, 0): 0.0016763479006666408, (29218291, 29218293,
0): 0.002436552181201513, (29218291, 29218295, 0):
0.00027289384429456944, (29218293, 29218291, 0):
0.0024170597637519007, (29218293, 29215049, 0): 0.060134107832053335,
(29218293, 21487232, 0): 0.08892440840513041, (29218294, 1285251472,
0): 0.04202565202136369, (29218294, 29218289, 0):
0.004249347004015438, (29218294, 29217280, 0): 0.031772640442867726,
(29218295, 29218291, 0): 0.0, (29218295, 29217342, 0):
0.006783361272465011, (29218295, 29218296, 0): 0.0037425441503255233,
(29218295, 29218299, 0): 0.004561225683209232, (29218296, 29218326,
0): 0.010369966083193637, (29218296, 29218295, 0):
0.01177342013956571, (29218296, 29218325, 0): 0.002144165919457331,
(29218299, 29218315, 0): 0.032259950879108025, (29218299, 29218295,
0): 0.0030992943744883243, (29218299, 29218302, 0):
0.053643132821332505, (29218299, 5450916417, 0): 0.006763868855015399,
(29218302, 29218325, 0): 0.059023040037425444, (29218302, 29218299,
0): 0.02912167166972048, (29218302, 29218305, 0):
0.006646914350З177265, (29218305, 29218302, 0): 0.004327316673813887,
(29218305, 29218318, 0): 0.006822346107З642355, (29218305, 29218322,
0): 0.0012475147167751745, (29218313, 29218315, 0):
0.004346809091263499, (29218315, 29218313, 0): 0.004405286343612335,
(29218315, 21487232, 0): 0.030973451327433628, (29218315, 29218299,
0): 0.05218120151261159б, (29218316, 29218319, 0):
0.009843670812054112, (29218316, 26960761, 0): 0.009960625316751784,
(29218318, 29218319, 0): 0.0002339090093953452, (29218318, 29218316,
0): 0.010038594986550232, (29218318, 29218305, 0):
0.0031967564617363846, (29218319, 5450916417, 0): 0.00844021675568204,
(29218319, 29218318, 0): 0.006705391602666563, (29218321, 29276217,
0): 0.07656621574207634, (29218321, 2625899861, 0):
0.08438267513937078, (29218321, 29218322, 0): 0.012007329148961054,
(29218322, 29218321, 0): 0.010623367510038595, (29218322, 29218323,
0): 0.009687731472457214, (29218322, 29276222, 0):
0.001734825153015477, (29218322, 29218305, 0): 0.002611983938248022,
(29218323, 29218324, 0): 0.06044598651124713, (29218323, 29218302, 0):
0.0338388З669252661, (29218323, 29218322, 0): 0.009395345210713033,
(29218324, 29218326, 0): 0.09943082141047133, (29218324, 29218323, 0):
0.037600873260301745, (29218324, 29276211, 0): 0.05426689017972009,
(29218325, 29218302, 0): 0.003040817122139487, (29218325, 29218324,
0): 0.05689836653541772, (29218325, 29218296, 0):
0.0011695450469767, (29218326, 29218327, 0): 0.07401270905617716,
(29218326, 29218324, 0): 0.07401270905617716, (29218326, 29218296, 0):

0.006062141826829364, (29218326, 2143522428, 0): 0.02960898210596078,
(29218327, 29218328, 0): 0.004405286343612335, (29218327, 29276743,
0): 0.0034306654711317З, (29218327, 29218326, 0): 0.06676152976492145,
(29218327, 29217317, 0): 0.07206346731121593, (29218328, 29218328, 0):
0.0, (29218328, 29218328, 1): 0.0, (29218328, 29218327, 0):
0.004346809091263499, (29219437, 21487224, 0): 0.02450196873416241,
(29219438, 29219437, 0): 0.024560445986511248, (29271261, 29273080,
0): 0.025262173014697284, (29271261, 29219438, 0):
0.015808350551635413, (29273048, 29273049, 0): 0.010837784101984329,
(29273048, 29273057, 0): 0.04810728626564267, (29273048, 29273071, 0):
0.017153327355658648, (29273049, 271872510, 0): 0.008147830493937858,
(29273049, 29273048, 0): 0.030096292542201086, (29273050, 9500676798,
0): 0.01966784920665861, (29273050, 29273056, 0):
0.004112900081868154, (29273056, 271872441, 0): 0.004405286343612335,
(29273056, 29273058, 0): 0.012221745740906788, (29273056, 29273050,
0): 0.0238392265408756, (29273057, 29273068, 0): 0.012104791236209115,
(29273057, 29273056, 0): 0.002066196249658827, (29273057, 29273048,
0): 0.029531012436162334, (29273057, 29273069, 0):
0.0610112666172858, (29273058, 1984006630, 0): 0.011792912557015322,
(29273058, 26960758, 0): 0.03023273946434837, (29273058, 29273056, 0):
0.029998830454953024, (29273064, 29273065, 0): 0.09110755915948696,
(29273064, 29276209, 0): 0.14135901134458695, (29273064, 3242035262,
0): 0.05880862344547971, (29273065, 29273064, 0): 0.12576507738489728,
(29273065, 29273058, 0): 0.03953062258781334, (29273065, 29273069, 0):
0.06633269658102998, (29273067, 2476048346, 0): 0.00881057268722467,
(29273067, 2479455794, 0): 0.01204631398386028, (29273068, 29273057,
0): 0.012358192663054072, (29273068, 2476048346, 0):
0.0014814237261705197, (29273068, 29273076, 0): 0.008498694008030876,
(29273069, 29273057, 0): 0.044306264862968306, (29273069, 29273065,
0): 0.08251140306420802, (29273069, 2814705317, 0):
0.018966122178472575, (29273071, 2757332333, 0): 0.013098904526139332,
(29273071, 29273076, 0): 0.0018712720751627616, (29273071, 29273048,
0): 0.016529569997271062, (29273075, 29273077, 0):
0.0018322872402635375, (29273075, 29273076, 0): 0.005477369303341001,
(29273076, 29273068, 0): 0.008654633347627774, (29273076, 29273071,
0): 0.0014814237261705197, (29273076, 29273075, 0):
0.005652801060387509, (29273077, 29273079, 0): 0.00881057268722467,
(29273077, 29273075, 0): 0.001715332735565865, (29273077, 29273078,
0): 0.004405286343612335, (29273078, 29273077, 0):
0.004346809091263499, (29273079, 29273077, 0): 0.00881057268722467,
(29273079, 28373648, 0): 0.012572609254999806, (29273080, 10312836355,
0): 0.019979727885852405, (29273080, 3411530669, 0):
0.019239016022767143, (29275853, 3242035262, 0): 0.002611983938248022,
(29275853, 29273065, 0): 0.05806791158239445, (29275983, 60237352, 0):
0.02151962886437176, (29275983, 29785137, 0): 0.004385793926162723,
(29276209, 29273064, 0): 0.13738255818486608, (29276209, 29276210, 0):
0.004405286343612335, (29276209, 29276213, 0): 0.1456083583486024,
(29276210, 29276209, 0): 0.004346809091263499, (29276211, 3243910627,
0): 0.0660403103192858, (29276211, 29276726, 0): 0.007465595883201435,

(29276213, 29276217, 0): 0.14775252426805974, (29276213, 29276209, 0): 0.1417488596935792, (29276217, 3243910627, 0): 0.12352344937819189, (29276217, 29218321, 0): 0.08212155471521579, (29276217, 29276222, 0): 0.05683988928306885, (29276222, 29218323, 0): 0.05645004093407665, (29276222, 29218322, 0): 0.0020661962496588827, (29276687, 29276755, 0): 0.03834158512338700 6, (29276687, 3243910627, 0): 0.07089392226423921, (29276687, 29276688, 0): 0.04385793926162722 5, (29276688, 29276689, 0): 0.03613894195158083 6, (29276688, 29276757, 0): 0.004015437994620093, (29276688, 29276687, 0): 0.026119839382480216, (29276689, 29276690, 0): 0.028205528049588712, (29276689, 29276688, 0): 0.017952516471092746, (29276689, 29276762, 0): 0.004054422829519317, (29276690, 29276689, 0): 0.009492807297961094, (29276690, 29276778, 0): 0.02524268059724767, (29276726, 29276755, 0): 0.004366301508713111, (29276726, 29276211, 0): 0.01929749327511598, (29276726, 29276728, 0): 0.0034306654711317 3, (29276728, 29276753, 0): 0.0006237573583875873, (29276728, 29276726, 0): 0.016724494171767183, (29276728, 29276730, 0): 0.0039959455771704806, (29276730, 29276752, 0): 0.0033916806362325057, (29276730, 29276728, 0): 0.016529569997271062, (29276730, 2143522428, 0): 0.009687731472457214, (29276737, 29276730, 0): 0.017016880433511364, (29276737, 29276743, 0): 0.026275778722077113, (29276743, 88071065, 0): 0.027893649370394916, (29276743, 29276737, 0): 0.016081244395929985, (29276743, 11391147437, 0): 0.0007017270281860356, (29276743, 29218327, 0): 0.017816069548945462, (29276745, 88071064, 0): 0.01549647187244162, (29276745, 88071069, 0): 0.002456044598651125, (29276745, 88070313, 0): 0.0020467038322092707, (29276745, 88071065, 0): 0.003917975907372032, (29276746, 27555221, 0): 0.007680012475147168, (29276746, 2677649274, 0): 0.036489805465673854, (29276750, 3900230865, 0): 0.014054032981170325, (29276750, 536286819, 0): 0.05261003469650306 4, (29276750, 318545593, 0): 0.0298623835328057 4, (29276752, 29276753, 0): 0.04921835406027056, (29276752, 11391147437, 0): 0.040505243460293945, (29276752, 29276759, 0): 0.012319207828154849, (29276752, 29276730, 0): 0.003878991072472808, (29276753, 29276759, 0): 0.00042883318389146623, (29276753, 29276755, 0): 0.04947175548711551 4, (29276753, 29276752, 0): 0.036704222057619584, (29276753, 29276728, 0): 0.0014424388912712956, (29276755, 29276753, 0): 0.03799072160929399, (29276755, 29276687, 0): 0.0488090132938287, (29276755, 29276726, 0): 0.0029628474523410394, (29276755, 29276757, 0): 0.0027679232778449183, (29276757, 29276755, 0): 0.0004093407664418541, (29276757, 29276762, 0): 0.0008186815328837082, (29276757, 29276759, 0): 0.005184983041596819, (29276757, 29276688, 0): 0.004522240848310008, (29276759, 29276764, 0): 0.010681844762387432, (29276759, 29276752, 0): 0.0069977778644107445, (29276759, 29276753, 0): 0.00027289384429456944, (29276759, 29276757, 0): 0.004054422829519317, (29276762, 29276689, 0): 0.00463919535300768, (29276762, 29276764, 0): 0.003820513820123972, (29276762, 29276757, 0): 0.0001559393395968968, (29276762, 29276778, 0): 0.0002339090093953452, (29276764, 29276762, 0): 0.003937468324821645, (29276764, 29276759, 0):

0.0041323924993177654, (29276764, 29276777, 0): 0.007212194456356478, (29276773, 122087860, 0): 0.0044832560134107835, (29276773, 318549151, 0): 0.03750341117305368, (29276773, 32248754, 0): 0.004405286343612335, (29276775, 29276773, 0): 0.024989279170402716, (29276775, 29276777, 0): 0.002280612841604616, (29276777, 29276778, 0): 0.006003664574480527, (29276777, 29276775, 0): 0.027328369264356167, (29276777, 29276764, 0): 0.0008381739503333204, (29276778, 29276762, 0): 9.746208724806051e-05, (29276778, 29276777, 0): 0.024735877743557757, (29276778, 29276690, 0): 0.006588437097968891, (29276959, 3900230865, 0): 0.028887762660325136, (29276959, 318545593, 0): 0.0098241783946045, (29276959, 318545595, 0): 0.004405286343612335, (29276961, 27555224, 0): 0.018888152508674125, (29276961, 29276750, 0): 0.024794354995906594, (29785137, 29785151, 0): 3.89848348992242e-05, (29785137, 29275983, 0): 0.004327316673813887, (29785137, 60237352, 0): 0.0, (29785151, 2028472686, 0): 3.89848348992242e-05, (29785151, 29275983, 0): 0.021636583369069432, (29785151, 29785137, 0): 0.0, (29787489, 122087860, 0): 0.004346809091263499, (29788946, 29217340, 0): 0.038692448637480024, (29788955, 267214998, 0): 0.0015399009785193561, (29788955, 5984003369, 0): 0.014502358582511404, (29788955, 164641499, 0): 0.012592101672449417, (32248754, 29276773, 0): 0.004346809091263499, (60237352, 29785137, 0): 3.89848348992242e-05, (60237352, 3242035261, 0): 0.021422166777123698, (88070313, 29276746, 0): 0.01103270827648045, (88070313, 29276745, 0): 0.0012280222993255624, (88070313, 88071064, 0): 0.01892713734357335, (88071064, 88071065, 0): 0.03512533624420101, (88071064, 29276745, 0): 0.002261120424155004, (88071064, 318545591, 0): 0.031090405832131303, (88071065, 29276743, 0): 0.0286733460683794, (88071065, 88071069, 0): 0.02085688667108495, (88071065, 29276745, 0): 0.01734825153015477, (88071069, 29276745, 0): 0.0031382792093875484, (88071069, 88070313, 0): 0.02919964133951893, (88071069, 29217317, 0): 0.018966122178472575, (122087860, 29276773, 0): 0.017114342520759426, (122087860, 29787489, 0): 0.004405286343612335, (164641499, 29217319, 0): 0.004405286343612335, (164641499, 29217293, 0): 0.018673735916728395, (173953089, 27555211, 0): 0.004405286343612335, (173953089, 1539712867, 0): 0.006724884020116175, (196724115, 660778774, 0): 0.0212662274375268, (196724115, 21487230, 0): 0.024774862578456982, (196725581, 29219438, 0): 0.00881057268722467, (196725581, 29273080, 0): 0.014015048146271102, (196774436, 9500676798, 0): 0.004346809091263499, (252778554, 2476048346, 0): 0.004346809091263499, (262484667, 3352479286, 0): 0.008420724338232427, (262484667, 29215058, 0): 0.00806986082413941, (267214998, 29788955, 0): 0.014755760009356361, (268466879, 11806567841, 0): 0.08373942536353358, (269476781, 269476859, 0): 0.004346809091263499, (269476782, 269476859, 0): 0.013098904526139332, (269476782, 29215057, 0): 0.015554949124790458, (269476782, 21487232, 0): 0.030193754629449145, (269476859, 269476781, 0): 0.004405286343612335, (269476859, 269476782, 0): 0.012923472769092822, (269476859, 9847831441, 0): 0.004405286343612335, (271872441,

29273056, 0): 0.004346809091263499, (271872510, 9787291407, 0):
0.004405286343612335, (271872510, 1289066093, 0):
0.004405286343612335, (271872510, 10593327815, 0):
0.0002534014268449573, (271872510, 29273049, 0): 0.02746481618650345,
(283251475, 164641499, 0): 0.006198588748976649, (283251475, 29217293,
0): 0.03489142723480566 4, (287447524, 29215046, 0):
0.08093251725078944, (287447524, 287447547, 0): 0.009122451366418463,
(287447547, 287447548, 0): 0.06461736384546411, (287447548, 29215047,
0): 0.07323301235819267, (287447548, 26517409, 0):
0.004366301508713111, (287447549, 287447524, 0): 0.09011344586955675,
(287447549, 29217276, 0): 0.03457954855561187, (292682410, 3352479272,
0): 0.05555338973139449, (292682410, 29215067, 0):
0.006237573583875873, (318545589, 29276746, 0): 0.030213247046898758,
(318545591, 318545589, 0): 0.0020272114147596586, (318545591,
88071064, 0): 0.03411173053682118, (318545591, 536286819, 0):
0.028634361233480177, (318545593, 29276961, 0): 0.027113952672410433,
(318545593, 27555214, 0): 0.012514132002650969, (318545595, 29276959,
0): 0.004346809091263499, (318549151, 27555224, 0):
0.007036762699309969, (318549151, 353457764, 0): 0.004405286343612335,
(318549151, 29276750, 0): 0.039062804569022655, (338894718, 28794519,
0): 0.006549452263069666, (338894718, 1491865195, 0):
0.0033721882187828936, (353457764, 318549151, 0):
0.004346809091263499, (357477858, 3242035266, 0):
0.0007212194456356478, (357477858, 29275853, 0): 0.02877080815562746,
(357477858, 2814705317, 0): 0.001325484386573623, (359541354,
29217348, 0): 0.004346809091263499, (412194367, 2757332331, 0):
0.004346809091263499, (442649842, 173953089, 0): 0.008771587852325445,
(442649842, 1539712867, 0): 0.0331760944992398, (442649846,
8078047384, 0): 0.0005068028536899146, (442649846, 29788946, 0):
0.038750925889828855, (536286819, 3900230865, 0): 0.02265018907644926,
(536286819, 29276750, 0): 0.03272776889789872, (536286819, 318545591,
0): 0.03374137460527855, (536286819, 11391147437, 0):
0.05839928267903786, (611862495, 292682410, 0): 0.041323924993177656,
(611862495, 28794519, 0): 0.03393629877977467, (611862495, 338894718,
0): 0.003937468324821645, (612417324, 28794539, 0):
0.004346809091263499, (660778774, 29215073, 0): 0.009843670812054112,
(660778774, 196725581, 0): 0.0146582979221083, (848523542, 28794517,
0): 0.004346809091263499, (1237777024, 5450916417, 0):
0.004346809091263499, (1285251472, 29218294, 0): 0.02968695177575923,
(1285251472, 29217276, 0): 0.041109508401231926, (1285251472,
1285251502, 0): 0.0019882265798604345, (1285251502, 29217277, 0):
0.002300105259054228, (1285251502, 1285251472, 0):
0.0016568554832170287, (1285251502, 29217276, 0): 0.005107013371798 37,
(1289066093, 271872510, 0): 0.004346809091263499, (1491865195,
3352479275, 0): 0.007095239951658805, (1491865195, 21487230, 0):
0.0039959455771704806, (1491865195, 338894718, 0):
0.0030408171221394877, (1491865195, 12036921938, 0):
0.004405286343612335, (1539712867, 2677649274, 0):
0.01502865385365093, (1539712867, 27555221, 0): 0.024813847413356207,

(1814668312, 1814668313, 0): 1.94924174496121e-05, (1814668312, 26517409, 0): 0.008732603017426222, (1814668313, 1814668312, 0): 0.004385793926162723, (1814668480, 292682410, 0): 0.013800631554325368, (1814668480, 611862495, 0): 0.041811235429417955, (1969136398, 5984003377, 0): 0.006179096331527036, (1969136398, 29218288, 0): 0.026022377295232154, (1984006630, 29273058, 0): 0.020330591399945423, (1984006630, 29273064, 0): 0.0281860356321391, (2028472686, 29785151, 0): 0.02169506062141827, (2143522428, 29276737, 0): 0.02726989201200733, (2143522428, 29218326, 0): 0.007192702038906865, (2143522428, 29276730, 0): 0.004775642275154965, (2476048346, 29273067, 0): 0.0085376788429301, (2476048346, 252778554, 0): 0.004405286343612335, (2476048346, 29273068, 0): 0.0016373630657674164, (2479455794, 29273067, 0): 0.012377685080503685, (2479455794, 2814705317, 0): 0.015925305056333088, (2625899861, 2891066381, 0): 0.004405286343612335, (2625899861, 29218321, 0): 0.08026977505750263, (2625899861, 26960761, 0): 0.08270632723870415, (2677649274, 29217325, 0): 0.05145998206697595, (2757332331, 2757332333, 0): 0.008674125765077385, (2757332333, 2757332331, 0): 0.004385793926162723, (2757332333, 412194367, 0): 0.004405286343612335, (2757332333, 29273071, 0): 0.012923472769092822, (2814705317, 2479455794, 0): 0.016315153405325328, (2814705317, 29273069, 0): 0.018498304159681886, (2814705317, 357477858, 0): 0.001344976804023235, (2891066381, 2625899861, 0): 0.004346809091263499, (3242035261, 29275853, 0): 0.031967564617363844, (3242035261, 3242035266, 0): 0.029043701999922032, (3242035262, 3242035261, 0): 0.039647577092511016, (3242035262, 2028472686, 0): 0.021714553038867882, (3242035266, 357477858, 0): 0.009044481696620015, (3242035266, 10254148068, 0): 0.020661962496588828, (3243910627, 29276213, 0): 0.14395150286538538, (3243910627, 29276217, 0): 0.03822463061868933, (3243910627, 29276687, 0): 0.07822307122529336, (3352479272, 3352479275, 0): 0.001929749327511598, (3352479272, 28373656, 0): 0.06834041557834003, (3352479275, 1491865195, 0): 0.007563057970449496, (3352479275, 3352479272, 0): 0.014775252426805973, (3352479275, 1814668480, 0): 0.05567034423609216, (3352479286, 3352479290, 0): 0.008654633347627774, (3352479286, 262484667, 0): 0.007855444232193677, (3352479290, 3463840860, 0): 0.029277611009317375, (3352479290, 28794519, 0): 0.03296167790729406, (3411530669, 10312836355, 0): 0.02781567970059647, (3411530669, 26960761, 0): 0.019414447779813652, (3463840860, 28794518, 0): 0.020934856340883396, (3463840860, 3352479290, 0): 0.020700947331488053, (3463840860, 11759269145, 0): 0.004405286343612335, (3900230865, 536286819, 0): 0.029842891115356127, (3900230865, 318545589, 0): 0.028244512884487934, (3900230865, 29276959, 0): 0.0074461034657518224, (5450916417, 29218299, 0): 0.009960625316751784, (5450916417, 29218319, 0): 0.005126505789247983, (5450916417, 1237777024, 0): 0.004405286343612335, (5984003368,

```
29217269, 0): 0.014502358582511404, (5984003368, 5984003369, 0):
0.0043078242563642745, (5984003368, 21487231, 0):
0.024151105220069392, (5984003369, 5984003368, 0):
0.011675958052317649, (5984003369, 29788955, 0): 0.005828232817434018,
(5984003369, 5984003376, 0): 0.0073486413785037626, (5984003376,
5984003377, 0): 0.02134419710732525, (5984003376, 29217269, 0):
0.010720829597286655, (5984003377, 1969136398, 0):
0.01929749327511598, (5984003377, 29216575, 0): 0.013137889361038557,
(5984003377, 5984003376, 0): 0.01132509453822463, (8078047384,
442649846, 0): 0.01216326848855795, (8078047384, 29218287, 0):
0.021285719854976414, (9500676798, 10593327815, 0):
0.019550894701960937, (9500676798, 196774436, 0):
0.004405286343612335, (9787291407, 271872510, 0):
0.004346809091263499, (9847831441, 269476859, 0):
0.004346809091263499, (10254148068, 357477858, 0):
0.02048653073954232, (10254148068, 28373648, 0): 0.016646524501968733,
(10312836355, 26960758, 0): 0.010954738606682002, (10312836355,
1984006630, 0): 0.036782191727418034, (10593327815, 271872510, 0):
0.019745818876457058, (11391147437, 29276752, 0):
0.058886593115278156, (11391147437, 29276743, 0): 0.00417137733421699,
(11391147437, 536286819, 0): 0.036489805465673854, (11759269145,
3463840860, 0): 0.004346809091263499, (11806567841, 29217327, 0):
0.08368094811118475, (12036921938, 1491865195, 0):
0.004346809091263499}
Closeness centrality: {21487224: np.float64(0.41984099504456124),
21487230: np.float64(0.46685952299321387), 21487231:
np.float64(0.4645543390478923), 21487232:
np.float64(0.5402815922889471), 26517409:
np.float64(0.48007257135271403), 26960758:
np.float64(0.47888760747542397), 26960761:
np.float64(0.5516755754847817), 26960762:
np.float64(0.5371807364170812), 27555211:
np.float64(0.38215074534797144), 27555214:
np.float64(0.39908085178502783), 27555215:
np.float64(0.3879167295987945), 27555221:
np.float64(0.45150484294149157), 27555224:
np.float64(0.3851690935358708), 27555225:
np.float64(0.3709823272712703), 27555226:
np.float64(0.3546366933698373), 27555227:
np.float64(0.37711990897363745), 28373648:
np.float64(0.38068925761928957), 28373656:
np.float64(0.516159837854256), 28794517:
np.float64(0.3980102029756231), 28794518:
np.float64(0.4157544847371475), 28794519:
np.float64(0.4467797195451137), 28794539:
np.float64(0.428974549598826), 28794542:
np.float64(0.5364163919943234), 29215046:
np.float64(0.5086096550389363), 29215047:
np.float64(0.49038712241274757), 29215049:
np.float64(0.5135187127959437), 29215057:
```

```
np.float64(0.4525575338008259), 29215058:
np.float64(0.40307675603479953), 29215060:
np.float64(0.38957592193691987), 29215063:
np.float64(0.3804274119801617), 29215067:
np.float64(0.4479366106032963), 29215071:
np.float64(0.3775463398730089), 29215073:
np.float64(0.4335861707299352), 29216571:
np.float64(0.3803426188672338), 29216572:
np.float64(0.3654629051988408), 29216575:
np.float64(0.3824862305453355), 29217269:
np.float64(0.3964516764355784), 29217276:
np.float64(0.485387149180984), 29217277:
np.float64(0.4263857520949245), 29217280:
np.float64(0.48269601395694006), 29217293:
np.float64(0.4905630954850368), 29217303: 0.0, 29217317:
np.float64(0.5188951919833829), 29217319:
np.float64(0.3907951080805852), 29217321:
np.float64(0.46244020463210744), 29217322:
np.float64(0.46935962474156656), 29217325:
np.float64(0.44768019246145413), 29217327:
np.float64(0.44396914497894285), 29217332:
np.float64(0.4608883069388377), 29217340:
np.float64(0.45429687907025446), 29217342:
np.float64(0.43946820944691023), 29217348:
np.float64(0.4634909117913195), 29218287:
np.float64(0.4057511887883831), 29218288:
np.float64(0.4050699794145385), 29218289:
np.float64(0.47801577571732157), 29218291:
np.float64(0.47979551122865755), 29218293:
np.float64(0.5272363209807451), 29218294:
np.float64(0.4765295388658766), 29218295:
np.float64(0.478380249564905), 29218296:
np.float64(0.48846227807861897), 29218299:
np.float64(0.5419651479652865), 29218302:
np.float64(0.5558211010207639), 29218305:
np.float64(0.4608124970960068), 29218313:
np.float64(0.44253626485325465), 29218315:
np.float64(0.5320923246588408), 29218316:
np.float64(0.4576684107113044), 29218318:
np.float64(0.44774876112454287), 29218319:
np.float64(0.4556593420268113), 29218321:
np.float64(0.5685083323583354), 29218322:
np.float64(0.5386858397735191), 29218323:
np.float64(0.5646212775430112), 29218324:
np.float64(0.5710946633189322), 29218325:
np.float64(0.5430128481976494), 29218326:
np.float64(0.5506609258803631), 29218327:
np.float64(0.5355473625799027), 29218328:
np.float64(0.4796532511672164), 29219437:
```

```
np.float64(0.42381155460210246), 29219438:
np.float64(0.4266534182435442), 29271261:
np.float64(0.4758373282966664), 29273048:
np.float64(0.39548500648160667), 29273049:
np.float64(0.34700340901218457), 29273050:
np.float64(0.4099897819467548), 29273056:
np.float64(0.42328377894671887), 29273057:
np.float64(0.445704143233606), 29273058:
np.float64(0.47987690143259115), 29273064:
np.float64(0.5118273912933673), 29273065:
np.float64(0.5050149455396253), 29273067:
np.float64(0.3956863079462114), 29273068:
np.float64(0.39140293184945557), 29273069:
np.float64(0.4880111267155593), 29273071:
np.float64(0.35883402848255874), 29273075:
np.float64(0.33062759665956265), 29273076:
np.float64(0.3710760065298192), 29273077:
np.float64(0.3459784336079882), 29273078:
np.float64(0.33074405971792237), 29273079:
np.float64(0.37044205815227116), 29273080:
np.float64(0.47261632237711937), 29275853:
np.float64(0.4234537879234122), 29275983:
np.float64(0.368557868644032), 29276209:
np.float64(0.543335501948312), 29276210:
np.float64(0.487642799954486), 29276211:
np.float64(0.4521886263439961), 29276213:
np.float64(0.5552859357377805), 29276217:
np.float64(0.5705846612251579), 29276222:
np.float64(0.5069619154973874), 29276687:
np.float64(0.5124921755804637), 29276688:
np.float64(0.480812227132911), 29276689:
np.float64(0.45217444336521856), 29276690:
np.float64(0.4065115266758454), 29276726:
np.float64(0.4457134140416055), 29276728:
np.float64(0.43833040923504313), 29276730:
np.float64(0.4606116040198484), 29276737:
np.float64(0.46614909580512015), 29276743:
np.float64(0.46115176767229726), 29276745:
np.float64(0.44706591761574394), 29276746:
np.float64(0.4547896191029137), 29276750:
np.float64(0.45006975407253735), 29276752:
np.float64(0.47291210192071703), 29276753:
np.float64(0.4717235123136922), 29276755:
np.float64(0.4816344708985704), 29276757:
np.float64(0.45080271613890865), 29276759:
np.float64(0.44283243473849665), 29276762:
np.float64(0.4205619118536586), 29276764:
np.float64(0.4138825941362839), 29276773:
np.float64(0.38504320501443345), 29276775:
```

```
np.float64(0.34145182315487965), 29276777:
np.float64(0.3681009436972255), 29276778:
np.float64(0.3737190511355788), 29276959:
np.float64(0.4477986263569066), 29276961:
np.float64(0.44021236083338044), 29785137:
np.float64(0.34740583799081265), 29785151:
np.float64(0.38517735540800885), 29787489:
np.float64(0.31130127732816276), 29788946:
np.float64(0.42663712792371655), 29788955:
np.float64(0.4425607880343457), 32248754:
np.float64(0.3753423288057022), 60237352:
np.float64(0.3604671142376287), 88070313:
np.float64(0.4402704348411617), 88071064:
np.float64(0.4430703585038269), 88071065:
np.float64(0.45265336497478237), 88071069:
np.float64(0.4506121013906984), 122087860:
np.float64(0.38490607485704725), 164641499:
np.float64(0.4532128680312187), 173953089:
np.float64(0.42803384955883406), 196724115:
np.float64(0.44091872675883914), 196725581:
np.float64(0.4236628181149004), 196774436:
np.float64(0.3664290216979125), 252778554:
np.float64(0.3326552452282859), 262484667:
np.float64(0.40199227615721167), 267214998:
np.float64(0.45287388896226893), 268466879:
np.float64(0.4481112104090986), 269476781:
np.float64(0.46686055260734693), 269476782:
np.float64(0.5188215905729702), 269476859:
np.float64(0.4776895983487515), 271872441:
np.float64(0.3959889240466515), 271872510:
np.float64(0.3431116145835074), 283251475:
np.float64(0.4624770036041907), 287447524:
np.float64(0.48431250641186613), 287447547:
np.float64(0.506189707371374), 287447548:
np.float64(0.4974104742214936), 287447549:
np.float64(0.49305481302931964), 292682410:
np.float64(0.4832101814695585), 318545589:
np.float64(0.44813767048190456), 318545591:
np.float64(0.4471214953628339), 318545593:
np.float64(0.44478000903620174), 318545595:
np.float64(0.43688783291183914), 318549151:
np.float64(0.38505705514753824), 338894718:
np.float64(0.4400020555140973), 353457764:
np.float64(0.3822580291112071), 357477858:
np.float64(0.4110667959888539), 359541354:
np.float64(0.45261200757849945), 412194367:
np.float64(0.3433696040175106), 442649842:
np.float64(0.4394687737495094), 442649846:
np.float64(0.4364688985812821), 536286819:
```

```
np.float64(0.4522646523608941), 611862495:
np.float64(0.47595408995540556), 612417324:
np.float64(0.39006637923679577), 660778774:
np.float64(0.4344930939807018), 848523542:
np.float64(0.3704419956635258), 1237777024:
np.float64(0.3808128390238446), 1285251472:
np.float64(0.481320213610953), 1285251502:
np.float64(0.47761930711077516), 1289066093:
np.float64(0.3105596140342911), 1491865195:
np.float64(0.45477653840587984), 1539712867:
np.float64(0.434412337692812), 1814668312:
np.float64(0.005572849700562319), 1814668313:
np.float64(0.005572849700562319), 1814668480:
np.float64(0.4850904174018197), 1969136398:
np.float64(0.40057308304180816), 1984006630:
np.float64(0.4786276809732615), 2028472686:
np.float64(0.3870460245750708), 2143522428:
np.float64(0.47334496076060034), 2476048346:
np.float64(0.393399879136504), 2479455794:
np.float64(0.4260132507949867), 2625899861:
np.float64(0.5590363680667929), 2677649274:
np.float64(0.4429742861163573), 2757332331:
np.float64(0.3456080303847875), 2757332333:
np.float64(0.3491265038604421), 2814705317:
np.float64(0.4422039730651418), 2891066381:
np.float64(0.4757960449000576), 3242035261:
np.float64(0.4254915035912275), 3242035262:
np.float64(0.43042988664870435), 3242035266:
np.float64(0.4133898452365329), 3243910627:
np.float64(0.5685269959300002), 3352479272:
np.float64(0.5071143133987787), 3352479275:
np.float64(0.5095543825797764), 3352479286:
np.float64(0.40037900266145054), 3352479290:
np.float64(0.41967457220487453), 3411530669:
np.float64(0.47830034802200716), 3463840860:
np.float64(0.4182304559223407), 3900230865:
np.float64(0.4510809824929084), 5450916417:
np.float64(0.45616157957631254), 5984003368:
np.float64(0.44388022179201325), 5984003369:
np.float64(0.44009823685160504), 5984003376:
np.float64(0.4026960745880053), 5984003377:
np.float64(0.40352514924919214), 8078047384:
np.float64(0.44133265964223456), 9500676798:
np.float64(0.3715692559678657), 9787291407:
np.float64(0.3048293151288482), 9847831441:
np.float64(0.4312369823764402), 10254148068:
np.float64(0.4078992083007357), 10312836355:
np.float64(0.47707983814777133), 10593327815:
np.float64(0.35420960351158015), 11391147437:
```

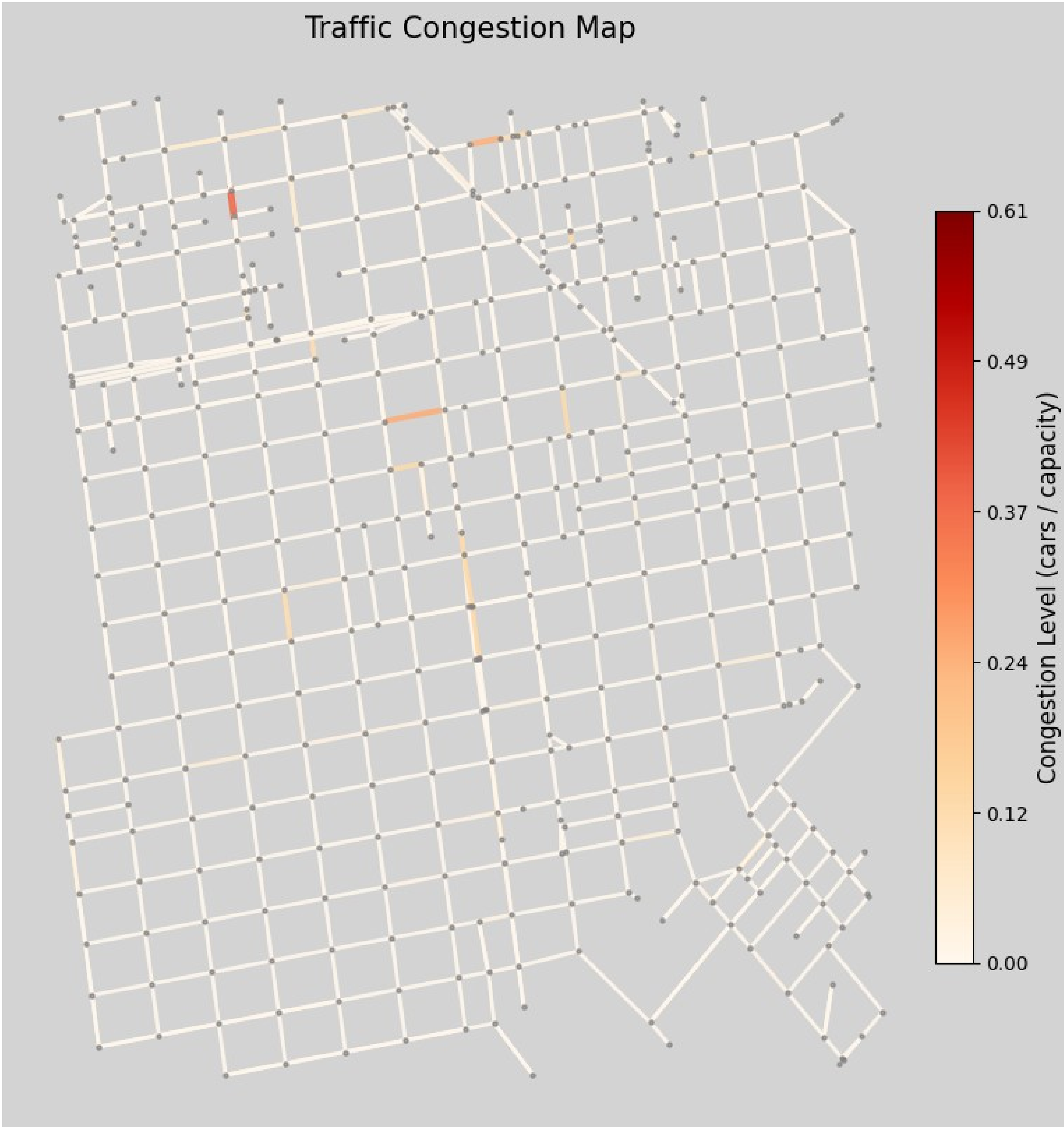
```
np.float64(0.46299088081721684), 11759269145:
np.float64(0.3691163596479905), 11806567841:
np.float64(0.444526313922846), 12036921938:
np.float64(0.4242474179532874)}
Degree centrality: {21487224: 0.01327433628318584, 21487230:
0.035398230088495575, 21487231: 0.017699115044247787, 21487232:
0.035398230088495575, 26517409: 0.01327433628318584, 26960758:
0.01327433628318584, 26960761: 0.035398230088495575, 26960762:
0.035398230088495575, 27555211: 0.01327433628318584, 27555214:
0.008849557522123894, 27555215: 0.01327433628318584, 27555221:
0.022123893805309734, 27555224: 0.02654867256637168, 27555225:
0.017699115044247787, 27555226: 0.01327433628318584, 27555227:
0.017699115044247787, 28373648: 0.017699115044247787, 28373656:
0.017699115044247787, 28794517: 0.02654867256637168, 28794518:
0.022123893805309734, 28794519: 0.02654867256637168, 28794539:
0.022123893805309734, 28794542: 0.022123893805309734, 29215046:
0.017699115044247787, 29215047: 0.017699115044247787, 29215049:
0.02654867256637168, 29215057: 0.02654867256637168, 29215058:
0.02654867256637168, 29215060: 0.017699115044247787, 29215063:
0.008849557522123894, 29215067: 0.02654867256637168, 29215071:
0.008849557522123894, 29215073: 0.02654867256637168, 29216571:
0.01327433628318584, 29216572: 0.008849557522123894, 29216575:
0.01327433628318584, 29217269: 0.017699115044247787, 29217276:
0.02654867256637168, 29217277: 0.017699115044247787, 29217280:
0.02654867256637168, 29217293: 0.02654867256637168, 29217303:
0.004424778761061947, 29217317: 0.035398230088495575, 29217319:
0.008849557522123894, 29217321: 0.01327433628318584, 29217322:
0.017699115044247787, 29217325: 0.01327433628318584, 29217327:
0.01327433628318584, 29217332: 0.01327433628318584, 29217340:
0.01327433628318584, 29217342: 0.017699115044247787, 29217348:
0.017699115044247787, 29218287: 0.022123893805309734, 29218288:
0.022123893805309734, 29218289: 0.035398230088495575, 29218291:
0.02654867256637168, 29218293: 0.02654867256637168, 29218294:
0.02654867256637168, 29218295: 0.030973451327433628, 29218296:
0.030973451327433628, 29218299: 0.035398230088495575, 29218302:
0.030973451327433628, 29218305: 0.02654867256637168, 29218313:
0.008849557522123894, 29218315: 0.02654867256637168, 29218316:
0.017699115044247787, 29218318: 0.022123893805309734, 29218319:
0.022123893805309734, 29218321: 0.02654867256637168, 29218322:
0.035398230088495575, 29218323: 0.02654867256637168, 29218324:
0.02654867256637168, 29218325: 0.022123893805309734, 29218326:
0.035398230088495575, 29218327: 0.035398230088495575, 29218328:
0.02654867256637168, 29219437: 0.008849557522123894, 29219438:
0.01327433628318584, 29271261: 0.01327433628318584, 29273048:
0.02654867256637168, 29273049: 0.017699115044247787, 29273050:
0.01327433628318584, 29273056: 0.030973451327433628, 29273057:
0.030973451327433628, 29273058: 0.02654867256637168, 29273064:
0.02654867256637168, 29273065: 0.02654867256637168, 29273067:
0.017699115044247787, 29273068: 0.02654867256637168, 29273069:
```

```
0.02654867256637168, 29273071: 0.02654867256637168, 29273075:
0.017699115044247787, 29273076: 0.02654867256637168, 29273077:
0.02654867256637168, 29273078: 0.008849557522123894, 29273079:
0.017699115044247787, 29273080: 0.017699115044247787, 29275853:
0.017699115044247787, 29275983: 0.017699115044247787, 29276209:
0.02654867256637168, 29276210: 0.008849557522123894, 29276211:
0.017699115044247787, 29276213: 0.017699115044247787, 29276217:
0.02654867256637168, 29276222: 0.017699115044247787, 29276687:
0.02654867256637168, 29276688: 0.02654867256637168, 29276689:
0.02654867256637168, 29276690: 0.017699115044247787, 29276726:
0.02654867256637168, 29276728: 0.02654867256637168, 29276730:
0.030973451327433628, 29276737: 0.017699115044247787, 29276743:
0.035398230088495575, 29276745: 0.035398230088495575, 29276746:
0.022123893805309734, 29276750: 0.02654867256637168, 29276752:
0.035398230088495575, 29276753: 0.035398230088495575, 29276755:
0.035398230088495575, 29276757: 0.035398230088495575, 29276759:
0.035398230088495575, 29276762: 0.035398230088495575, 29276764:
0.02654867256637168, 29276773: 0.02654867256637168, 29276775:
0.01327433628318584, 29276777: 0.02654867256637168, 29276778:
0.02654867256637168, 29276959: 0.02654867256637168, 29276961:
0.017699115044247787, 29785137: 0.02654867256637168, 29785151:
0.022123893805309734, 29787489: 0.008849557522123894, 29788946:
0.008849557522123894, 29788955: 0.02654867256637168, 32248754:
0.008849557522123894, 60237352: 0.017699115044247787, 88070313:
0.022123893805309734, 88071064: 0.02654867256637168, 88071065:
0.02654867256637168, 88071069: 0.02654867256637168, 122087860:
0.022123893805309734, 164641499: 0.022123893805309734, 173953089:
0.017699115044247787, 196724115: 0.017699115044247787, 196725581:
0.017699115044247787, 196774436: 0.008849557522123894, 252778554:
0.008849557522123894, 262484667: 0.017699115044247787, 267214998:
0.01327433628318584, 268466879: 0.008849557522123894, 269476781:
0.008849557522123894, 269476782: 0.02654867256637168, 269476859:
0.02654867256637168, 271872441: 0.008849557522123894, 271872510:
0.035398230088495575, 283251475: 0.01327433628318584, 287447524:
0.01327433628318584, 287447547: 0.01327433628318584, 287447548:
0.017699115044247787, 287447549: 0.017699115044247787, 292682410:
0.022123893805309734, 318545589: 0.01327433628318584, 318545591:
0.022123893805309734, 318545593: 0.017699115044247787, 318545595:
0.008849557522123894, 318549151: 0.02654867256637168, 338894718:
0.022123893805309734, 353457764: 0.008849557522123894, 357477858:
0.02654867256637168, 359541354: 0.008849557522123894, 412194367:
0.008849557522123894, 442649842: 0.01327433628318584, 442649846:
0.017699115044247787, 536286819: 0.035398230088495575, 611862495:
0.022123893805309734, 612417324: 0.008849557522123894, 660778774:
0.017699115044247787, 848523542: 0.008849557522123894, 1237777024:
0.008849557522123894, 1285251472: 0.02654867256637168, 1285251502:
0.02654867256637168, 1289066093: 0.008849557522123894, 1491865195:
0.035398230088495575, 1539712867: 0.017699115044247787, 1814668312:
0.01327433628318584, 1814668313: 0.008849557522123894, 1814668480:
```

```
0.01327433628318584, 1969136398: 0.017699115044247787, 1984006630:
0.017699115044247787, 2028472686: 0.01327433628318584, 2143522428:
0.022123893805309734, 2476048346: 0.02654867256637168, 2479455794:
0.017699115044247787, 2625899861: 0.02654867256637168, 2677649274:
0.01327433628318584, 2757332331: 0.01327433628318584, 2757332333:
0.022123893805309734, 2814705317: 0.02654867256637168, 2891066381:
0.008849557522123894, 3242035261: 0.017699115044247787, 3242035262:
0.017699115044247787, 3242035266: 0.017699115044247787, 3243910627:
0.02654867256637168, 3352479272: 0.017699115044247787, 3352479275:
0.02654867256637168, 3352479286: 0.017699115044247787, 3352479290:
0.022123893805309734, 3411530669: 0.017699115044247787, 3463840860:
0.02654867256637168, 3900230865: 0.02654867256637168, 5450916417:
0.02654867256637168, 5984003368: 0.02654867256637168, 5984003369:
0.02654867256637168, 5984003376: 0.022123893805309734, 5984003377:
0.02654867256637168, 8078047384: 0.017699115044247787, 9500676798:
0.017699115044247787, 9787291407: 0.008849557522123894, 9847831441:
0.008849557522123894, 10254148068: 0.017699115044247787, 10312836355:
0.017699115044247787, 10593327815: 0.01327433628318584, 11391147437:
0.02654867256637168, 11759269145: 0.008849557522123894, 11806567841:
0.008849557522123894, 12036921938: 0.008849557522123894}
```

```
### My address in SF
sf_network = RoadNetwork('California 851, San Francisco, United
States')
simulator = TrafficSimulator(sf_network)
simulator.simulate(num_steps = 100)
simulator.plot_traffic()
```

```
<ipython-input-6-5b8c658043ff>:220: MatplotlibDeprecationWarning: The
get_cmap function was deprecated in Matplotlib 3.7 and will be removed
in 3.11. Use ``matplotlib.colormaps[name]`` or
``matplotlib.colormaps.get_cmap()`` or ``pyplot.get_cmap()`` instead.
  cmap = cm.get_cmap("OrRd")
```




```
(<Figure size 1000x1000 with 2 Axes>,
 <Axes: title={'center': 'Traffic Congestion Map'}>)
```