EJERCICIO DE DIJKSTRA

Presenta

Cristian David Mora Sáenz

Docente

Segundo Fidel Puerto Garavito

Asignatura

Diseño de Algoritmos

NRC: 7487

Corporación Universitaria Minuto de Dios

Facultad de Ingeniería

Colombia, Bogotá abril 28 de 2020.

Para el código de dijkstra se predeterminaron unos pesos para cada uno de los vértices, además de otorgar una letra diferente para cada nodo, esto con el fin de que al ejecutar el ejercicio se pueda observar el recorrido que realiza el algoritmo de dijkstra.

1. Como primer paso definimos los nodos y los pesos de los vértices del grafo.

```
nodes = ('A', 'B', 'C', 'D', 'E', 'F', 'G')
distances = {
        'B': {'A': 5, 'D': 1, 'G': 2},
        'A': {'B': 5, 'D': 3, 'E': 12, 'F' :5},
        'D': {'B': 1, 'G': 1, 'E': 1, 'A': 3},
        'G': {'B': 2, 'D': 1, 'C': 2},
        'C': {'G': 2, 'E': 1, 'F': 16},
        'E': {'A': 12, 'D': 1, 'C': 1, 'F': 2},
        'F': {'A': 5, 'E': 2, 'C': 16}}
```

2. Luego creamos variables para definimos un arreglo para almacenar los nodos que no han sido visitados, una variable para medir las distancias con los pesos de los vértices, otra para almacenar los nodos que no han sido visitados y finalmente una variable para definir la recurrencia.

```
unvisited = {node: None for node in nodes} #using None as +inf
visited = {}
current = 'B'
currentDistance = 0
unvisited[current] = currentDistance
```

3. Utilizando un ciclo de tipo WHILE recorremos el grafo, teniendo en cuenta los vecinos de cada nodo y los pesos de los vértices entre estos. Teniendo en cuenta esto, nos apoyamos de condicionales de tipo IF para la comparación entre los nodos visitados y los no visitados, hasta recorrer el grafo al punto deseado.

```
while True:
        for neighbour, distance in distances[current].items():
            if neighbour not in unvisited: continue
            newDistance = currentDistance + distance
            if unvisited[neighbour] is None or unvisited[neighbour] > newDistance
                unvisited[neighbour] = newDistance
        visited[current] = currentDistance
        del unvisited[current]
        if not unvisited: break
        candidates = [node for node in unvisited.items() if node[1]]
        current, currentDistance = sorted(candidates, key = lambda x: x[1])[0]
```

4. Por último, se imprime el recorrido realizado por el algoritmo.

```
print(visited)
```

Finalmente obtenemos como resultado el recorrido de los nodos visitados hasta el nodo final. Teniendo en cuenta que este recorrido es la ruta más rápida.

```
PS C:\Users\CristianM\Desktop\Ejercicios de Diseño> & C:/Users/CristianM/AppData/Local/Programs/Python/Python38-32/python.exe "c:/Users/C
ristianM/Desktop/Ejercicios de Diseño/Dijkstra.py"
{'B': 0, 'D': 1, 'E': 2, 'G': 2, 'C': 3, 'A': 4, 'F': 4}
PS C:\Users\CristianM\Desktop\Ejercicios de Diseño>
```

Enlace al Repositorio en GitHub: https://github.com/CristianMoraS/Dijkstra