Ejercicio planteado

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.

Funciones que intervienen en la solución:

- Main: Se crea el grafo, y se inicializan todos los arreglos necesarios para la solución. Además, se llama a la función que calcula el camino al puerto mas cercano.
- getMenorCaminoPuerto: Se encarga de encontrar el puerto más cercano y retornar una lista con el camino al puerto más cercano.
  Llama a la función getCaminosDijsktra y una vez que tiene los caminos mas cortos desde el vértice de origen a los demás, busca cuales son puerto y buscar cual es el mas cercano. Luego genera la lista de salida con el camino si se encontró.
- getCaminosDijsktra: Se encarga de calcular el camino mínimo desde una ciudad hacia las demás ciudades del grafo. Guarda en el arreglo distancia, la distancia mínima a cada ciudad desde su padre que es guardado en el arreglo padre.
- isPuerto: Devuelve true si es una ciudad puerto y false si no lo es.
- Factible: Verifica que halla un camino valido hasta el vértice, si la distancia es infinita es porque no lo hay.
- getDistancia: Devuelve la distancia desde el vértice a su padre calculada con Dijsktra.
- Solucion: Verifica que la lista tenga elementos lo que significa que hay un camino que es una solución y devuelve true.
- Seleccionar: Selecciona el vértice adyacente que no fue visitado, que está más cercano a su padre.

Pseudocódigo

(Codigo esquema para implementar una solucion en algun lenguaje de programacion)

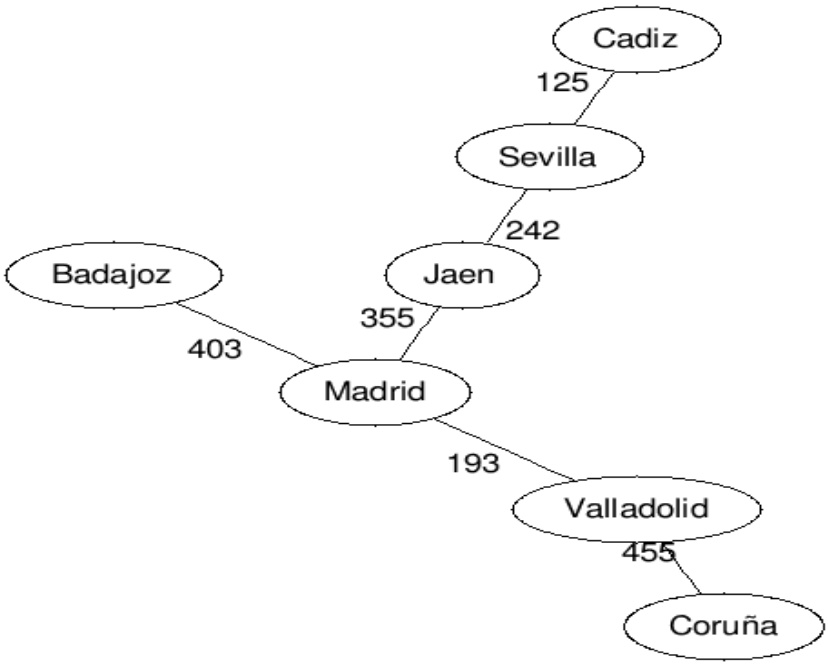
```
//Variables globales
ciudades[]; //Todos los vertices ciudades se almacenan
distancia[]; //En cada posicion pertenceciente
        //a cada ciudad la distancia respecto a la tomada
ciudadVisitada[]; //Se utiliza para el seguimiento de Dijkstra
padre[]; //En cada posicion pertenceciente a cada ciudad
         //Padre con respecto al vertice anterior
ciudadesPuerto[]; //Se almacenan las ciudades que son puertos

Function main()
{
   Grafo g;
   ciudades[] := getCiudades(g); //Carga el arreglo con los nombres de los vertices (ciudades)
   distancia[] := ∞; //Inicializo distancia con ∞
   ciudadVisitada[] := false; //Inicializo ciudadVisitada con false
   padre[] := -1; //Carga el arreglo de padres en -1
   ciudadesPuerto[]  := getCiudadesPuerto(g); //Carga el arreglo con los puertos
   getMenorCaminoPuerto(g,ciudadOrigen);
}

Function getMenorCaminoPuerto(Grafo g, Vertice origen)
{
   // Calcula la ruta mas corta desde la ciudad origen a las demas ciudades del grafo
   getCaminosDijsktra(g,origen);
   distanciaPuerto := 0;
   puertoCercano := null;
   //Recorre todas las ciudades del grafo para encontrar el puerto mas cercano
```

```
    For (cada ciudad en g){
      if (ciudad.isPuerto()){ //Verifica si es un puerto
        if (factible(ciudad)){ //Verifica si hay un camino al Puerto
          //Verifica si la distancia al puerto encontrado es menor a la distancia a otro puerto
           if (distanciaPuerto > getDistancia(origen, ciudad) || (distanciaPuerto = 0)){
             //distanciaPuerto se vuelve la nueva distancia
              distanciaPuerto := getDistancia(origen, ciudad);
              puertoCercano := ciudad; // el puertoCercano se vuelve la ciudad puerto encontrada
           }
        }
      }
    }
    camino := null;
    ciudad := puertoCercano;
    //Comienza por el puerto encontrado y va insertando al principio sus padres para formar el
camino hacia el origen
    while (ciudad != null){
      camino.add(ciudad);
      ciudad := ciudad.getPadre();
    }
    //Si encontro una solucion la devuelve, sino devuelve null
    if (solucion(camino))
      return camino;
    else{
      print("no hay solucion");
      return null;
    }
}
Function factible(Vertice puerto){
```

```
    return distancia[puerto.getNombre()] != ∞;
}
Function solucion(Lista camino){
    return camino.isEmpty();
}
Procedure getCaminosDijsktra(Grafo g,Vertice origen){
    ciudadSinVisitar[]; //cola de prioridad para recorrer las ciudades
    ciudadSinVisitar.add(origen); //agrego el origen a la cola de prioridad
    distancia[origen] := 0; //Coloco como distancia hacia el mismo en 0
    //Recorre todas las ciudades sin visitar y va observando las distancias a sus adyacentes
            while (!ciudadesSinVisitar.isEmpty()){
                     verticeMinimo := seleccionar(ciudadesSinVisitar);
                    ciudadVisitada[verticeMinimo] := true;
                    for(cada ciudad en verticeMinimo.getAdyacentes()){
                                if (ciudadVisitada[ciudad] == false && ciudadSinVisitar[ciudad] == false ){
                                            ciudadSinVisitar.add(ciudad);
                                }
                        distanciaNueva := distancia[verticeMinimo] + getDistancia (verticeMinimo, ciudad);
                        if (distanciaNueva < distancia[ciudad]){
                                distancia[ciudad] := distanciaNueva;
                                padre[ciudad] := verticeMinimo;
                        }
                }
    }
    //Selecciona el vertice que menor distancia tiene de los de la cola de prioridad
    Function seleccionar (aux[]){
        distanciaAux := 0;
        verticeSolucion := null;
        for (cada ciudad en aux)
```

```
        distanciaCiudad := distancia[ciudad];
        if (distanciaAux == 0){
           distanciaAux := distanciaCiudad;
           verticeSolucion := ciudad;
        }
        else{
           if (distanciaAux > distanciaCiudad) {
              distanciaAux := distanciaCiudad;
              verticeSolucion := ciudad;
           }
        }
     }
     return verticeSolucion;
  }
}
```

Seguimiento del algoritmo



Se utilizara el grafo de la figura anterior, considerando que Badajoz, Cadiz y Coruña son puertos.

Inicialización de estructuras:

1) Ciudades

Badajoz	Madrid	Valladolid	Coruña	Jaen	Sevilla	Cadiz

2) Distancia

∞	∞	∞	∞	∞	∞	∞

3) ciudadVisitada

False	False	False	False	False	False	False

4) Padre

-1	-1	-1	-1	-1	-1	-1

5) ciudadesPuerto

Badajoz	Coruña	Cadiz

<u>Seguimiento Dijkstra</u>

1er columna es ciudades

2da columna es distancia

3ra columna es ciudadVisitada

4ta columna es padre

5ta columna es la cola de prioridades

Se considera para este seguimiento como punto de origen la ciudad de Madrid

Iteracion 1	Badajoz	Madrid	Valladolid	Coruña	Jaen	Sevilla	Cadiz
	403	0	193	∞	355	∞	∞
	False	False	False	False	False	False	false
	-1	-1	-1	-1	-1	-1	-1
	Madrid						

Iteracion 2	Badajoz	Madrid	Valladolid	Coruña	Jaen	Sevilla	Cadiz
	403	0	193	455	355	∞	∞
	False	True	True	False	False	False	false
	-1	-1	Madrid	-1	-1	-1	-1

Iteracion	Badajoz	Madrid	Valladolid	Coruña	Jaen	Sevilla	Cadiz

3	403	0	193	455	355	∞	∞
	False	True	True	True	False	False	false
	-1	-1	Madrid	Valladolid	-1	-1	-1

Iteracion 4	Badajoz	Madrid	Valladolid	Coruña	Jaen	Sevilla	Cadiz
	403	0	193	455	355	242	∞
	False	True	True	True	True	False	false
	-1	-1	Madrid	Valladolid	Madrid	-1	-1

Iteracion 5	Badajoz	Madrid	Valladolid	Coruña	Jaen	Sevilla	Cadiz
	403	0	193	455	355	242	125
	False	True	True	True	True	True	false
	-1	-1	Madrid	Valladolid	Madrid	Jaen	-1

Iteracion 5	Badajoz	Madrid	Valladolid	Coruña	Jaen	Sevilla	Cadiz
	403	0	193	455	355	242	125
	False	True	True	True	True	True	True
	-1	-1	Madrid	Valladolid	Madrid	Jaen	Sevilla

Iteracion 6	Badajoz	Madrid	Valladolid	Coruña	Jaen	Sevilla	Cadiz
	403	0	193	455	355	242	125
	True	True	True	True	True	True	True
	Madrid	-1	Madrid	Valladolid	Madrid	Jaen	Sevilla

Luego se calcula la distancia teniendo en cuenta los padres y se puede ver que Badajoz queda a una distancia 403 de Madrid y el camino es directo, Coruña queda a una distancia 648 de Madrid y recorre Madrid, Valladolid y Coruña, y Cadiz queda a una distancia 722 de Madrid y recorre Madrid, Jaen, Sevilla y Cadiz. Por lo tanto para este ejemplo el camino al puerto mas cercano desde Madrid es el puerto de Badajoz y el camino resultante es {Madrid, Badajoz}.