public class CaminoSinControles{

private Grafo<String> mapa;

public ListaGenerica<ListaGenerica<String>> devolverCaminos(String origen, String destino){

ListaGenerica<ListaGenerica<String>> caminos = new ListaEnlazadaGenerica<ListaGenerica<String>>();

ListaGenerica<String> caminoActual = new ListaEnlazadaGenerica<String>();

ListaGenerica<Vertice<String>> ciudades = mapa.listaDeVertices();

boolean[] visitados = new boolean[ciudades.tamanio()+1];

// Busco la ciudad de origen

ciudades.comenzar();

while(!ciudades.fin()){

Vertice<String> ciudad = ciudades.proximo();

if(ciudad.dato().equals(origen)){

this.dfs(ciudad, destino, visitados, caminoActual, caminos);

break;//Corto el while

}

}

return caminos;

}

private void dfs(Vertice<String> origen, String destino, boolean[] visitados, ListaGenerica<String> caminoActual, ListaGenerica<ListaGenerica<String>> resultados){

// Agrego la ciudad a visitados y al camino actual

visitados[origen.posicion()] = true;

caminoActual.agregarFinal(origen.dato());

// Si llego a destino me guardo el camino con el que llegue en los resultados

if(origen.dato().equals(destino)){

resultados.agregarFinal(caminoActual.clonar());

}

else{

ListaGenerica<Arista<String>> adyacentes = this.mapa.listaDeAdyacentes(origen);

Arista<String> adyacente;

// Recorro cada adyacente si no fue ya visitado y no hay controles policiales

adyacentes.comenzar();

while(!adyacentes.fin()){

adyacente = adyacentes.proximo();

if(!visitados[adyacente.verticeDestino().posicion()] && adyacente.peso() == 0){

this.dfs(adyacente.verticeDestino(), destino, visitados, caminoActual, resultados);

}

}

}

// Desmarco y desarmo el camino para que otras llamadas al dfs puedan pasar por la ciudad.

caminoActual.eliminar(caminoActual.tamanio());

visitados[origen.posicion()] = false;

}

}