Programación 3 - TUDAI

Práctico 4 - Algoritmos greedy

Facultad de Cs. Exactas

Univ. del Centro de la Pcia. de Bs. As.

MUJICA, Martin S.
mujicamartin@gmail.com

Tandil, 23 de Mayo de 2018

Enunciado del Ejercicio

Desde un cierto conjunto grande de ciudades del interior de una provincia, se desean transportar cereales hasta alguno de los 3 puertos pertenecientes al litoral de la provincia. Se pretende efectuar el transporte total con mínimo costo sabiendo que el flete es más caro cuanto más distancia tiene que recorrer. Dé un algoritmo que resuelva este problema, devolviendo para cada ciudad el camino que debería recorrer hacia el puerto de menor costo.

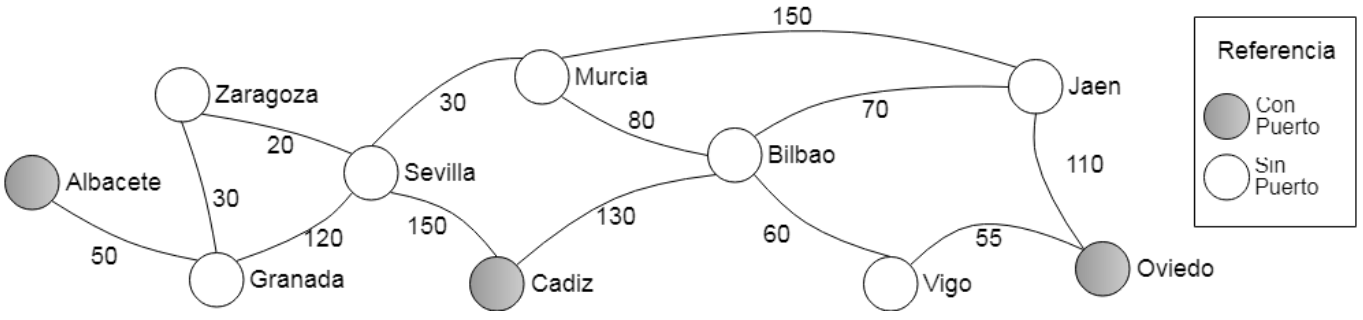
Respuesta planteada

Para resolver el ejercicio se planteó crear un método *mejorCamioAPuetos()* al que se le pasan como parámetro un grafo con las ciudades, la ciudad que se desea obtener el mejor camino, y los tres puertos posibles. Este método internamente explorara con un algoritmo Dijkstra el grafo desde los tres puertos, generando tres pares de arreglos que contendran:

En el primer arreglo (dist[]) las distancias de cada ciudad al puerto.
En el segundo arreglo (padres[]) el camino que se debe recorrer para llegar.

Luego se comparará en los tres arreglos de distancia, el valor correspondiente a la ciudad deseada y de esta manera sabremos cual es el arreglo de padres que se debe recorrer para mostrar el camino más corto.

A modo de ejemplo usando el siguiente grafo:



El método propuesto recorrera el grafo desde los tres puetos y generar los siguientes Arreglos nombre[] (para identificar el indice de la ciudad) y tres pares de arreglos dist[] (distancia al puerto) y padres[] (recorrido hasta el puerto)

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
padres1[]	0	6	8	1	6	8	2	10	2	4
dist1[]	0	210	250	50	280	130	325	100	270	80
padres2[]	4	3	0	10	2	8	9	3	2	8
dist2[]	170	130	0	200	200	180	245	150	190	170
padres3[]	4	9	2	10	7	2	0	6	7	8
dist3[]	325	115	245	275	110	195	0	225	55	245

Suponiendo que quisiéramos saber el recorrido desde **Sevilla** al puerto más cercano. El método compara en las distancias de los tres arreglos de distancia, devolverá la menor, en este caso **100** y recorrera el arreglo de padres imprimiendo las ciudades que debe atravesar: **Zaragoz, Granada, Albacete**

Pseudocogido del método *mejorCaminoAPuertos*() y los principales métodos auxiliares (*cargarDistanciasA() , actualizarDistancias()*)

```
mejorCaminoAPuertos(Grafo mapa, String ciudad, String Puerto1, String Puerto2, String Puerto3){
        // recorre el grafo (mapa) y carga los nombres de las ciudades ordenados alfabéticamente
1       cargarNombres(mapa, nomb[]);
        // Dijkstra obteniendo mejores recorridos para llegar a un puerto
2       cargarDistanciasA(mapa, puerto1, nomb[], dist1[], padres1[]);
3       cargarDistanciasA(mapa, puerto2, nomb[], dist2[], padres2[]);
4       cargarDistanciasA(mapa, puerto3, nomb[], dist3[], padres3[]);
        // obtiene la posicion de la ciudad en el arreglo de nombres
5       i = obtenerPos(ciudad, nomb[]);
        // comparó las distancias a cada uno de los puertos buscando la menor para imprimir
6       if (dist1[i] <=  dist2[i] ) && (dist1[i] <=  dist3[i] )
                // imprime el camino que se debe seguir desde una ciudad a un puerto
                imprimirCamino(i, nomb[], dist1[], padres1[]);
        else
                if (dist2[i] <=  dist3[i] )
                        imprimirCamino(i,nomb[],dist2[],padres2[]);
                else
                imprimirCamino(i,nomb[],dist3[],padres3[]);
        }
```

```
cargarDistanciasA(Grafo mapa, String puerto, nomb[], dist[], padres[]){
        // inicia arrglo de ciudades visitadas con todas las ciudades sin visitar, en 0
1       visitadas[] = 0;
        // Actualizo las distancias en infinito, los padres en 0 y puerto como primera ciudad
2       dist[] = 999999; padres[] = 0; aux = obtenerPos(puerto,nombres[]);
        // comienzo a recorrer las ciudades
4       While (hayCiudadesSinVisitar(Visitadas[])) {
                // la cargo ciudad a visitadas
        1       marcarVisitada(aux,visitadas[]);
        //obtengo las ciudades conectadas, si es mas cerca actualizo las distancias y coloco a aux como padre
        2       actualizarDistancias(Grafo mapa, aux, dist[], padres[]);
                //obtengo las ciudad con menor distancia al puerto que no fue visitada
        3       aux = proximaCiudad(Visitadas[],dist[]);
        }
}
```

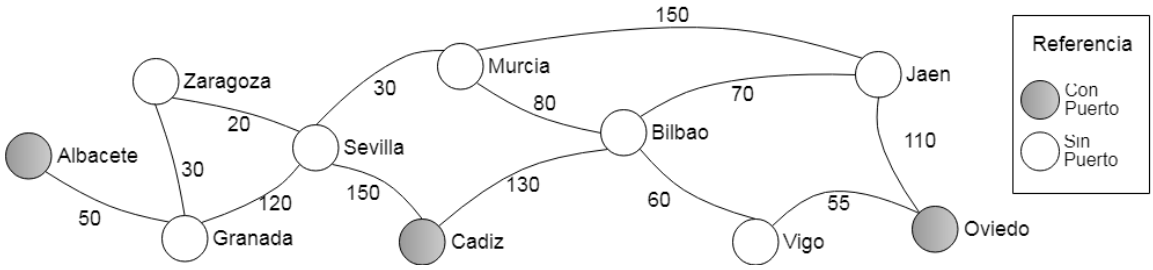
```
actualizarDistancias(Grafo mapa, aux, nomb[], dist[], padres[]){
        //para cada una de las conexiones ciudades
        for (Arista : mapa.getNodo(nomb[aux]).getAristas())
                // si la distancia + la distancia actual es menor a la que ya tenia guardada para esa ciudad
                if ((dist[aux] + Arista.getValor()) < dist[obtenerPos(Arista.getDestino(),nomb[])]){
                        // actualizo la nueva distancia
                        dist[obtenerPos(Arista.getDestino(),nomb[])] = (dist[aux] + Arista.getValor();
                        // actualizo el padre
                        padres[obtenerPos(Arista.getDestino(),nomb[])] = aux;
                }
}
```

A Continuación se muestra un seguimiento detallado del metodo con sus iteraciones donde se puede observar el estado de los arreglos que se usaran para realizar los calculos y las variables auxiliares.

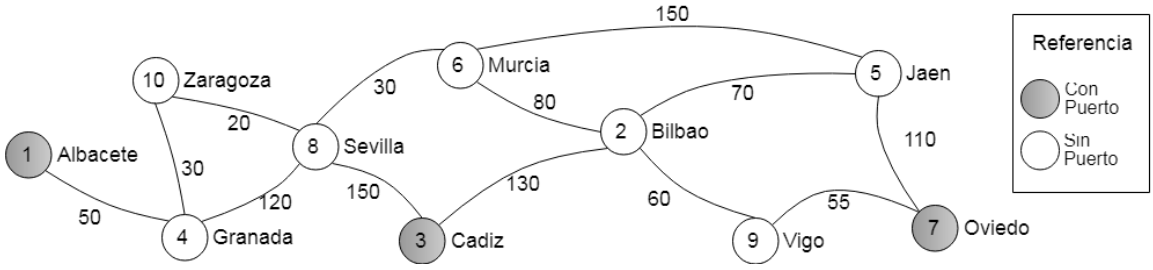
El seguimiento se realiza suponiendo que necesitamos obtener el mejor camino desde Sevilla al puerto mas cercano. Los puertos disponibles son Albacete, Cadiz y Oviedo.

Seguimiento del Metodo *mejorCaminoAPuertos*

Se ingresa el grafo, la ciudad origen y los tres puertos (Albacete, Cadiz y Oviedo)



1 Se carga el arreglo con los nombres de las ciudades del grafo, este arreglo se usa para obtener los ID correspondiente a cada ciudad



Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza

2 Se cargan los arreglos dist1[] y padres1[] con los caminos minimos para llegar desde cualqueir ciudad al puerto de Albacete

1 Se genera una arreglo de ciudades visitadas para llevar la cuenta de las ciudades recorridas y una variable auxiliar para saber cual es la ciudad actual

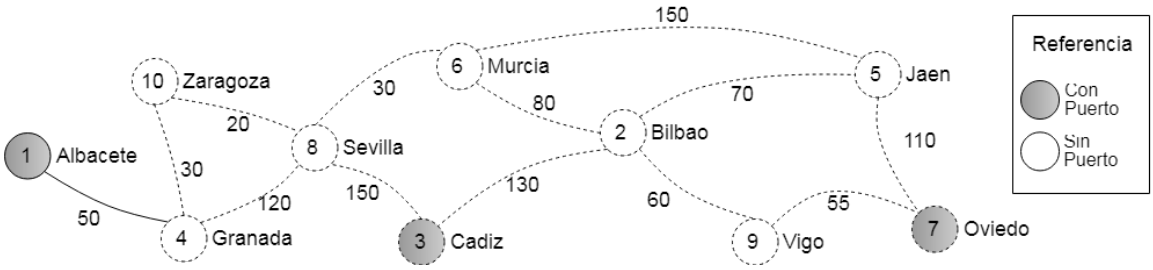
2 Se inicializan los arreglos dist[] y padres[] y se inicializa la variable aux con el indice la ciudad "Albacete"

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
visitados[]	0	0	0	0	0	0	0	0	0	0
padres1[]	0	0	0	0	0	0	0	0	0	0
dist1[]	999999	999999	999999	999999	999999	999999	999999	999999	999999	999999

3 Se comienza a realizar los calculos por ciudad hasta que esten todas las ciudades visitadas

1 Se carga la ciudad actual (1) como visitada

2 Se obienen las ciudades adyacentes (en este caso Granada) y si la distancia para llegar a esa ciudad (50) es menor que la actual (999999) se actualiza la distancia y el padre
Se actualiza la distancia a [1] Granada (0 + 50) y se coloca a [1] Albacete como padre, ya que anteriormente estaba en infinito



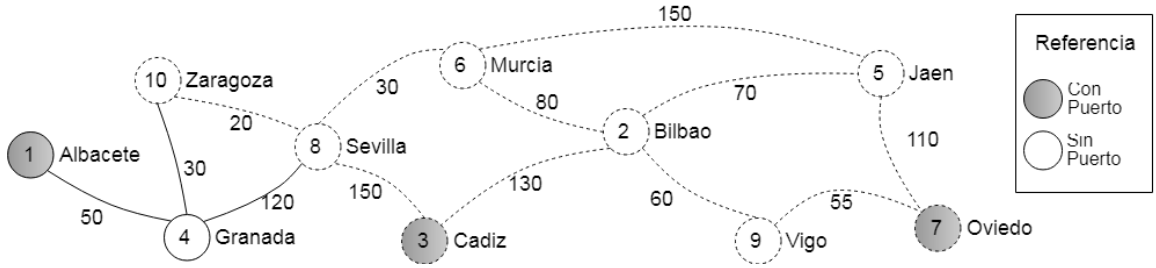
aux = 1 (Albacere)

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
visitados[]	1	0	0	0	0	0	0	0	0	0
padres1[]	0	0	0	1	0	0	0	0	0	0
dist1[]	0	999999	999999	50	999999	999999	999999	999999	999999	999999
				50						

3 Se actualiza la proxima ciudad a visitar, se elije la ciudad con la menor distancia de las que no fueron visitadas, en este caso [4] Granada

1 Se carga la ciudad actual [4] Granada como visitada

2 Se obienen las ciudades adyacentes (en este caso Zaragoza y Sevilla) y si la distancia para llegar a esa ciudad (50+30 para zaragoza y 170 para Sevilla)
Se actualiza la distancia a [8] Sevilla y [10] Zaragoza y se les coloca a [4] Granada como padre



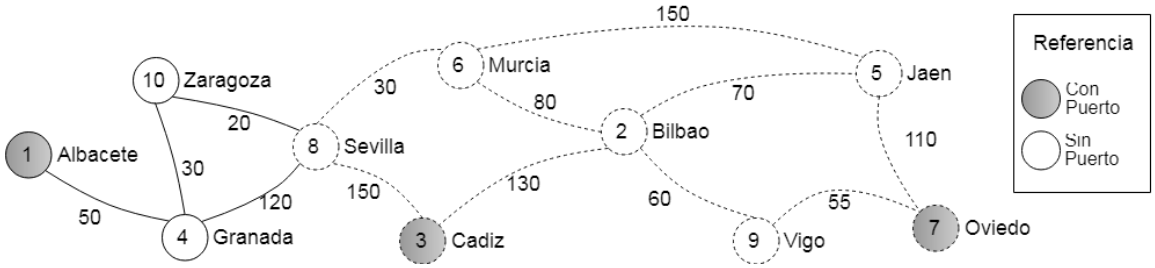
aux = 4 (Granada)

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
visitados[]	1	0	0	1	0	0	0	0	0	0
padres1[]	0	0	0	1	0	0	0	4	0	4
dist1[]	0	999999	999999	50	999999	999999	999999	170	999999	80
								120		30

3 Se actualiza la proxima ciudad a visitar, se elije la ciudad con la menor distancia de las que no fueron visitadas, en este caso [10] Zaragoza

1 Se carga la ciudad actual [10] Zaragoza como visitada

2 Se obienen las ciudades adyacentes (en este caso Sevilla) y si la distancia para llegar a esa ciudad (80+20)
Se actualiza la distancia a [8] Sevilla y se le coloca a [10] Zaragoza como padre



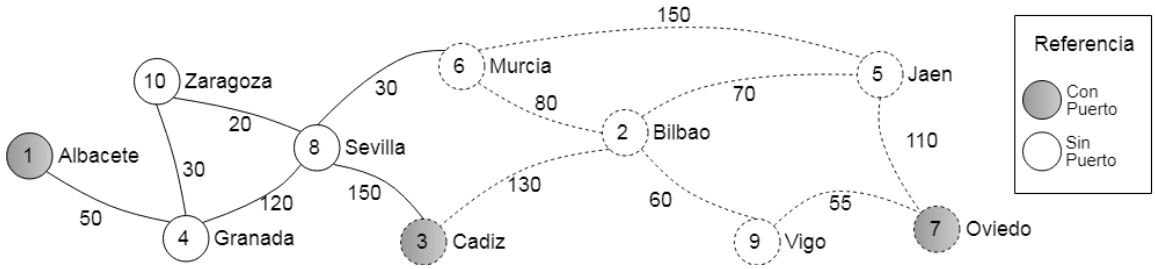
aux = 10 (Zaragoza)

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
visitados[]	1	0	0	1	0	0	0	0	0	1
padres1[]	0	0	0	1	0	0	0	10	0	4
dist1[]	0	999999	999999	50	999999	999999	999999	100	999999	80
								20		

3 Se actualiza la proxima ciudad a visitar, se elije la ciudad con la menor distancia de las que no fueron visitadas, en este caso, [8] Sevilla

1 Se carga la ciudad actual [8] Sevilla como visitada

2 Se obienen las ciudades adyacentes (en este caso Cadiz y Murcia) y si las distancia para llegar a esas ciudades (30 y 150)
Se actualiza la distancia a [3]Cadiz (100+150) y [6] Murcia (100+30) y se les coloca a [8] Sevilla como padre



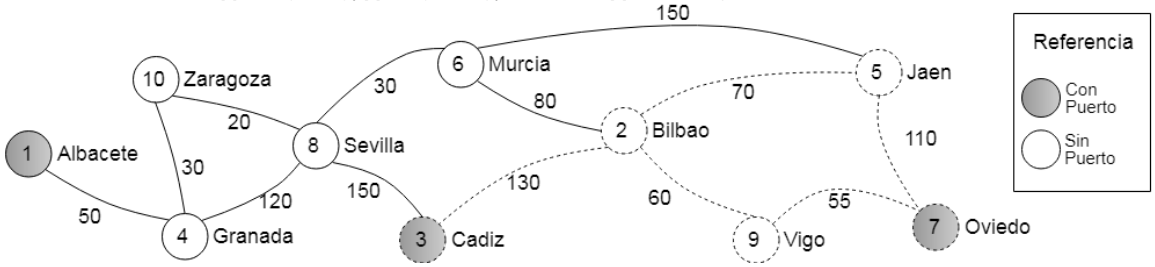
aux = 8 (Sevilla)

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
visitados[]	1	0	0	1	0	0	0	1	0	1
padres1[]	0	0	8	1	0	8	0	10	0	4
dist1[]	0	999999	250	50	999999	130	999999	100	999999	80
			150			30				

3 Se actualiza la proxima ciudad a visitar, en este caso, [6] Murcia

1 Se carga la ciudad actual [6] Murcia como visitada

2 Se obienen las ciudades adyacentes (en este caso jean y Bilbao) y si las distancia para llegar a esas ciudades (150 y 80)
Se actualiza la distancia a [2] Bilbao (130+80) y [5] Jean (130+150) y se les coloca a [6] Murcia como padre



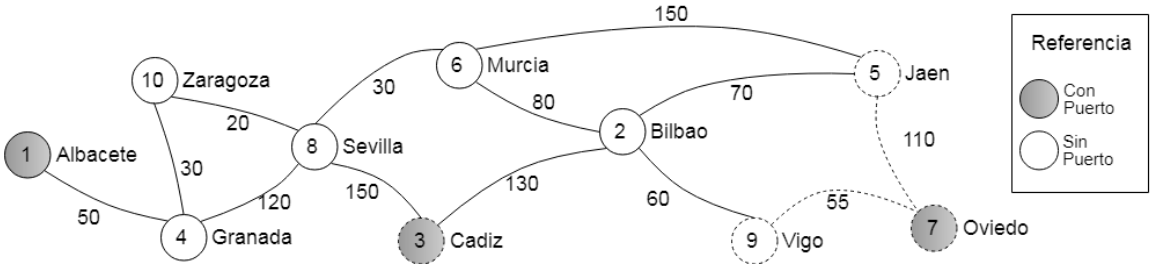
aux = 6 (Murcia)

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
visitados[]	1	0	0	1	0	1	0	1	0	1
padres1[]	0	6	8	1	6	8	0	10	0	4
dist1[]	0	210	250	50	280	130	999999	100	999999	80
		80			150					

3 Se actualiza la proxima ciudad a visitar, en este caso, [2] Bilbao

1 Se carga la ciudad actual [2] Bilbao como visitada

2 Se obienen las ciudades adyacentes (en este caso Vigo, Cadizy Jean) y si las distancia para llegar a esas ciudades (60, 130 y 70)
Se actualiza la distancia a [9] Vigo y se les coloca a [2] Bilbao como padre
Las distancias y padres de [3] Cadiz y [5] Jean, no se actualizan porque la distancia seria mayor a la actual



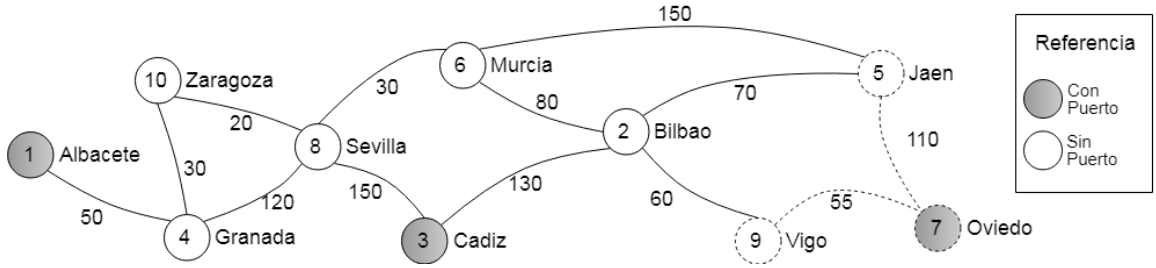
aux = 2 (Bilbao)

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
visitados[]	1	1	0	1	0	1	0	1	0	1
padres1[]	0	6	8	1	6	8	0	10	2	4
dist1[]	0	210	250	50	280	130	999999	100	270	80
			130		70				60	

3 Se actualiza la proxima ciudad a visitar, en este caso, [3] Cadiz

1 Se carga la ciudad actual [3] Cadiz como visitada

2 Cadiz no tiene conexiones a ciudades que no se hayan visitado



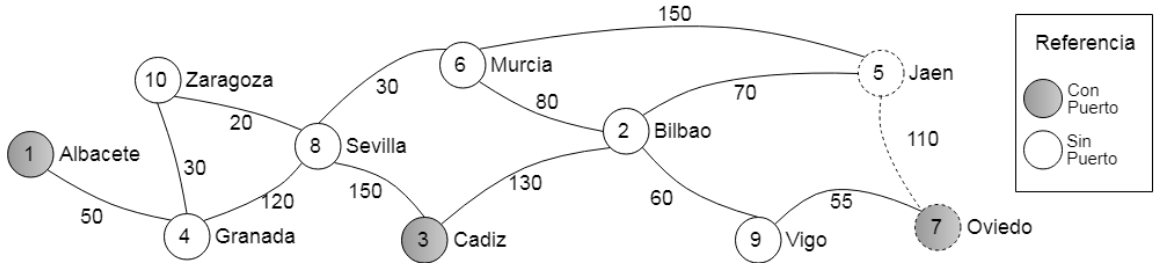
aux = 3 (Cadiz)

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
visitados[]	1	1	1	1	0	1	0	1	0	1
padres1[]	0	6	8	1	6	8	0	10	2	4
dist1[]	0	210	250	50	280	130	999999	100	270	80

3 Se actualiza la proxima ciudad a visitar, en este caso, [9] Vigo

1 Se carga la ciudad actual [9] Vigo como visitada

2 Se obienen las ciudades adyacentes (en este caso Oviedo) y si las distancia para llegar a esas ciudades (55)
Se actualiza la distancia a [7] Oviedo (270+55) y se le coloca a [9] Vigo como padre



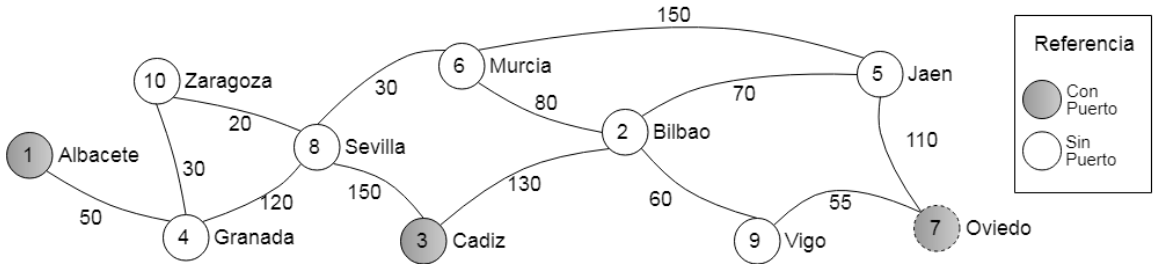
aux = 9 (Vigo)

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
visitados[]	1	1	1	1	0	1	0	1	1	1
padres1[]	0	6	8	1	6	8	9	10	2	4
dist1[]	0	210	250	50	280	130	325	100	270	80
							55			

3 Se actualiza la proxima ciudad a visitar, en este caso, [5] Jean

1 Se carga la ciudad actual [5] Jean como visitada

2 Se obienen las ciudades adyacentes (en este caso Oviedo) y si las distancia para llegar a esas ciudades (110)
La distancia para Ovideo seria mayor a la actual asi que no se actualiza



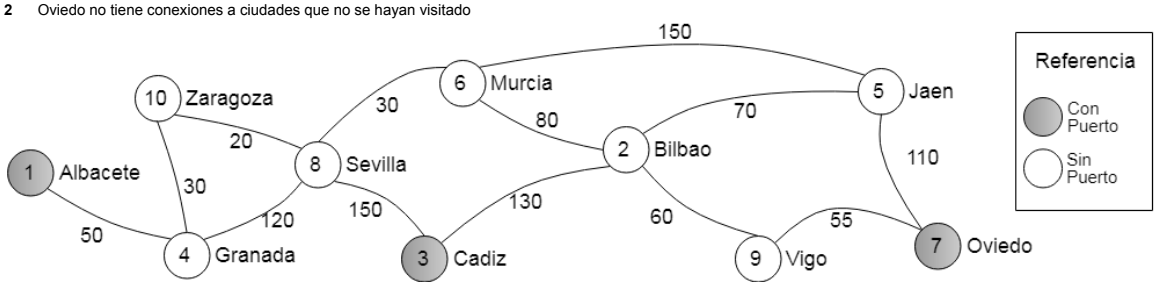
aux = 5 (Jean)

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
visitados[]	1	1	1	1	1	1	0	1	1	1
padres1[]	0	6	8	1	6	8	9	10	2	4
dist1[]	0	210	250	50	280	130	325	100	270	80
							110			

3 Se actualiza la proxima ciudad a visitar, en este caso, [7] Oviedo

1 Se carga la ciudad actual [7] Oviedo como visitada

2 Oviedo no tiene conexiones a ciudades que no se hayan visitado



aux = 7 (Oviedo)

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
visitados[]	1	1	1	1	1	1	1	1	1	1
padres1[]	0	6	8	1	6	8	9	10	2	4
dist1[]	0	210	250	50	280	130	325	100	270	80

3 Ya no hay ciudades sin visitar

Resultado

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
padres1[]	0	6	8	1	6	8	2	10	2	4
dist1[]	0	210	250	50	280	130	325	100	270	80

3 Se cargan los arreglos dist2[] y padres2[] con los caminos minimos para llegar desde cualqueir ciudad al puerto de Cadiz

Resultado

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
padres2[]	4	3	0	10	2	8	9	3	2	8
dist2[]	170	130	0	200	200	180	245	150	190	170

4 Se cargan los arreglos dist3[] y padres3[] con los caminos minimos para llegar desde cualqueir ciudad al puerto de Oviedo

Resultado

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
padres3[]	4	9	2	10	7	2	0	6	7	8
dist3[]	325	115	245	275	110	195	0	225	55	245

5 Se obtiene el id de la ciudad origen, para Sevilla corresponderia el 8

6 Se compara las distancias obtenidas y se imprimi el camino para llegar al mas cercano.

Indice	1	2	3	4	5	6	7	8	9	10
nombres[]	Albacete	Bilbao	Cadiz	Granada	Jaen	Murcia	Oviedo	Sevilla	Vigo	Zaragoza
padres1[]	0	6	8	1	6	8	2	10	2	4
dist1[]	0	210	250	50	280	130	325	100	270	80
padres2[]	4	3	0	10	2	8	9	3	2	8
dist2[]	170	130	0	200	200	180	245	150	190	170
padres3[]	4	9	2	10	7	2	0	6	7	8
dist3[]	325	115	245	275	110	195	0	225	55	245

7 Fin del metodo