Actividad 2 - Búsqueda y sistemas basados en reglas

Brayan Steven Bonilla Castellanos
Juan Carlos Monsalve Gómez

Universidad Ibero.
Ingeniería de Software
Inteligencia artificial

Enlace repositorio GitHub

https://github.com/Brayan-Bonilla1224/IA_unidad2_act4

Enlace video-tutorial

https://youtu.be/wubtqTtSJbo

Descripción del ejercicio

Escribir en Python las instrucciones para el desarrollo de un sistema inteligente que, a partir de una base de conocimiento escrito en reglas lógicas, desarrolle la mejor ruta para moverse desde un punto A y un punto B en el sistema de transporte masivo local.

Ejecución del ejercicio

Importamos las librerías necesarias para dibujar y mostrar el grafo de red

```
import networkx as nx
import matplotlib.pyplot as plt
```

Construimos una base de conocimiento escrito en reglas lógicas

```
#Representación del grafo del sistema de transporte
#Distancia en metros (simulada) desde cada uno de los lugares hacia el otro
grafo_transporte = {
    'marcotobon': {'info': 'Marco Tobon', 'distancias': {'parque': 150, 'capilla': 300, 'hospital': 450, 'basilica': 700}},
    'parque': {'info': 'Parque Principal', 'distancias': {'marcotobon': 150, 'capilla': 100}},
    'capilla': {'info': 'Capilla de la Humildad', 'distancias': {'parque': 100, 'hospital': 100, 'marcotobon': 300}},
    'hospital': {'info': 'Hospital', 'distancias': {'capilla': 100, 'basilica': 200, 'marcotobon': 450}},
    'basilica': {'info': 'basilica', 'distancias': {'hospital': 200, 'marcotobon': 700}},
}
```

Construimos un método que se encargue de extraer las distancias de la estación que recibe como parámetro

```
#Extraccion de las distancias segun la estacion a validar
def expansion(origen_actual):
    adyacentes = []
    for nodo_adyacente in grafo_transporte[origen_actual]['distancias']:
        distancia = grafo_transporte[origen_actual]['distancias'][nodo_adyacente]
        adyacentes.append((nodo_adyacente, distancia))
    return adyacentes
```

Construimos un método que se encargue de la lógica analítica que me permita identificar la mejor ruta para desplazarme de un punto A a un punto B

```python
#Busqueda de la mejor ruta para desplazarse
def buscar_ruta(origen, destino):
    matriz = [(origen, 0, [])]
    estaciones = set()

    while matriz:
        origen_actual, distancia_recorrida, ruta_optima = matriz.pop(0)

        if origen_actual == destino:
            return ruta_optima, distancia_recorrida

        estaciones.add(origen_actual)

        for siguiente_estacion, distancia in expansion(origen_actual):
            distancia_total = distancia_recorrida + distancia
            ruta_optimizada = ruta_optima + [siguiente_estacion]
            if siguiente_estacion not in estaciones:
                matriz.append((siguiente_estacion, distancia_total, ruta_optimizada))

        matriz.sort(key=lambda x: x[1])
    return None, None
```

Construimos un método para dibujar el grafo de red (grafos dirigidos) y mostrarlo

```python
# Función para dibujar el grafo del sistema de transporte
def dibujar_grafo():
    G = nx.Graph()

    # Agregar nodos al grafo
    for nodo in grafo_transporte:
        G.add_node(nodo, info=grafo_transporte[nodo]['info'])

    # Agregar arcos al grafo
    for nodo_actual in grafo_transporte:
        for nodo_adyacente, distancia in grafo_transporte[nodo_actual]['distancias'].items():
            G.add_edge(nodo_actual, nodo_adyacente, weight=distancia)

    # Obtener la posición de los nodos para dibujar el grafo
    pos = nx.spring_layout(G)

    # Dibujar los nodos con su información
    labels = nx.get_node_attributes(G, 'info')
    nx.draw_networkx_labels(G, pos, labels=labels)

    # Dibujar los arcos con las distancias
    edge_labels = nx.get_edge_attributes(G, 'weight')
    nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)
    nx.draw(G, pos, with_labels=False, node_size=2000, node_color='lightblue')

    # Mostrar la gráfica
    plt.show()
```

Definimos el main de la aplicación

```python
# Interfaz de usuario
def main():
    origen = input("Ingrese la estación de origen: ")
    destino = input("Ingrese la estación de destino: ")

    ruta_optima, distancia = buscar_ruta(origen, destino)

    if ruta_optima is not None:
        print("La mejor ruta es:", ruta_optima)
        print("La mejor ruta tiene una distancia de:", distancia)
    else:
        print("No se encontró una ruta desde", origen, "hasta", destino)

    dibujar_grafo()
```

Por último, la ejecución del programa

```python
# Ejecución del programa
if __name__ == '__main__':
    main()
```

Ejecución del ejercicio

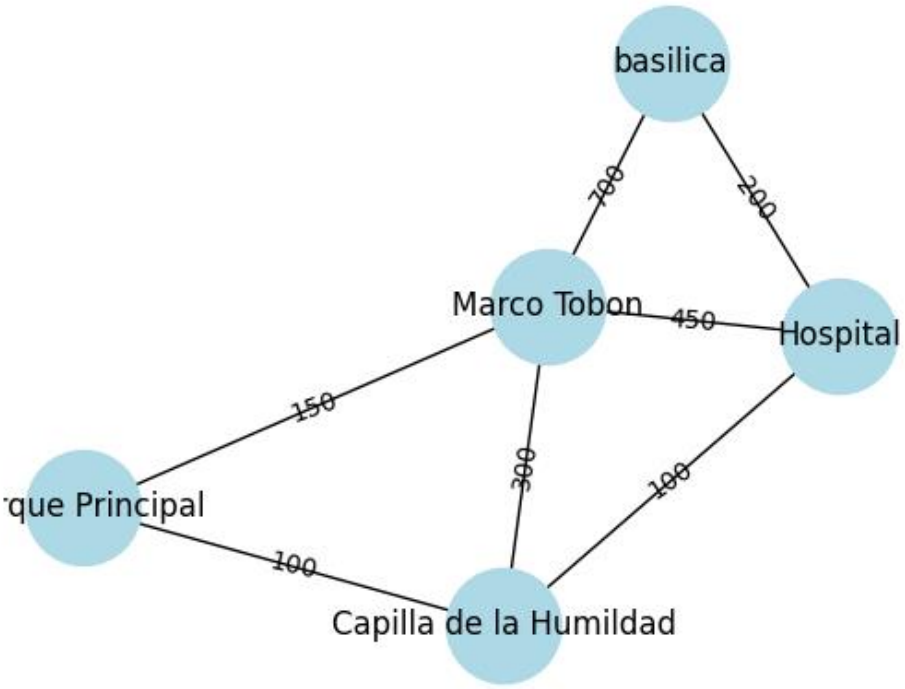
Ingresamos los valores solicitados

```
Ingrese la estación de origen: parque
Ingrese la estación de destino: basilica
```

Obtenemos una respuesta conforme a la ejecución correcta del programa

```
La mejor ruta es: ['capilla', 'hospital', 'basilica']
La mejor ruta tiene una distancia de: 400
```

Validamos la información obtenida por el programa con el grafo



En donde confirmamos el funcionamiento correcto del programa, cumpliendo con el objetivo principal de identificar la mejor ruta de desplazamiento de un punto A a un punto B

Conclusiones

- Todo análisis de datos parte de unos objetivos, siendo entonces lo más importante el diseño de la investigación o la estrategia de análisis, no la herramienta, es fundamental tener en cuenta que la herramienta es solo un medio.

Bibliografía

Rojas-Jimenez, K. (s. f.). *Capítulo 9 Análisis de Redes | Ciencia de Datos para Ciencias Naturales*. https://bookdown.org/keilor_rojas/CienciaDatos/an%C3%A1lisis-de-redes.html