PUBLIC TRANSPORT OPTIMIZING

Optimizing public transport involves various aspects such as route planning, scheduling, and efficiency improvements. Here's a basic example of Python code for optimizing public transport routes using a simple graph representation and Dijkstra's algorithm for finding the shortest paths:

```
import heapq



class Graph:
    def _init_(self):
        self.nodes = set()
        self.edges = {}



    def add_node(self, value):
        self.nodes.add(value)
        self.edges[value] = []



    def add_edge(self, from_node, to_node, weight):
```

```
        self.edges[from_node].append((to_node, weight))
        self.edges[to_node].append((from_node, weight))


def dijkstra(graph, start):
    shortest_path = {node: float('infinity') for node in graph.nodes}
    shortest_path[start] = 0
    priority_queue = [(0, start)]


    while priority_queue:
        current_distance, current_node = heapq.heappop(priority_queue)


        if current_distance > shortest_path[current_node]:
            continue


        for neighbor, weight in graph.edges[current_node]:
         …
```

In this example, the Graph class represents the public transport network, and the dijkstra function calculates the shortest paths from a specified starting node. You can modify the Graph class and the edges between nodes to represent your specific public transport network and distances between stops.