





```
File Edit Selection View Go Run Terminal Help
Dagim, Gerardo TSP

class DagimGerardoTSP:
    def __init__(self, ruta):
        self.ruta = ruta
        self.vertices = []
        self.edges = []
        self.resultado = None

    def leer_datos(self):
        with open(self.ruta, 'r') as f:
            lines = f.readlines()
            for line in lines:
                line = line.strip()
                if line.startswith('v'):
                    vertex = line.split()[1]
                    self.vertices.append(vertex)
                elif line.startswith('e'):
                    edge = line.split()[2:]
                    self.edges.append(edge)

    def crear_figura(self):
        fig = Figure(figsize=(10, 10))
        ax = fig.add_subplot(111)
        ax.set_title('Grafo del Problema del Viajero (TSP)')
        ax.axis('off')

        # Dibujar grafos
        for i, j, w in self.edges:
            ax.plot([i, j], [0, 0], 'b-')
            label = f'({i}, {j}) {w}'
            ax.annotate(label, xy=(i, 0), xytext=(j, 0), textangle=90)

        # Mostrar la figura en Tk
        canvas = FigureCanvasTkAgg(fig, master=parent)
        canvas.draw()
        widget = canvas.get_tk_widget()
        widget.pack(fill='both', expand=True)
        return canvas

    def mostrar_resultado(self):
        resultado = self.resultado.get()
        if resultado:
            ruta = resultado['ruta']
            if ruta is None:
                resultado.config(text='Resultado: No se encontró ruta válida.')
            else:
                resultado.config(text=f'Resultado: Ruta: {ruta} Costo: {resultado["costo"]}')

# Crear la ventana principal
root = Tk()
root.title('DagimGerardoTSP')
root.geometry('400x300')

# Crear la figura
figura = DagimGerardoTSP(ruta).crear_figura()

# Botón para calcular
boton = Button(root, text='Calcular Ruta', command=figura.mostrar_resultado())
boton.pack(pady=10)

root.mainloop()
```

```
File Edit Selection View Go Run Terminal Help
Dagim, Gerardo TSP

class DagimGerardoTSP:
    def __init__(self, ruta):
        self.ruta = ruta
        self.vertices = []
        self.edges = []
        self.resultado = None

    def leer_datos(self):
        with open(self.ruta, 'r') as f:
            lines = f.readlines()
            for line in lines:
                line = line.strip()
                if line.startswith('v'):
                    vertex = line.split()[1]
                    self.vertices.append(vertex)
                elif line.startswith('e'):
                    edge = line.split()[2:]
                    self.edges.append(edge)

    def crear_figura(self):
        fig = Figure(figsize=(10, 10))
        ax = fig.add_subplot(111)
        ax.set_title('Grafo del Problema del Viajero (TSP)')
        ax.axis('off')

        # Dibujar grafos
        for i, j, w in self.edges:
            ax.plot([i, j], [0, 0], 'b-')
            label = f'({i}, {j}) {w}'
            ax.annotate(label, xy=(i, 0), xytext=(j, 0), textangle=90)

        # Mostrar la figura en Tk
        canvas = FigureCanvasTkAgg(fig, master=parent)
        canvas.draw()
        widget = canvas.get_tk_widget()
        widget.pack(fill='both', expand=True)
        return canvas

    def mostrar_resultado(self):
        resultado = self.resultado.get()
        if resultado:
            ruta = resultado['ruta']
            if ruta is None:
                resultado.config(text='Resultado: No se encontró ruta válida.')
            else:
                resultado.config(text=f'Resultado: Ruta: {ruta} Costo: {resultado["costo"]}')

# Crear la ventana principal
root = Tk()
root.title('DagimGerardoTSP')
root.geometry('400x300')

# Crear la figura
figura = DagimGerardoTSP(ruta).crear_figura()

# Botón para calcular
boton = Button(root, text='Calcular Ruta', command=figura.mostrar_resultado())
boton.pack(pady=10)

root.mainloop()
```