Materia: Sistemas Expertos

Profesor: Mauricio Alejandro Cabrera Arellano

Grado y grupo: 7F

Alumno: Daniel Alejandro Flores Sepúlveda 21310203

Fecha de entrega: 01/09/2024

Traer sus repositorios para verificar el funcionamiento de los algoritmos de Inteligencia Artificial.

Modificar los algoritmos para un caso de la vida diaria y arrojar un resultado entre la data.

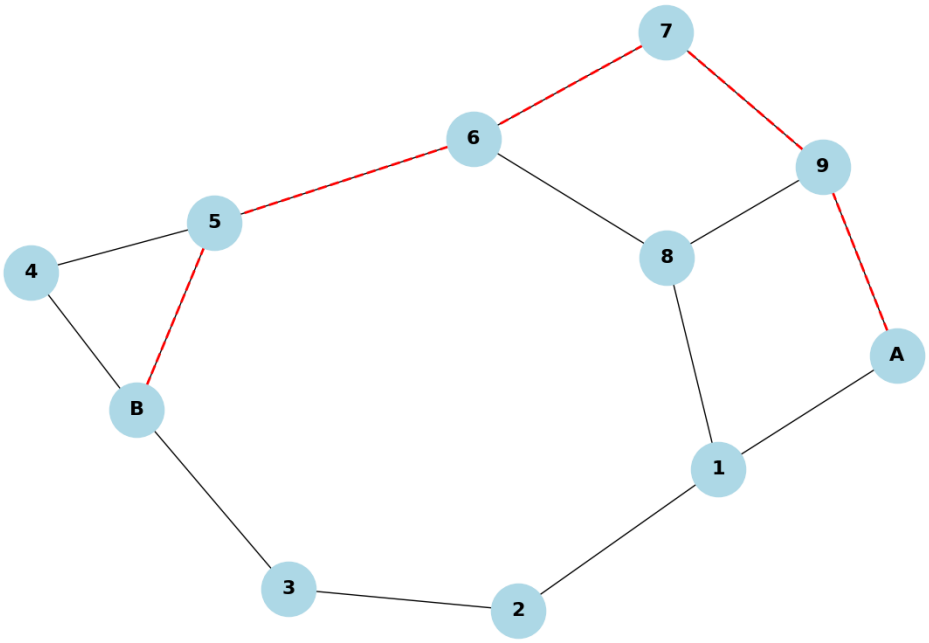
1. En el caso de los grafos nos ayudan a encontrar el camino entre un punto a y un punto B, en este caso se escoge el llegar desde mi casa a la escuela en un mapa.

2. En este mapa se colocan las intersecciones de las avenidas principales las cuales podemos tomar y las conexiones entre si para poder obtener un grafo.

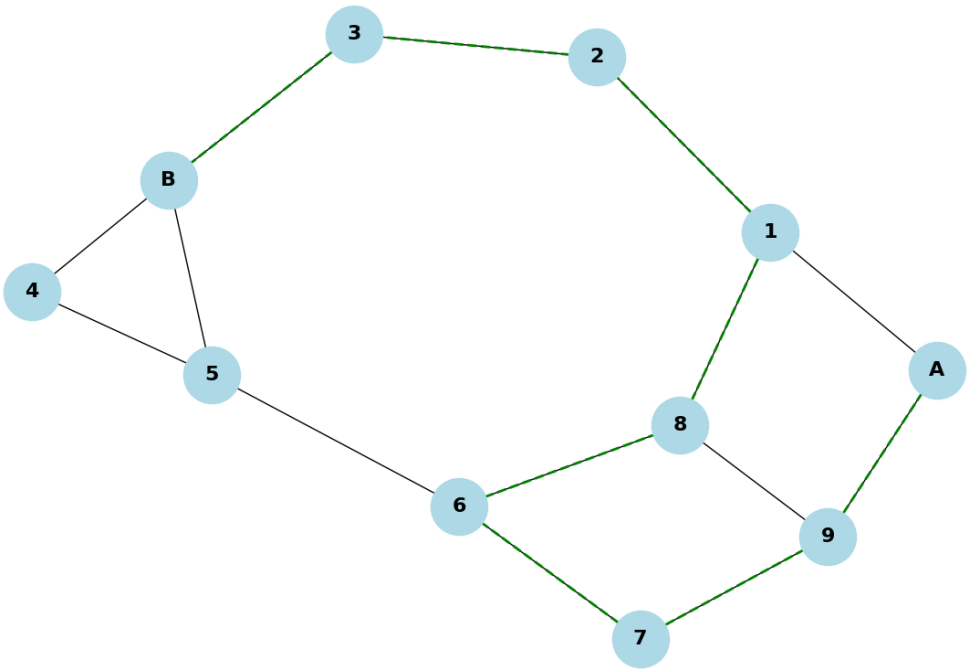
3. Con este grafo lo pasamos a código y obtenemos lo siguiente

Ruta más corta: ['A', 9, 7, 6, 5, 'B']

Ruta más larga: ['A', 9, 7, 6, 8, 1, 2, 3, 'B']



Camino mas corto



Camino mas largo

Código:

```
import networkx as nx
import matplotlib.pyplot as plt
from collections import deque

# Definir el grafo
grafo = {
    'A': [9],
    'B': [3, 4, 5],
    1: ['A', 8, 2],
    2: [3, 1],
    3: [2, 'B'],
    4: [5, 'B'],
    5: [4, 'B', 6],
    6: [5, 7, 8],
    7: [6, 9],
    8: [1, 9, 6],
    9: [ 7, 8, 'A']
}

# Crear un grafo de NetworkX
G = nx.Graph()

# Añadir nodos y aristas al grafo
for nodo, vecinos in grafo.items():
    for vecino in vecinos:
        G.add_edge(nodo, vecino)

# Función para encontrar la ruta más corta usando BFS
def ruta_mas_corta(grafo, inicio, fin):
    queue = deque([[inicio]])
    visitado = set()

    while queue:
        path = queue.popleft()
        nodo = path[-1]

        if nodo == fin:
            return path

        if nodo not in visitado:
            visitado.add(nodo)
            for vecino in grafo.get(nodo, []):
                nueva_ruta = list(path)
                nueva_ruta.append(vecino)
                queue.append(nueva_ruta)

# Función para encontrar la ruta más larga usando DFS
def ruta_mas_larga(grafo, inicio, fin):
    stack = [[inicio]]
    rutas_largas = []

    while stack:
        path = stack.pop()
        nodo = path[-1]

        if nodo == fin:
            rutas_largas.append(path)
```

```
        else:
            for vecino in grafo.get(nodo, []):
                if vecino not in path:  # Evitar ciclos
                    nueva_ruta = list(path)
                    nueva_ruta.append(vecino)
                    stack.append(nueva_ruta)

    # Devolver la ruta más larga encontrada
    return max(rutas_largas, key=len)

# Encontrar la ruta más corta y más larga de A a B
ruta_corta = ruta_mas_corta(grafo, 'A', 'B')
ruta_larga = ruta_mas_larga(grafo, 'A', 'B')

print("Ruta más corta:", ruta_corta)
print("Ruta más larga:", ruta_larga)

# Visualizar el grafo
pos = nx.spring_layout(G)  # Posicionamiento de los nodos
plt.figure(figsize=(12, 8))

# Dibujar el grafo
nx.draw(G, pos, with_labels=True, node_size=2000, node_color='lightblue', font_size=16, font_weight='bold')

# Dibujar las rutas
if ruta_corta:
    nx.draw_networkx_edges(G, pos, edgelist=list(zip(ruta_corta[:-1], ruta_corta[1:])), edge_color='red', width=2,
style='dashed')
if ruta_larga:
        nx.draw_networkx_edges(G,  pos,  edgelist=list(zip(ruta_larga[:-1],  ruta_larga[1:])),  edge_color='green',
width=2)

plt.title('Grafo con Rutas Más Corta (Rojo) y Más Larga (Verde)')
plt.show()
```