Resolución Segunda Entrega Grafos
Turno Jueves 10 hs

```
public class BuscadorDeCaminos {

   private ListaGenerica<String> camino;
   /*
    * camino: una variable de instancia para mantener el camino
    * al volver de la recursión.
    */


   public ListaGenerica<String> recorridoQuePasePorAlMenos5(Grafo<String> ciudades){

            ListaGenerica<Vertice<String>> vertices = ciudades.listaDeVertices();
            boolean seguir = true;

            ListaGenerica<String> caminoActual = new ListaEnlazadaGenerica<String>();
            /*
             * caminoActual es la lista donde vamos guardando el camino actual
             */
            camino = new ListaEnlazadaGenerica<String>();
            Vertice<String> vIni = null;

            vertices.comenzar();
            /*
             * Se busca el vertice de inicio donde se dispara el recorrido.
             * Sabemos que La Plata existe
             */
            while(!vertices.fin() && seguir) {
                     Vertice<String> v = vertices.proximo();
                     if (v.dato().equals ("La Plata"))
                              vIni = v;

                     if (vIni != null)
                              seguir = false;
            }
            recorridoQuePasePorAlMenos5(ciudades, vIni, caminoActual);

            return camino;

   }


   private void recorridoQuePasePorAlMenos5(Grafo<String> ciudades, Vertice<String> vertice, ListaGenerica<String>
caminoActual) {

            caminoActual.agregarFinal (vertice.dato());
            if (vertice.dato()=="Madariaga") {
                     if (caminoActual.tamanio()>=5) {
                              this.camino = caminoActual.clonar(); // Actualiza el camino
                     }

            }
            else {
                     ListaGenerica<Arista<String>> adyacentes = ciudades.listaDeAdyacentes(vertice);
                     adyacentes.comenzar ();
                     while (!adyacentes.fin()) {
                              Arista<String> arista = adyacentes.proximo ();
```

```
                                    if (!caminoActual.incluye (arista.verticeDestino().dato())) { // Si el vertice al que apunta la arista está la lista es porque ya lo visité, es una alternativa para no usar un arreglo de booleanos
                                                if (arista.verticeDestino().dato()!= "Tandil"){ // Si es Tandil no sigo recorriendo
                                                            recorridoQuePasePorAlMenos5(ciudades, arista.verticeDestino (), caminoActual);

                                                }
                                    }
                        }


            }
            caminoActual.eliminarEn(caminoActual.tamanio());
    }
```

Si se quiere probar...

```
    public static void main(String[] args) {
            /*
             * Se crea el grafo
             */
            Grafo<String> ciudades = new GrafoImplListAdy<String>();
            Vertice<String> laplata=new VerticeImplListAdy<String>("La Plata");
            Vertice<String> tandil=new VerticeImplListAdy<String>("Tandil");
            Vertice<String> lezama=new VerticeImplListAdy<String>("Lezama");
            Vertice<String> pila=new VerticeImplListAdy<String>("Pila");
            Vertice<String> mardelplata=new VerticeImplListAdy<String>("Mar Del Plata");
            Vertice<String> villagesell=new VerticeImplListAdy<String>("Villa Gesell");
            Vertice<String> pinamar=new VerticeImplListAdy<String>("Pinamar");
            Vertice<String> mardeajo=new VerticeImplListAdy<String>("Mar de Ajó");
            Vertice<String> madariaga=new VerticeImplListAdy<String>("Madariaga");
            ciudades.agregarVertice(laplata);
            ciudades.agregarVertice(tandil);
            ciudades.agregarVertice(lezama);
            ciudades.agregarVertice(pila);
            ciudades.agregarVertice(mardelplata);
            ciudades.agregarVertice(villagesell);
            ciudades.agregarVertice(pinamar);
            ciudades.agregarVertice(mardeajo);
            ciudades.agregarVertice(madariaga);
            ciudades.conectar(laplata, lezama);
            ciudades.conectar(lezama, pila);
            ciudades.conectar(lezama, mardeajo);
            ciudades.conectar(pila, mardeajo);
            ciudades.conectar(mardeajo,madariaga);
            ciudades.conectar(laplata,tandil );
            ciudades.conectar(tandil, mardelplata);
            ciudades.conectar(mardelplata, villagesell);
            ciudades.conectar(villagesell, pinamar);
            ciudades.conectar(pinamar, madariaga);
            ciudades.conectar(tandil, madariaga);
            ciudades.conectar(pila,tandil);
            /*
             * Se crea instancia de buscador con el grafo creado
             */
            BuscadorDeCaminos buscador = new BuscadorDeCaminos();
            ListaGenerica<String> recorrido = buscador.recorridoQuePasePorAlMenos5(ciudades);
            /* Se imprimen los valores del recorrido obtenido
             *
             */
            recorrido.comenzar();
```

```
        while(!recorrido.fin()){
                System.out.print(recorrido.proximo() + " - ");
        }



    }

}
```