Tabla de Contenidos

ESCUELA POLITÉCNICA NACIONAL Estructuras de Datos y Algoritmos II – Computación	INFORME No.
Nombre: Luis Enrique Perez Señalin	9

Algoritmo de Dijkstra

1. Objetivos

1. Aprender el uso del algoritmo de Dijstra para encontrar la ruta más corta en grafos con pesos

2. Introducción

Anteriormente, descubriste una forma de llegar del punto A al punto B. No es necesariamente el camino más rápido. Es la ruta más corta, porque tiene el menor número de segmentos (tres segmentos o saltos). Pero supongamos que agrega tiempos de viaje a esos segmentos. Ahora ves que hay un camino más rápido. Cada segmento tiene un tiempo de viaje en minutos. Utilizaremos el algoritmo de Dijkstra para ir de Start a Finish en el menor tiempo posible.

3. EJERCICIOS PLANTEADOS Y/O PROGRAMAS IMPLEMENTADOS

3.1 Implemente el algoritmo de Dijkstra y despliegue la ruta más barata desde S hasta F - Trabaje con el siguiente ejemplo:

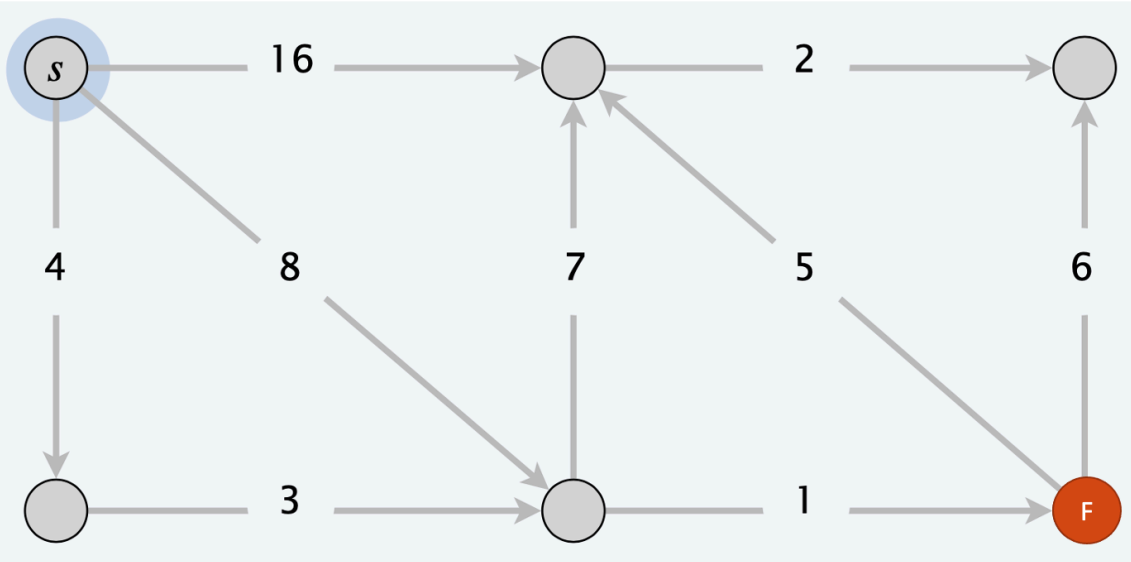

Figura 1: grafo 1, imagen_muestra

```
# Definimos las función del algoritmo
def find_lowest_cost_node(costs, processed):
    lowest_cost = float("inf")
    lowest_cost_node = None
    for node in costs:
        cost = costs[node]
        if cost < lowest_cost and node not in processed:
            lowest_cost = cost
            lowest_cost_node = node
    return lowest_cost_node

def dijkstra(graph, costs, parents):
    processed = []
    node = find_lowest_cost_node(costs, processed)
    while node is not None:
        cost = costs[node]
        neighbors = graph[node]
        if neighbors == {None}:
            processed.append(node)
            node = find_lowest_cost_node(costs, processed)
```

```
            continue
        for n in neighbors.keys():
            new_cost = cost + neighbors[n]
            if costs[n] > new_cost:
                costs[n] = new_cost
                parents[n] = node
        processed.append(node)
        node = find_lowest_cost_node(costs, processed)
    return parents, costs
```

Vamos a pasar del grafo a diccionario

```
grafo1 = {
    'S': {'A': 4,'B': 8,'C':16},
    'A': {'B': 3},
    'B': {'C': 7,'F':1},
    'C': {'D':2},
    'D': {None},
    'F': {'C':5, 'D':6}
}
parents1 = {
    'A': 'S',
    'B': 'S',
    'C': 'S',
    'D': None,
    'F': None
}
costs1 = {
    'A': 4,
    'B': 8,
    'C': 16,
    'D': float("inf"),
    'F': float("inf")
}
f_parents1, f_costs1 = dijkstra(grafo1, costs1, parents1)
print(f'{f_parents1} - {f_costs1}')
```

```
{'A': 'S', 'B': 'A', 'C': 'F', 'D': 'F', 'F': 'B'} - {'A': 4, 'B': 7, 'C': 13, 'D': 14, 'F
```

Con esto podemos concluir que la ruta más corta de S hasta F es: S -> A -> B -> F

3.2 Implemente el algoritmo de Dijkstra y despliegue la ruta más barata desde 0 hasta 1 - Trabaje con el siguiente ejemplo. Los pesos de cada arista están listados a continuación:

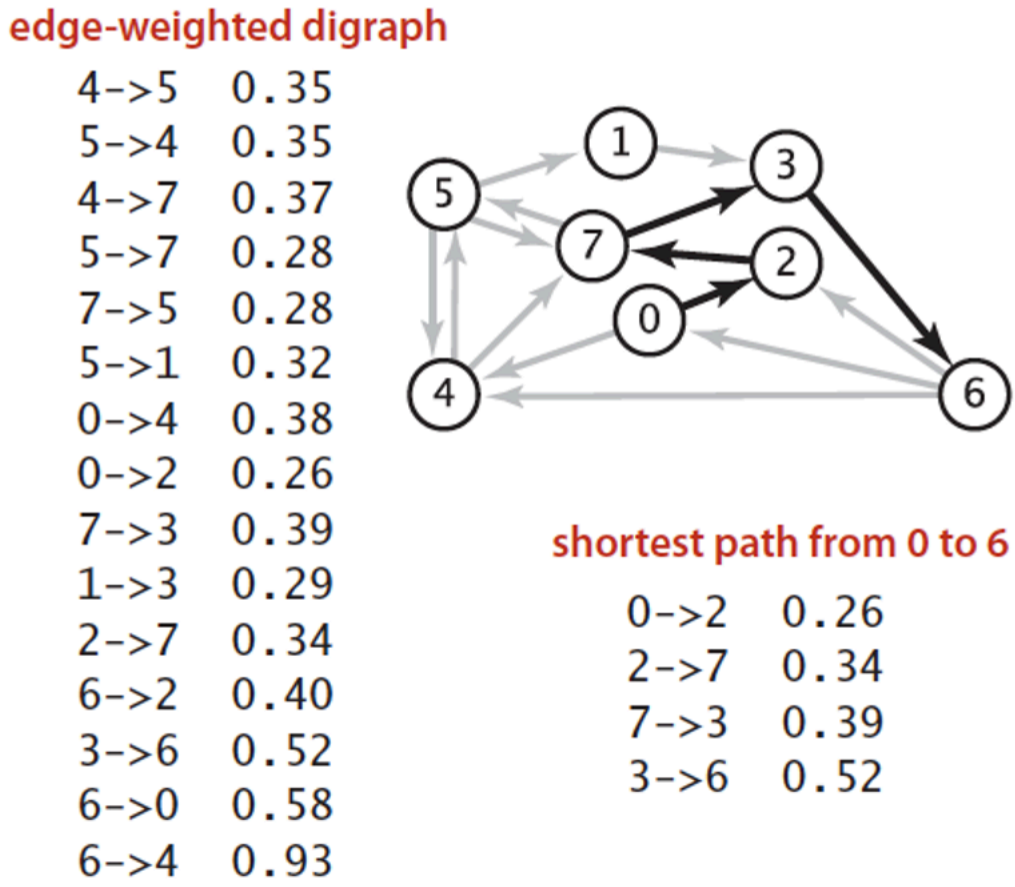

Figura 2: grafo 2, imagen_muestra

```
grafo2 = {
    '0': {'2': 0.26, '4': 0.38},
    '1': {'3': 0.29},
    '2': {'7': 0.34},
    '3': {'6': 0.52},
    '4': {'5': 0.35, '7': 0.37},
    '5': {'1': 0.32, '4': 0.35, '7': 0.28},
    '6': {'0': 0.58, '2': 0.40, '4': 0.93},
    '7': {'3': 0.39, '5': 0.28}
}
parents2 = {
```

```
    '4': '0',
    '2': '0',
    '7': None,
    '3': None,
    '6': None,
    '5': None,
    '1': None
}
costs2 = {
    '0': float("inf"),
    '4': 0.38,
    '2': 0.26,
    '7': float("inf"),
    '3': float("inf"),
    '6': float("inf"),
    '5': float("inf"),
    '1': float("inf")
}
```

```
f_parents2, f_costs2 = dijkstra(grafo2, costs2, parents2)
print(f'{f_parents2} - {f_costs2}')
```

```
{'4': '0', '2': '0', '7': '2', '3': '7', '6': '3', '5': '4', '1': '5', '0': '6'} - {'0': 2
```

Con esto podemos concluir que la ruta más corta de 0 hasta 1 es: 0 -> 4 -> 5 -> 1

4. Conclusiones

El algoritmo de Dijkstra funciona correctamente para calcular la ruta más corta ponderada

5 .Referencias bibliográficas

Aditya Bhargava. “Grokking Algorithms: An illustrated guide for programmers and other curious people”. 2016. Manning Publications.