Code

```
`pyth

port networkx as

port matplotlib.pyplot as p

port hea


ass Grap

f __init__(self

lf.G = nx.DiGraph


f add_node(self, node_id

lf.G.add_node(node_i


f add_edge(self, node1, node2, weight

lf.G.add_edge(node1, node2, weight=weigh


f dijkstra(self, start_node, end_node

stances = {node: float('inf') for node in self.G.nodes(

evious_nodes = {node: None for node in self.G.nodes(

stances[start_node] =

iority_queue = [(0, start_node
```

```
ile priority_queu

rrent_distance, current_node = heapq.heappop(priority_queu


 current_distance > distances[current_node

ntin


r neighbor in self.G.neighbors(current_node

ight = self.G[current_node][neighbor]['weight

stance = current_distance + weig


 distance < distances[neighbor

stances[neighbor] = distan

evious_nodes[neighbor] = current_no

apq.heappush(priority_queue, (distance, neighbor


th =

rrent_node = end_no

ile current_node is not Non

th.append(current_nod

rrent_node = previous_nodes[current_nod
```

```
th.reverse


turn distances[end_node], pa


f visualize(self, path

s = nx.spring_layout(self.

.draw(self.G, pos, with_labels=True, node_color='lightblue

.draw_networkx_edges(self.G, pos, edgelist=[(path[i], path[i+1]) for i in range(len(p

t.show


ample usa

aph = Graph

aph.add_node(

aph.add_node(

aph.add_node(

aph.add_edge(1, 2,

aph.add_edge(1, 3,

aph.add_edge(2, 3,


stance, path = graph.dijkstra(1,
```

int(f"Shortest distance: {distance}

int(f"Shortest path: {path}

aph.visualize(pat