```
import networkx as nx
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm
from os import path
from time import sleep
# หาตำแหน่งของผู้ใช้งาน
find_current_path = path.dirname(path.abspath(__file__))
# โหลดฟอนต์ภาษาไทย
font_prop_bold =
···fm.FontProperties(fname=f"{find_current_path}/fonts/Sarabun-Bold.ttf")
font_prop_regular =
···fm.FontProperties(fname=f"{find_current_path}/fonts/Sarabun-
···Regular.ttf")
class Graph:
    def __init__(self):
        self.line_spacing_egde = 0.0345
        self.line_spacing_node = 0.045
        self.top_graph = 0.97
        self.bottom_graph = 0.03
        self.legend_y = 0.995
        # กำหนดตำแหน่งโหนดเอง
        self.pos = {1: (0, 1), 2: (0, 3), 3: (1, 1), 4: (2, 3), 5: (2,
···1), 6: (2, 2), 7: (3, 1), 8: (3, 3), 9: (4, 2), 10: (5, 0), 11: (5, 1),
···12: (5, 2), 13: (5, 3), 14: (6, 2), 15: (6, 3), 16: (7, 0), 17: (7, 2),
···20: (6, -1), 19: (5, -1), 18: (3, -1)}
        # (โหนดต้น, โหนดปลาย, ระยะทาง)
        self.routes = [(13, 15, 5.8), (15, 17, 1.9), (17, 16, 0.14),
···(16, 17, 0.14), (16, 5, 23), (5, 3, 10), (3, 1, 0.18), (1, 3, 0.18),
···(16, 9, 12.8), (16, 10, 2.5), (10, 7, 7.1), (14, 11, 0.04), (11, 14,
···0.04), (11, 12, 37), (7, 11, 34.1), (11, 7, 34.1), (9, 7, 0.6), (9, 6,
···0.29), (9, 8, 0.21), (8, 7, 0.45), (8, 6, 0.3), (8, 9, 0.21), (6, 9,
···0.29), (6, 8, 0.3), (6, 7, 0.26), (7, 6, 0.26), (7, 8, 0.45), (7, 9,
···0.6), (8, 17, 13.2), (6, 5, 9.65), (1, 2, 3.5), (2, 4, 0.06), (4, 6,
···17.05), (8, 12, 13.85), (13, 8, 12.75), (12, 13, 0.05), (13, 12, 0.05),
···(18, 19, 0.4), (19, 18, 0.4), (19, 20, 0.11), (20, 19, 0.11), (19, 16,
···0.95), (17, 20, 1)]
        self.node_labels = {1: "มจธ.", 2: "ตรงข้ามตลาดบางประกอก", 3:
···"ตรงข้าม มจธ.", 4: "ตลาดบางประกอก", 5: "ตลาดวงเวียนใหญ่", 6:
···"อนุสาวรีย์ชัยสมรภูมิ (เกาะพญาไท)", 7: "อนุสาวรีย์ชัยสมรภูมิ
···(เกาะดินแดง)", 8: "อนุสาวรีย์ชัยสมรภูมิ (เกาะพหลโยธิน)", 9:
···"อนุสาวรีย์ชัยสมรภูมิ (เกาะราชวิถี)", 10: "สถานีดับเพลิงบางโพ", 11:
···"ท่ารถตผู้ สจล.", 12: "หมอชิต 2", 13: "สถานีขนส่งหมอชิต 2 (หน้าเสาธง)",
···14: "สจล.", 15: "ตรงข้ามวัดน้อย", 16: "มจพ.", 17: "มทร.พระนคร
···วิทนาเขตพระนครเหนือ", 18: "หอพักชมรมมุสลิม มจพ.", 19:
···"ตรงข้ามโรงเรียนสตรีนนทบุรี", 20: "โรงเรียนสตรีนนทบุรี"}
        self.edge_labels = {(13, 15): '1', (15, 17): '2', (17, 16): '3',
···(16, 5): '4', (5, 3): '5', (3, 1): '6', (16, 9): '7', (16, 10): '8',
···(10, 7): '9', (14, 11): '10', (11, 12): '11', (7, 11): '12', (9, 7):
···'13', (9, 6): '14', (9, 8): '15', (8, 7): '16', (8, 6): '17', (6, 7):
···'18', (8, 17): '19', (6, 5): '20', (1, 2): '21', (2, 4): '22', (4, 6):
···'23', (8, 12): '24', (13, 8): '25', (12, 13): '26', (18, 19): '27', (19,
···20): '28', (19, 16): '29', (17, 20): '30'}
        # ระยะทางจริงที่ต้องแสดงใน Legend
        self.distance_mapping = {'1': 'สายรถ 5 (5.8 km)', '2': 'สายรถ
···543ก/1-34 (1.9 km)', '3': 'เดิน (0.14 km)', '4': 'สายรถ 175/2-22 (23
```

```
…km)', '5': 'สายรถ 21/4-6 (10 km)', '6': 'เดิน (0.18 km)', '7': 'สายรถ
…97/2-15 (12.8 km)', '8': 'สายรถ 33/2-6 (2.5 km)', '9': 'สายรถ 3-55 (7.1
…km)', '10': 'เดิน (0.04 km)', '11': 'รถมินิบัส (37 km)', '12': 'สายรถ
…ต.156 (34.1 km)', '13': 'เดิน (0.6 km)', '14': 'เดิน (0.29 km)', '15':
…'เดิน (0.21 km)', '16': 'เดิน (0.45 km)', '17': 'เดิน (0.3 km)', '18':
…'เดิน (0.26 km)', '19': 'สายรถ 97/2-15 (13.2 km)', '20': 'สายรถ 529/4-28
…(9.65 km)', '21': 'สายรถ 21/4-6 (3.5 km)', '22': 'เดิน (0.06 km)', '23':
…'สายรถ 140/4-23E (17.05 km)', '24': 'สายรถ 140/4-23E (13.85 km)', '25':
…'สายรถ 4-33E (12.75 km)', '26': 'เดิน (0.05 km)', '27': 'เดิน (0.4 km)',
…'28': 'เดิน (0.11 km)', '29': 'สายรถ 203 (0.95 km)', '30': 'สายรถ
…90/1-27 (1 km)'}
        self.G = nx.DiGraph()
        for start, end, distance in self.routes:
            self.G.add_edge(start, end, weight=distance)
        self.source = None
        self.target = None
    def draw_graph(self):
        # ใช้ subplot เพื่อควบคุมตำแหน่ง legend
        fig, ax = plt.subplots(figsize=(16, 10))
        plt.subplots_adjust(left=0, right=0.6, top=self.top_graph,
…bottom=self.bottom_graph)
        nx.draw(self.G, self.pos, with_labels=True,
…node_color="lightblue", edge_color='gray',
                node_size=1000, font_size=12, arrows=True, arrowsize=10,
…font_weight='bold')
        # เพิ่ม Legend แบบตัวเลข
        legend_x = 1  # เปลี่ยนจาก 1.1 เป็น 1.05 เพื่อให้เห็นชัดขึ้น
        for i, (count, label) in enumerate(self.node_labels.items()):
            ax.text(legend_x, self.legend_y - i *
…self.line_spacing_node, f"{count}",
                    fontsize=12, color='black',
                    bbox=dict(boxstyle="circle,pad=0.3", fc="lightblue",
…ec="white", lw=1),
                    transform=ax.transAxes, verticalalignment='center',
…horizontalalignment='center', fontproperties=font_prop_bold)
            ax.text(legend_x + 0.03, self.legend_y - i *
…self.line_spacing_node, label,
                    fontsize=12, fontweight='bold', color='black',
                    transform=ax.transAxes, verticalalignment='center',
…horizontalalignment='left', fontproperties=font_prop_regular)
        # วาดป้ายกำกับบนเส้นเชื่อม (พื้นหลังสีแดง)
        legend_x = 1.35
        for (n1, n2), label in self.edge_labels.items():
            if label in ["14", "16", "19", "26"]:
                # ตำแหน่งบนเส้นที่แบ่ง 1/3 และ 2/3 ของเส้น
                x, y = (self.pos[n1][0] * 1.15 + 1.85 * self.pos[n2][0])
…/ 3, (self.pos[n1][1] * 1.15 + 1.85 * self.pos[n2][1]) / 3
            else:
                # ตำแหน่งกึงกลางของเส้น
                x, y = (self.pos[n1][0] + self.pos[n2][0]) / 2,
…(self.pos[n1][1] + self.pos[n2][1]) / 2
            ax.text(x, y, label, fontsize=10, color='white',
                    bbox=dict(boxstyle="round,pad=0.3", fc="red",
…ec="black", lw=1),
                    horizontalalignment='center',
```

```
…verticalalignment='center', fontproperties=font_prop_bold)
        for i, (num, text) in enumerate(self.distance_mapping.items()):
            ax.text(legend_x, self.legend_y - i *
…self.line_spacing_egde, num,
                    fontsize=10, color='white',
                    bbox=dict(boxstyle="round,pad=0.3", fc="red",
…ec="black", lw=1),
                    transform=ax.transAxes, verticalalignment='center',
…horizontalalignment='center', fontproperties=font_prop_bold)
            ax.text(legend_x + 0.03, self.legend_y - i *
…self.line_spacing_egde, text,
                    fontsize=12, color='black',
                    transform=ax.transAxes, verticalalignment='center',
…horizontalalignment='left', fontproperties=font_prop_regular)
        plt.show()
    def find_shortest_path(self):
        try:
            path = nx.shortest_path(
                self.G, source=self.source, target=self.target,
…weight="weight", method="dijkstra")
            length = nx.shortest_path_length(
                self.G, source=self.source, target=self.target,
…weight="weight")
            print("[Start]", end=" ")
            for route in zip(path, path[1:]):
                try:
                    print(
                        f"{self.node_labels[route[0]]}\n==[{self.distanc
…e_mapping[self.edge_labels[(route[0], route[1])]]}]==>", end=" ")
                except KeyError:
                    print(
                        f"{self.node_labels[route[0]]}\n==[{self.distanc
…e_mapping[self.edge_labels[(route[1], route[0])]]}]==>", end=" ")
            print(
                f"[End] {self.node_labels[route[1]]}\n(ระยะทาง:
…{length:.2f} km)")
        except nx.NetworkXNoPath:
            print("No path found")
    # ฟังก์ชันค้นหาเส้นทางที่สั้นที่สุด
    def draw_shortest_path(self):
        try:
            path = nx.shortest_path(
                self.G, source=self.source, target=self.target,
…weight="weight", method="dijkstra")
        except nx.NetworkXNoPath:
            print("No path found")
            return
        # ใช้ subplot เพื่อควบคุมตำแหน่ง legend
        fig, ax = plt.subplots(figsize=(16, 10))
        plt.subplots_adjust(left=0, right=0.6, top=self.top_graph,
…bottom=self.bottom_graph)
        node_colors = []  # สีของแต่ละโหนด
        edge_colors = []  # สีของแต่ละเส้น
        for node in self.G.nodes():
            if node == self.source:
```

```
            node_colors.append("lightgreen")
        elif node == self.target:
            node_colors.append("salmon")
        elif node in path:
            node_colors.append("lightblue")
        else:
            node_colors.append("lightgray")
    for edge in self.G.edges():
        if edge in zip(path, path[1:]) or edge in zip(path[1:],
…path):
            edge_colors.append("red")
        else:
            edge_colors.append("lightgray")
    # วาดโหนดโดยไม่ใส่ label
    nx.draw(self.G, self.pos, with_labels=False,
            node_color=node_colors, edge_color=edge_colors,
…node_size=1000)
    for node, (x, y) in self.pos.items():
        if node in path:
            plt.text(x, y, str(node), fontsize=10,
…fontweight="bold", color="black", verticalalignment="center",
…horizontalalignment="center")
        else:
            plt.text(x, y, str(node), fontsize=10,
…fontweight="bold", color="white", verticalalignment="center",
…horizontalalignment="center")
    # เพิ่ม Legend แบบตัวเลข
    legend_x = 1
    font_size = 12
    font_weight = 'semibold'
    for i, (count, label) in enumerate(self.node_labels.items()):
        if count == self.source:
            fc_color = "lightgreen"
            fc_font_color = 'black'
            font_color = 'black'
        elif count == self.target:
            fc_color = "salmon"
            fc_font_color = 'black'
            font_color = 'black'
        elif count in path:
            fc_color = "lightblue"
            fc_font_color = 'black'
            font_color = 'black'
        else:
            fc_color = "lightgray"
            fc_font_color = 'white'
            font_color = 'lightgray'
        ax.text(legend_x, self.legend_y - i *
…self.line_spacing_node, f"{count}",
                fontsize=font_size, color=fc_font_color,
                bbox=dict(boxstyle="circle,pad=0.3", fc=fc_color,
…ec="white", lw=1),
                transform=ax.transAxes, verticalalignment='center',
…horizontalalignment='center', fontproperties=font_prop_bold)
        ax.text(legend_x + 0.03, self.legend_y - i *
```

```
…self.line_spacing_node, label,
                fontsize=font_size, fontweight=font_weight,
…color=font_color,
                transform=ax.transAxes, verticalalignment='center',
…horizontalalignment='left', fontproperties=font_prop_regular)
        # วาดป้ายกำกับบนเส้นเชื่อม (พื้นหลังสีแดง)
        legend_x = 1.35
        font_size = 9
        for (n1, n2), label in self.edge_labels.items():
            if label in ["14", "16", "19", "26"]:
                # ตำแหน่งบนเส้นที่แบ่ง 1/3 และ 2/3 ของเส้น
                x, y = (self.pos[n1][0] * 1.15 + 1.85 * self.pos[n2][0])
…/ 3, (self.pos[n1][1] * 1.15 + 1.85 * self.pos[n2][1]) / 3
            else:
                # ตำแหน่งกึ่งกลางของเส้น
                x, y = (self.pos[n1][0] + self.pos[n2][0]) / 2,
…(self.pos[n1][1] + self.pos[n2][1]) / 2
            if (n1, n2) in zip(path, path[1:]) or (n2, n1) in zip(path,
…path[1:]):
                ax.text(x, y, label, fontsize=font_size,
…fontweight=font_weight, color='white',
                        bbox=dict(boxstyle="round,pad=0.3", fc="red",
…ec="black", lw=1),
                        horizontalalignment='center',
…verticalalignment='center')
            else:
                ax.text(x, y, label, fontsize=font_size,
…fontweight=font_weight, color="white",
                        bbox=dict(boxstyle="round,pad=0.3",
…fc="lightgray", ec="lightgray", lw=1),
                        horizontalalignment='center',
…verticalalignment='center')
        font_size = 12
        for i, (num, text) in enumerate(self.distance_mapping.items()):
            if num in [self.edge_labels.get((path[j], path[j+1]),
…self.edge_labels.get((path[j+1], path[j]), None)) for j in
…range(len(path)-1)]:
                fc_color = "red"
                fc_font_color = 'white'
                font_color = 'black'
            else:
                fc_color = "lightgray"
                fc_font_color = 'white'
                font_color = 'lightgray'
            ax.text(legend_x, self.legend_y - i *
…self.line_spacing_egde, num,
                    fontsize=10, color=fc_font_color,
                    bbox=dict(boxstyle="round,pad=0.3", fc=fc_color,
…ec=font_color, lw=1),
                    transform=ax.transAxes, verticalalignment='center',
…horizontalalignment='center', fontproperties=font_prop_bold)
            ax.text(legend_x + 0.03, self.legend_y - i *
…self.line_spacing_egde, text,
                    fontsize=font_size, color=font_color,
                    transform=ax.transAxes, verticalalignment='center',
```

```
…horizontalalignment='left', fontproperties=font_prop_regular)
        # แสดงผลโดย full size window
        plt.show()
    def set_source_target(self, source, target):
        self.source = source
        self.target = target
if __name__ == "__main__":
    graph = Graph()
    while True:
        print("ระบบค้นหาเส้นทางที่สั้นที่สุด")
        print("1. แสดงกราฟ")
        print("2. ระบุจุดเริ่มต้นและปลายทาง")
        if graph.source and graph.target:
            print(
                f"3. หาเส้นทางที่สั้นที่สุดจาก
…{graph.node_labels[graph.source]} ไป {graph.node_labels[graph.target]}")
            print(
                f"4. แสดงกราฟที่สั้นที่สุดจาก
…{graph.node_labels[graph.source]} ไป {graph.node_labels[graph.target]}")
        print("0. ออกจากโปรแกรม")
        choice = input("เลือกการทำงาน: ")
        if choice == "0":
            break
        elif choice == "1":
            graph.draw_graph()
        elif choice == "2":
            for i, node in graph.node_labels.items():
                print(f"{i:>2} = {node}")
            try:
                source = int(input("ระบุจุดเริ่มต้น: "))
                target = int(input("ระบุจุดปลายทาง: "))
            except ValueError:
                print("ระบุจุดเริ่มต้นและปลายทางไม่ถูกต้อง")
                input("กด Enter เพื่อดำเนินการใหม่อีกครั้ง...")
                continue
            graph.set_source_target(source, target)
            input("ระบุจุดเริ่มต้นและปลายทางเรียบร้อย กด Enter
…เพื่อดำเนินการต่อ...")
        elif choice == "3":
            graph.find_shortest_path()
            input("กด Enter เพื่อดำเนินการต่อ...")
        elif choice == "4":
            graph.draw_shortest_path()
        else:
            print("ระบุหมายเลขไม่ถูกต้อง")
            input("กด Enter เพื่อดำเนินการใหม่อีกครั้ง...")
        print()
```