

Actividad 2 - Búsqueda y sistemas basados en reglas

Jorge Andres Sandoval Sanchez

Marcela Alejandra Mendieta Teran

Juan David Aya Pesca

Carlos Alberto Gallego Benítez

Corporación Universitaria Iberoamericana

Facultad de Ingeniería

Ingeniería de software virtual

Inteligencia artificial

Docente Jorge Isaac Castañeda Valbuena

Septiembre 26, 2024

Tabla de contenido

Contenido	3
Pruebas del algoritmo	3
Enlace del repositorio	5
Enlace del video	5
Referencias	6

Contenido

Pruebas del algoritmo

1. Importación de las librerías a usar, creación del grafo, los nodos y aristas que representan las conexiones de ciudades y el tiempo

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

# Crear un grafo dirigido
G = nx.DiGraph()

# Agregar las conexiones entre ciudades y los tiempos de viaje (en minutos)
G.add_edge('Ciudad_A', 'Estación_X', weight=40) # De Ciudad_A a Estación_X en 40 minutos
G.add_edge('Estación_X', 'Ciudad_B', weight=15) # De Estación_X a Ciudad_B en 15 minutos
G.add_edge('Ciudad_A', 'Estación_Y', weight=25) # De Ciudad_A a Estación_Y en 25 minutos
G.add_edge('Estación_Y', 'Ciudad_B', weight=45) # De Estación_Y a Ciudad_B en 45 minutos
G.add_edge('Estación_X', 'Estación_Y', weight=30) # De Estación_X a Estación_Y en 30 minutos
G.add_edge('Ciudad_A', 'Ciudad_C', weight=60) # De Ciudad_A a Ciudad_C en 60 minutos
G.add_edge('Ciudad_C', 'Estación_Z', weight=35) # De Ciudad_C a Estación_Z en 35 minutos
G.add_edge('Estación_Z', 'Ciudad_B', weight=20) # De Estación_Z a Ciudad_B en 20 minutos
G.add_edge('Ciudad_C', 'Ciudad_D', weight=50) # De Ciudad_C a Ciudad_D en 50 minutos
G.add_edge('Ciudad_D', 'Estación_Y', weight=30) # De Ciudad_D a Estación_Y en 30 minutos

print('>>>> Finalizo la creación del grafo.')

[1] ✓ 22s
... >>>> Finalizo la creación del grafo.
```

2. Creación de la función para obtener la mejor ruta, entre un origen y un destino de acuerdo con el tiempo de viaje entre los nodos, usando el algoritmo de dijkstra.

```
# Función para encontrar la mejor ruta utilizando Dijkstra
def mejor_ruta(origen, destino):
    # Usamos el algoritmo de Dijkstra para encontrar la ruta más corta basada en los pesos
    ruta_mas_corta = nx.dijkstra_path(G, source=origen, target=destino, weight='weight')
    tiempo_total = nx.dijkstra_path_length(G, source=origen, target=destino, weight='weight')

    print(f"La mejor ruta de {origen} a {destino} es: {ruta_mas_corta}, con un tiempo de {tiempo_total} minutos.")

    return ruta_mas_corta
print('>>>> Se creo la función Mejor Ruta.')

[2] ✓ 0.0s
... >>>> Se creo la función Mejor Ruta.
```

- Creación de la función para dibujar el grafo, los nodos, las aristas y resaltar la mejor ruta entre el origen y el destino.

```
# Función para dibujar el grafo, resaltando la mejor ruta
def dibujar_grafo(G, ruta_mas_corta):
    pos = nx.spring_layout(G) # Posiciones para todos los nodos
    nx.draw(G, pos, with_labels=True, node_color='lightblue', node_size=2000, font_size=16, font_color='black', font_weight='bold', arrows=True)

    # Resaltar la mejor ruta
    rutas_aristas = [(ruta_mas_corta[i], ruta_mas_corta[i + 1]) for i in range(len(ruta_mas_corta) - 1)]
    nx.draw_networkx_edges(G, pos, edgelist=rutas_aristas, edge_color='green', width=3) # Resaltar en verde

    # Dibujar etiquetas de peso
    edge_labels = nx.get_edge_attributes(G, 'weight')
    nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_color='red', font_size=14)

    plt.title('Grafo de Rutas de Transporte')
    plt.axis('off') # Desactivar ejes
    plt.show()
    print('>>> Se Creo la Función Dibujar Grafo.')

✓ 0.0s
>>> Se Creo la Función Dibujar Grafo.
```

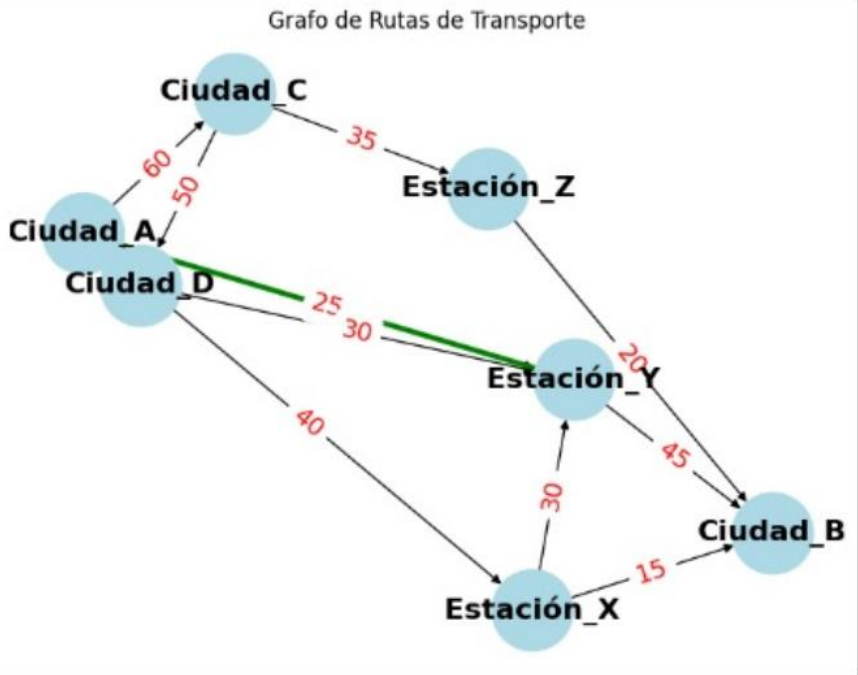
- Se llama la función del algoritmo y se obtiene la mejor ruta, para hacer el llamado a la función para dibujar el grafo.

```
# Buscar la mejor ruta de A a B
ruta = mejor_ruta('Ciudad_A', 'Estación_Y')
print('>>> Prueba del algoritmo Dijkstra')
# Dibujar el grafo, resaltando la mejor ruta
dibujar_grafo(G, ruta)
print('>>> Muestra el mejor Recorrido para la prueba.')

✓ 1.0s

La mejor ruta de Ciudad_A a Estación_Y es: ['Ciudad_A', 'Estación_Y'], con un tiempo de 25 minutos.
>>> Prueba del algoritmo Dijkstra
```

Grafo de Rutas de Transporte



```
>>> Muestra el mejor Recorrido para la prueba.
```

Enlace del repositorio

https://github.com/andressandoval21/Actividad_2_IA.git

Enlace del video

[Actividad 2 Inteligencia Artificial-20240926_213312-Grabación de la reunión.mp4](#)

Referencias

- Matplotlib — Visualization with Python. (s. f.). <https://matplotlib.org/>
- Navone, E. C. (2023, 2 agosto). Algoritmo de la ruta más corta de Dijkstra - Introducción gráfica y detallada. freeCodeCamp.org. <https://www.freecodecamp.org/espanol/news/algoritmo-de-la-ruta-mas-corta-de-dijkstra-introduccion-grafica/>
- NetworkX — NetworkX documentation. (s. f.). <https://networkx.org/>