Atividade: Exercícios Moodle sobre Caminho Mínimo usando Dijkstra

Aluno: Breno Farias da Silva
Registro do Aluno: 2300516
Curso: Bacharelado em Ciência da Computação
Disciplina: BCC35B - Teoria dos Grafos - IC5A_CM
Campus: Campo Mourão

----- Output Exercício 01 -----:

```
shortest path from A to D is 9
```

----- Output Exercício 02 -----:

```
Seattle -> Seattle is 0 km
Seattle -> Minneapolis is 2661 km
Seattle -> Chicago is 3322 km
Seattle -> Boston is 4935 km
Seattle -> San Francisco is 1306 km
Seattle -> Denver is 2161 km
Seattle -> Washington DC is 4467 km
Seattle -> New York is 4850 km
Seattle -> Las Vegas is 2225 km
Seattle -> Dallas is 3419 km
Seattle -> Miami is 5580 km
Seattle -> Los Angeles is 1935 km
```

----- Código Exercício 01 -----

```
def Dijkstra(Graph, starting_vertex):
    array_of_distances = [float('inf')] * len(Graph) # Create an
array with the distances from the start edge to all the others
    array_of_distances[starting_vertex] = 0 # The distance from the
start edge to itself is 0
    Queue = set (range(len(Graph))) # Set of all the points

    while (Queue):
        minimum = min(Queue, key = lambda x: array_of_distances[x])
# Get the edge with the minimum distance
        Queue.remove(minimum) # Remove the edge from the queue
        for adjacent in Graph[minimum]: # For each adjacent edge
            if array_of_distances[minimum] + Graph[minimum]
[adjacent] < array_of_distances[adjacent]: # If the distance is less
than the current distance
                array_of_distances[adjacent] =
array_of_distances[minimum] + Graph[minimum][adjacent] # Update the
distance
```

```
    return array_of_distances # Return the array of distances

def main():
    vertex_name = ["A", "B", "C", "D", "E", "F"]
    raw_input = input().split()
    vertex = int(raw_input [0])
    edges = int(raw_input [1])
    Graph = {}

    for i in range(vertex): # Create a graph with all the vertex
        Graph[i] = {}

    for i in range(edges): # Add the edges
        raw_input = input().split() # Get the edges with its origin, destination and weight
        origin = int(raw_input [0])
        destination = int(raw_input [1])
        weight = int(raw_input [2])
        Graph[origin][destination] = weight  # The weight of the edge

    raw_input = input().split() # Get the start and end points
    starting_vertex = int(raw_input [0])
    destination_vertex = int(raw_input [1])
    array_of_distances = Dijkstra(Graph, starting_vertex) # Get the array_of_distances from the start edge to all the others
    print ("shortest path from" , vertex_name[starting_vertex] , "to" , vertex_name[destination_vertex] , "is" , array_of_distances[destination_vertex])

if __name__ == '__main__':
    main ()
```

----- Código Exercício 02 -----

```
def Dijkstra(Graph, starting_vertex):
    array_of_distances = [float('inf')] * len(Graph) # Create an array with the distances from the start edge to all the others
    array_of_distances[starting_vertex] = 0 # The distance from the start edge to itself is 0
    Queue = set (range(len(Graph))) # Set of all the points

    while (Queue):
        minimum = min(Queue, key = lambda x: array_of_distances[x]) # Get the edge with the minimum distance
        Queue.remove(minimum) # Remove the edge from the queue
        for adjacent in Graph[minimum]: # For each adjacent edge
            if array_of_distances[minimum] + Graph[minimum][adjacent] < array_of_distances[adjacent]: # If the distance is less than the current distance
```

```
                array_of_distances[adjacent] =
array_of_distances[minimum] + Graph[minimum][adjacent] # Update the
distance

    return array_of_distances # Return the array of distances

def main():
    cities_name = [ "Seattle", "Minneapolis", "Chicago", "Boston",
"San Francisco", "Denver", "Washington DC", "New York", "Las Vegas",
"Dallas", "Miami", "Los Angeles" ]
    raw_input = input().split()
    vertex = int(raw_input [0])
    edges = int(raw_input [1])
    Graph = {}

    for i in range(vertex): # Create a graph with all the vertex
        Graph[i] = {}

    for i in range(edges): # Add the edges
        raw_input = input().split() # Get the edges with its
origin, destination and weight
        origin = int(raw_input [0]) - 1
        destination = int(raw_input [1]) - 1
        weight = int(raw_input [2])
        Graph[origin][destination] = weight  # The weight of the
edge
        Graph[destination][origin] = weight  # The weight of the
edge

    starting_vertex = (int(input()) - 1) # Get the start point
    array_of_distances = Dijkstra(Graph, starting_vertex) # Get the
array_of_distances from the start edge to all the others

    for i in range(len(cities_name)):
        print(cities_name[starting_vertex] + " -> " +
str(cities_name[i]) + " is " + str(array_of_distances[i]) + " km")

if __name__ == '__main__':
    main ()
```