- <u>Práctico 4 - Algoritmos greedy</u>

Teniendo el siguiente grafo de ciudades. Vamos a representar el mismo en una matriz de adyacencia.

Le damos un valor dentro del grafo a cada ciudad:

Cádiz	0
Granada	1
Jaén	2
Murcia	3
Sevilla	4

Luego construimos la matriz de adyacencias correspondiente. El valor +∞ indica que no hay conexión entre el nodo i y el nodo j:

	0	1	2	3	4
0	+∞	+∞	+∞	+∞	125
1	+∞	+∞	99	278	256
2	+∞	99	+∞	+∞	242
3	+∞	278	+∞	+∞	+∞
4	125	256	242	+∞	+∞

Vamos a mostrar el pseudocodigo para resolver el problema enunciado. Utilizaremos el algoritmo de Dijkstra para determinar el camino más corto de cada ciudad a uno de los puertos mostrados. Y además determinar cuál es el camino más corto a un puerto en general.

- Pseudocodigo de la solución:

ciudades = { 0, 1, 2, 3, 4 }, n = 5

puertos = { 0, 3 }, m = 2

```
procedure camino_mas_corto ( Grafo, n, m, puertos )
{ Devuelve el camino más corto de todas las ciudades del grafo hacia un puerto, teniendo en cuenta el conjunto de puertos }
begin
    S_caminos_cortos[] = {}
    for i := 0 to n-1 do begin
        caminos_cortos[] = {}
        for j := 0 to m-1 do begin
            if puertos[ j ] != i  ||  !puertos.contain( i ) do begin
                camino_mas_corto[] = {}
                Dijkstra( puertos[j], i, Grafo, camino_mas_corto )
                caminos_cortos.add(camino_mas_corto)
            end
        end
        S_caminos_cortos.add( camino_mas_corto( caminos_cortos ) )
    end
    return camino_mas_corto ( S_caminos_cortos )
end; {camino_mas_corto}


procedure camino_mas_corto ( caminos_cortos[] )
{ En esta función dado un vector de caminos cortos devuelve el más corto de todos }
```

Dado el pseudocodigo de la solución haremos un seguimiento de una iteración del mismo.

- <u>Seguimiento de una iteración:</u>

Para el seguimiento tomaremos la posibilidad de ir desde Murcia a Sevilla

Iteración	S	w	$D[0]$	$D[1]$	$D[2]$	$D[4]$
inicial	$\{3\}$	—	∞	278	∞	∞
1	$\{3, 1\}$	1	∞	278	377	534

$P[2] = 1$ y $P[4] = 1$