UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

Guatemala 29 de Abril de 2024
Adrián López 231361

Algoritmos y Estructuras de Datos
HDT 9: Algoritmo de Dijkstra

Objetivos:
a. Implementar el algoritmo de Dijkstra.
b. Uso de grafos para modelar rutas de entrega.

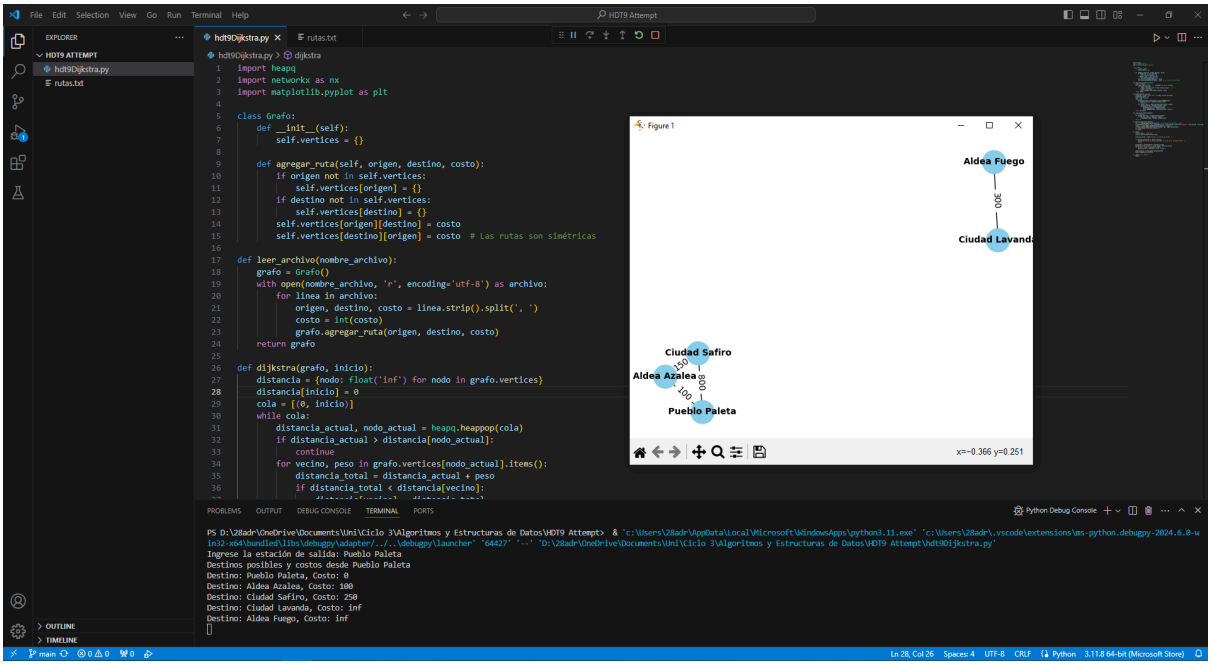
"Se debe de implementar una agencia de viajes alimentado por un archivo rutas.txt El usuario debe de ingresar el nombre de la ruta de salida y el sistema debe de utilizar el algoritmo de Dijkstra para detectar cuales son los destinos posibles con el sistema de buses y el costo de la ruta más barata a cada destino. Todas las rutas son simétricas por lo que sí existe una ruta de X a Y también existe una ruta de Y a X por el mismo costo."

Link de Repositorio:
https://github.com/adirnnn/HDT9-Algoritmo-de-Dijkstra.git

a. *Implemente un sistema que cree un grafo en NetworkX que representa las rutas del archivo rutas.txt.*

```
41  def crear_grafo_networkx(grafo):
42      G = nx.Graph()
43      for origen, destinos in grafo.vertices.items():
44          for destino, costo in destinos.items():
45              G.add_edge(origen, destino, weight=costo)
46      return G
47
48  def mostrar_mapa(grafo_networkx):
49      pos = nx.spring_layout(grafo_networkx)  # Layout para posicionar los nodos
50      nx.draw(grafo_networkx, pos, with_labels=True, node_size=700, node_color='skyblue', font_size=10, font_weight='bold')
51      labels = nx.get_edge_attributes(grafo_networkx, 'weight')
52      nx.draw_networkx_edge_labels(grafo_networkx, pos, edge_labels=labels)
53      plt.title("Mapa de posibles destinos")
54      plt.show()
55
```

b. *Implemente la funcionalidad de ver un mapa de posibles destinos desde una estación de salida.*



c. *Implemente el algoritmo de Dijkstra para encontrar las mejores rutas para llegar a todos los destinos posibles*

```python
def dijkstra(grafo, inicio):
    distancia = {nodo: float('inf') for nodo in grafo.vertices}
    distancia[inicio] = 0
    cola = [(0, inicio)]
    while cola:
        distancia_actual, nodo_actual = heapq.heappop(cola)
        if distancia_actual > distancia[nodo_actual]:
            continue
        for vecino, peso in grafo.vertices[nodo_actual].items():
            distancia_total = distancia_actual + peso
            if distancia_total < distancia[vecino]:
                distancia[vecino] = distancia_total
                heapq.heappush(cola, (distancia_total, vecino))
    return distancia
```