# ALGORITMOS Y ESTRUCTURA DE DATOS 2

## Documentación - Obligatorio

Andrés Martínez

135526

Sebastián Rodríguez

181227

N1A

María Florencia Polcaro

15/06/2016

Índice

[ESTRUCTURA DE DATOS Y ALGORITMOS 1](#_Toc359097655)

[Obligatorio 1](#_Toc359097656)

[Interfaz Sistema: Pre y post condiciones 3](#_Toc359097657)

[Solución escogida 5](#_Toc359097658)

[Diagrama de la estructura de datos 5](#_Toc359097659)

[Justificación 6](#_Toc359097660)

[Testing 7](#_Toc359097661)

# Interfaz Sistema: Pre y post condiciones

**//Precondiciones:** la cantidad de puntos debe ser mayor a 0.

**//Postcondiciones:** el sistema se inicializa.

**public** Retorno **inicializarSistema**(int cantPuntos)

**//Precondiciones:** no tiene.

**//Postcondiciones:** se libera la memoria utilizada.

**public** Retorno **destruirSistema**()

**//Precondiciones:** la cedula, el mail y el celular deben tener formato correcto y no debe existir un productor con la misma cedula ya ingresado al sistema.

**//Postcondiciones:** el productor queda registrado en el sistema.

**public** Retorno **registrarProductor**(String cedula, String nombre, String direccion, String email, String celular)

**//Precondiciones:** debe haber puntos disponibles en el sistema para registrar la ciudad y no debe existir una ciudad registrada con las mismas coordenadas.

**//Postcondiciones:** la ciudad queda registrada en el sistema.

**public** Retorno **registrarCiudad**(String nombre, Double coordX, Double coordY)

**//Precondiciones:**debe haber puntos disponibles en el sistema para ingresar el tambo, la capacidad debe ser mayor o igual a 0, las coordenadas no deben existir previamente en el sistema,debe existir un productor ya registrado en el sistema con esa cedula.

**//Postcondiciones:** el tambo queda registrado en el sistema

**public** Retorno **registrarTambo**(String nombre, Double coordX, Double coordY, String cedula\_productor, int capacidad)

**//Precondiciones:** debe haber puntos disponibles en el sistema para ingresar el centro, la capacidad debe ser mayor que 0, las coordenadas no deben existir previamente en el sistema.

**//Postcondiciones:** el centroqueda registrado en el sistema.

**public** Retorno **registrarCentro**(String nombre, Double coordX, Double coordY, int capacidad)

**//Precondiciones:** las coordenadas ingresadas deben existir en el sistema, no debe existir ese tramo ya ingresado, el peso debe ser mayor a 0.

**//Postcondiciones:** el tramo queda ingresado en el sistema.

**public** Retorno **registrarTramo**(Double coordXi, Double coordYi, Double coordXf, Double coordYf, int peso) {

**//Precondiciones:** deben existir puntos ingresados al sistema con las coordenadas ingresadas, debe existir un tramo ingresado previamente en el sistema entre las coordenadas ingresadas.

**//Postcondiciones:** el tramo entre las coordenadas queda registrado en el sistema.

**public** Retorno **eliminarTramo**(Double coordXi, Double coordYi, Double coordXf, Double coordYf)

**//Precondiciones:** debe existir un punto con las coordenadas ingresadas.

**//Postcondiciones:** se quita del sistema el punto registrado con esas coordenadas.

**public** Retorno **eliminarPunto**(Double coordX, Double coordY)

**//Precondiciones:** no tiene

**//Postcondiciones:** se ubican los puntos registrados en el sistema asi como los tramos ingresados entre los puntos.

**public** Retorno **mapaEstado()**

**//Precondiciones:** el tambo debe estar registrado con las coordenadas indicadas, debe existir algún centro registrado con la necesidad de pasteurización del tambo.

**//Postcondiciones:** devuelve las coordenadas correspondientes al camino más cercano desde el tambo al centro de pasteurización.

**public** Retorno **rutaACentroMasCercano**(Double coordX, Double coordY)

**//Precondiciones:** debe existir una ciudad ingresada al sistema con las coordenadas ingresadas.

**//Postcondiciones:** devuelve el listado de los tambos que se encuentran en la ciudad.

**public** Retorno **listadoDeTambosEnCiudad**(Double coordX, Double coordY)

**//Precondiciones:** no tiene.

**//Postcondiciones:** devuelve el listado de los centros de pasteurización ingresados al sistema.

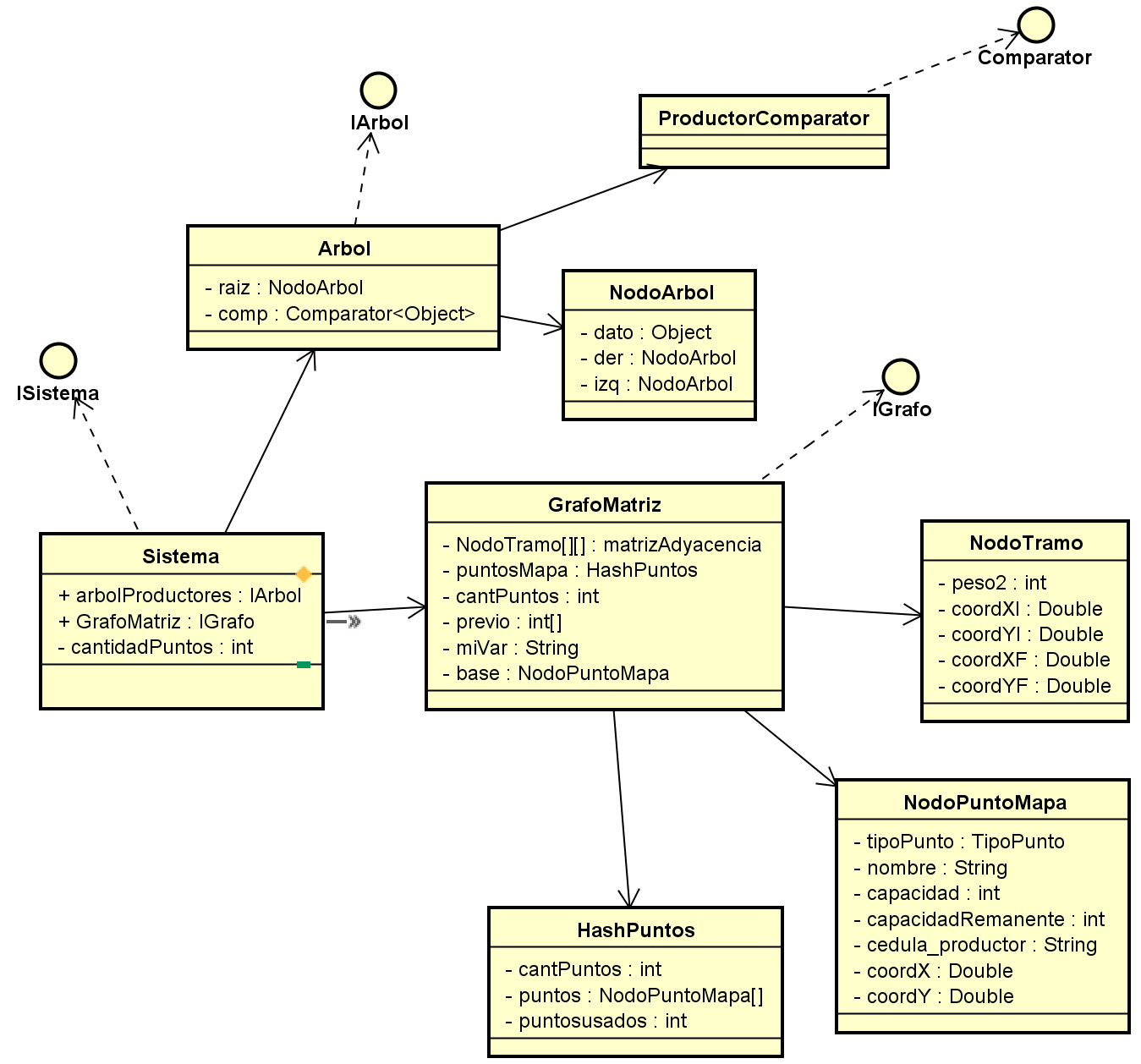
**public** Retorno **listadoDeCentros**

**//Precondiciones:** no tiene.

**//Postcondiciones:** devuelve el listado de los productores ingresados al sistema.

**public** Retorno **listadoProductores**()

## Diagrama de la estructura de datos



## Justificación

Justificación de las estructuras elegidas para modelar las entidades del problema.

|  |  |
| --- | --- |
| **PRODUCTOR** | |
| Estructura | Árbol binario de búsqueda por cedula de identidad. |
| Justificación | Árbol binario de búsqueda: esta estructura nos permite obtener de forma eficiente los productores a través del número de cedula, de esta forma se realiza en orden (log n) promedio. |

|  |  |
| --- | --- |
| **PUNTO** | |
| Estructura | Grafo  Hash de Puntos |
| Justificación | Grafo: para representar las conexiones entre los puntos se implementa un hash de puntos, matriz de adyacencia para obtener las distancias entre los puntos.  Hash de Puntos: se obtiene de forma eficiente la referencia a todos los puntos del grafo a través de las coordenadas.  Matriz de adyacencia: nos permite chequear rápidamente el tramo entre dos puntos de forma simple.  Vector de enteros: se utiliza un vector de enteros para ir almacenando los puntos ya visitados al recorrer el grafo. |

## Testing

@Test

public void inicializarSistema() {

assertEquals(Sistema.TipoError.OK, s.inicializarSistema(10).getResultado());

}

@Test

public void inicializarSistema\_Error1() {

assertEquals(Sistema.TipoError.ERROR\_1, s.inicializarSistema(-1).getResultado());

}

@Test

public void destruirSistema() {

assertEquals(Sistema.TipoError.OK, s.destruirSistema().getResultado());

}

@Test

public void registrarProductor() {

s.inicializarSistema(8);

assertEquals(Sistema.TipoError.OK, s.registrarProductor("4.624.303-3", "Luis", "25 de Mayo", "luis@gmail.com", "098123456").getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarProductor("4.624.303-4", "Sebastian", "Gral Rivera 722", "srodriguez@gmail.com", "099123456").getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarProductor("4.624.303-2", "Juan", "18 de Julio 221", "juan@gmail.com", "099123456").getResultado());

}

//TEST registroProductor\_ERROR1 - Cedula no tiene formato correcto

@Test

public void registrarProductor\_ERROR1() {

s.inicializarSistema(8);

assertEquals(Sistema.TipoError.ERROR\_1, s.registrarProductor("3.7322", "Andres", "Cuareim 1212", "andresm@gmail.com", "099232111").getResultado());

assertEquals(Sistema.TipoError.ERROR\_1, s.registrarProductor("3373250", "Andres", "Cuareim 1212", "andresm@gmail.com", "099232111").getResultado());

assertEquals(Sistema.TipoError.ERROR\_1, s.registrarProductor("", "Sebastian", "Gral Rivera 722", "srodriguez@gmail.com", "099123456").getResultado());

assertEquals(Sistema.TipoError.ERROR\_1, s.registrarProductor("aaaaa", "Sebastian", "Gral Rivera 722", "srodriguez@gmail.com", "099123456").getResultado());

}

//TEST registroProductor\_ERROR2 - Celular no tiene formato correcto 09

@Test

public void registrarProductor\_ERROR2() {

s.inicializarSistema(8);

assertEquals(Sistema.TipoError.ERROR\_2, s.registrarProductor("3.732.570-2", "Andres", "Cuareim 1212", "andresm@gmail.com", "24035512").getResultado());

assertEquals(Sistema.TipoError.ERROR\_2, s.registrarProductor("4.732.570-3", "Martin", "Vazquez 1299", "martinz@gmail.com", "").getResultado());

assertEquals(Sistema.TipoError.ERROR\_2, s.registrarProductor("4.732.570-4", "Estaban", "Avda Bolivia 1421", "esteban@gmail.com", "a").getResultado());

assertEquals(Sistema.TipoError.ERROR\_2, s.registrarProductor("4.732.570-5", "Marcelo", "Rambla Ghandi 766", "marcelot@gmail.com", "001").getResultado());

}

//TEST registroProductor\_ERROR3 - Mail no tiene formato correcto

@Test

public void registrarProductor\_ERROR3() {

s.inicializarSistema(8);

assertEquals(Sistema.TipoError.ERROR\_3, s.registrarProductor("4.624.303-4", "Sebastian", "Gral Rivera 722", "srodriguezgmail.com", "099123456").getResultado());

assertEquals(Sistema.TipoError.ERROR\_3, s.registrarProductor("4.624.303-2", "Juan", "18 de Julio 221", "", "099123456").getResultado());

assertEquals(Sistema.TipoError.ERROR\_3, s.registrarProductor("4.624.303-3", "Luis", "25 de Mayo", "11111", "098123456").getResultado());

}

//TEST registroProductor\_ERROR4 - Productor ya existe

@Test

public void registrarProductor\_ERROR4() {

s.inicializarSistema(8);

assertEquals(Sistema.TipoError.OK, s.registrarProductor("3.732.570-2", "Andres", "Cuareim 1212", "andresm@gmail.com", "099232111").getResultado());

assertEquals(Sistema.TipoError.ERROR\_4, s.registrarProductor("3.732.570-2", "Andres", "Cuareim 1212", "andresm@gmail.com", "099232111").getResultado());

}

//TEST Registro Ciudad

@Test

public void registrarCiudad() {

s.inicializarSistema(8);

assertEquals(Sistema.TipoError.OK, s.registrarCiudad("Montevideo", -34.83346, -56.16735).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarCiudad("Libertad", -34.63333, -56.61917).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarCiudad("Piriapolis", -34.86819, -55.27436).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarCiudad("La Paloma", -34.66667, -54.16667).getResultado());

}

//TEST Registro Ciudad\_ERROR1 - Cargo mas ciudades que las permitidas

@Test

public void registrarCiudad\_ERROR1() {

s.inicializarSistema(3);

assertEquals(Sistema.TipoError.OK, s.registrarCiudad("Montevideo", -34.83346, -56.16735).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarCiudad("Libertad", -34.63333, -56.61917).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarCiudad("Piriapolis", -34.86819, -55.27436).getResultado());

assertEquals(Sistema.TipoError.ERROR\_1, s.registrarCiudad("La Paloma", -34.66667, -54.16667).getResultado());

}

//TEST Registro Ciudad\_ERROR2 - Ciudad ya existe

@Test

public void registrarCiudad\_ERROR2() {

s.inicializarSistema(8);

assertEquals(Sistema.TipoError.OK, s.registrarCiudad("Montevideo", -34.83346, -56.16735).getResultado());

assertEquals(Sistema.TipoError.ERROR\_2, s.registrarCiudad("Montevideo", -34.83346, -56.16735).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarCiudad("Piriapolis", -34.86819, -55.27436).getResultado());

assertEquals(Sistema.TipoError.ERROR\_2, s.registrarCiudad("Piriapolis", -34.86819, -55.27436).getResultado());

}

//TEST Registro Tambos

@Test

public void registrarTambo() {

s.inicializarSistema(8);

assertEquals(Sistema.TipoError.OK, s.registrarTambo("Tambo Central", -33.41306, -56.50056, "4.624.303-4", 15).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarTambo("Tambo Artigas", -31.08333, -56.01667, "4.624.303-2", 40).getResultado());

}

//TEST Registro Tambos\_ERROR1 - Mas tambos que puntos permitidos

@Test

public void registrarTambo\_ERROR1() {

s.inicializarSistema(4);

assertEquals(Sistema.TipoError.OK, s.registrarTambo("Tambo Central", -33.41306, -56.50056, "4.624.303-4", 15).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarTambo("Tambo Artigas", -31.08333, -56.01667, "4.624.303-2", 20).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarTambo("Tambo Soriano", -33.25583, -58.01917, "4.624.303-3", 10).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarTambo("Tambo Santa Clara", -32.91667, -54.96667, "4.624.303-3", 60).getResultado());

assertEquals(Sistema.TipoError.ERROR\_1, s.registrarTambo("Tambo Flores", -33.53889, -56.88861, "4.624.303-2", 15).getResultado());

}

//TEST Registro Tambos\_ERROR3 - El tambo ya existe

@Test

public void registrarTambo\_ERROR3() {

s.inicializarSistema(8);

assertEquals(Sistema.TipoError.OK, s.registrarTambo("Tambo Central", -33.41306, -56.50056, "4.624.303-4", 15).getResultado());

assertEquals(Sistema.TipoError.ERROR\_3, s.registrarTambo("Tambo Central", -33.41306, -56.50056, "4.624.303-4", 15).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarTambo("Tambo Artigas", -31.08333, -56.01667, "4.624.303-2", 20).getResultado());

assertEquals(Sistema.TipoError.ERROR\_3, s.registrarTambo("Tambo Artigas", -31.08333, -56.01667, "4.624.303-2", 20).getResultado());

}

//TEST Registro Centros de Distribucion

@Test

public void registrarCentroDistribucion() {

s.inicializarSistema(8);

assertEquals(Sistema.TipoError.OK, s.registrarCentro("Joanico", -34.58944, -56.25278, 18).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarCentro("Florida", -34.09556, -56.21417, 28).getResultado());

}

//TEST Registro Centros de Distribucion\_ERROR1 - Mas centros que puntos permitidos

@Test

public void registrarCentroDistribucion\_ERROR1() {

s.inicializarSistema(3);

assertEquals(Sistema.TipoError.OK, s.registrarCentro("Joanico", -34.58944, -56.25278, 18).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarCentro("Florida", -34.09556, -56.21417, 28).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarCentro("Centro Soriano", -33.25583, -58.01917, 10).getResultado());

assertEquals(Sistema.TipoError.ERROR\_1, s.registrarCentro("Tambo Santa Clara", -32.91667, -54.96667, 60).getResultado());

assertEquals(Sistema.TipoError.ERROR\_1, s.registrarCentro("Tambo Flores", -33.53889, -56.88861, 15).getResultado());

}

//TEST Registro Centros de Distribucion\_ERROR2 - Centro de distribucion repetidos

@Test

public void registrarCentroDistribucion\_ERROR2() {

s.inicializarSistema(3);

assertEquals(Sistema.TipoError.OK, s.registrarCentro("Joanico", -34.58944, -56.25278, 18).getResultado());

assertEquals(Sistema.TipoError.ERROR\_3, s.registrarCentro("Joanico", -34.58944, -56.25278, 18).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarCentro("Florida", -34.09556, -56.21417, 28).getResultado());

assertEquals(Sistema.TipoError.ERROR\_3, s.registrarCentro("Florida", -34.09556, -56.21417, 28).getResultado());

}

//TEST Registro Tramo

@Test

public void registrarTramo() {

s.inicializarSistema(8);

s.registrarCiudad("Montevideo", -34.83346, -56.16735);

s.registrarCiudad("Libertad", -34.63333, -56.61917);

s.registrarCiudad("Piriapolis", -34.86819, -55.27436);

s.registrarCiudad("La Paloma", -34.66667, -54.16667);

s.registrarCentro("Joanico", -34.58944, -56.25278, 18);

assertEquals(Sistema.TipoError.OK, s.registrarTramo(-34.83346, -56.16735, -34.63333, -56.61917, 1).getResultado()); //Mdo Libertad

assertEquals(Sistema.TipoError.OK, s.registrarTramo(-34.83346, -56.16735, -34.86819, -55.27436, 9).getResultado()); //Mdo Piriapolis

assertEquals(Sistema.TipoError.OK, s.registrarTramo(-34.83346, -56.16735, -34.66667, -54.16667, 5).getResultado()); //Mdo La Paloma

assertEquals(Sistema.TipoError.OK, s.registrarTramo(-34.83346, -56.16735, -34.58944, -56.25278, 1).getResultado()); //Mdo Joanico

}

//TEST Registrar TRamo\_ERROR1 - Peso < 0

@Test

public void registrarTramo\_ERROR1() {

s.inicializarSistema(8);

assertEquals(Sistema.TipoError.ERROR\_1, s.registrarTramo(-34.83346, -56.16735, -34.63333, -56.61917, -1).getResultado()); //Mdo Libertad

assertEquals(Sistema.TipoError.ERROR\_1, s.registrarTramo(-34.83346, -56.16735, -34.86819, -55.27436, -9).getResultado()); //Mdo Piriapolis

}

//TEST Eliminar Tramo

@Test

public void eliminarTramo() {

s.inicializarSistema(8);

s.registrarCiudad("Montevideo", -34.83346, -56.16735);

s.registrarCiudad("Libertad", -34.63333, -56.61917);

s.registrarCiudad("Piriapolis", -34.86819, -55.27436);

assertEquals(Sistema.TipoError.OK, s.registrarTramo(-34.83346, -56.16735, -34.63333, -56.61917, 1).getResultado()); //Mdo Libertad

assertEquals(Sistema.TipoError.OK, s.eliminarTramo(-34.83346, -56.16735, -34.63333, -56.61917).getResultado()); //Mdo Libertad

assertEquals(Sistema.TipoError.OK, s.registrarTramo(-34.83346, -56.16735, -34.86819, -55.27436, 9).getResultado()); //Mdo Piriapolis

assertEquals(Sistema.TipoError.OK, s.eliminarTramo(-34.83346, -56.16735, -34.86819, -55.27436).getResultado()); //Mdo Piriapolis

}

//TEST Eliminar Tramo\_ERROR1 - Si las coordenadas no existen ( no fueron cargadas previamente)

@Test

public void eliminarTramo\_ERROR1() {

s.inicializarSistema(8);

//En este caso tienen que dar el error ya que no fueron cargadas estas coordenadas, deberiamos ingresar otras para las pruebas? Consultar Prof.

assertEquals(Sistema.TipoError.ERROR\_1, s.eliminarTramo(-34.83346, -56.16735, -34.63333, -56.61917).getResultado()); //Mdo Libertad

assertEquals(Sistema.TipoError.ERROR\_1, s.eliminarTramo(-34.83346, -56.16735, -34.86819, -55.27436).getResultado()); //Mdo Piriapolis

}

//TEST Eliminar Tramo\_ERROR2 - Si los tramos no existen

@Test

public void eliminarTramo\_ERROR2() {

s.inicializarSistema(8);

//En este caso estos tramos son inexistentes, nunca se cargaron y tienen coordenadas que no corresponden a los ejemplos previamente cargados.

assertEquals(Sistema.TipoError.ERROR\_1, s.eliminarTramo(-34.83346, -56.16735, -34.86819, -34.86819).getResultado()); //Mdo Libertad

assertEquals(Sistema.TipoError.ERROR\_1, s.eliminarTramo(-56.61917, -56.16735, -34.86819, -55.27436).getResultado()); //Mdo Piriapolis

}

//TEST Eliminar Punto , es necesario probar la eliminacion de cada tipo de punto? Ciudad, centro y tambo ?

@Test

public void eliminarPunto() {

s.inicializarSistema(8);

assertEquals(Sistema.TipoError.OK, s.registrarCiudad("Montevideo", -34.83346, -56.16735).getResultado());

assertEquals(Sistema.TipoError.OK, s.eliminarPunto(-34.83346, -56.16735).getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarCiudad("Libertad", -34.63333, -56.61917).getResultado());

assertEquals(Sistema.TipoError.OK, s.eliminarPunto(-34.63333, -56.61917).getResultado());

}

//TEST Eliminar Punto\_ERROR1 - Intento eliminar puntos que no fueron cargados previamente, o sea que no existen.

@Test

public void eliminarPunto\_ERROR1() {

s.inicializarSistema(8);

assertEquals(Sistema.TipoError.ERROR\_1, s.eliminarPunto(-34.83346, -56.16735).getResultado());

assertEquals(Sistema.TipoError.ERROR\_1, s.eliminarPunto(-34.63333, -56.61917).getResultado());

}

@Test

public void mapaEstado() {

s.inicializarSistema(11);

assertEquals(Sistema.TipoError.OK, s.mapaEstado().getResultado());

}

@Test

public void rutaACentroMasCercano() {

s.inicializarSistema(11);

s.rutaACentroMasCercano(-33.41306, -56.50056);

s.registrarCiudad("Montevideo", -34.83346, -56.16735).getResultado();

s.registrarCiudad("Libertad", -34.63333, -56.61917).getResultado();

s.registrarCiudad("Piriapolis", -34.86819, -55.27436).getResultado();

s.registrarCiudad("La Paloma", -34.66667, -54.16667).getResultado();

s.registrarCentro("Joanico", -34.58944, -56.25278, 18).getResultado();

s.registrarCentro("Florida", -34.09556, -56.21417, 14).getResultado();

s.registrarTambo("Tambo Central", -33.41306, -56.50056, "4.624.303-4", 15).getResultado();

s.registrarTambo("Tambo Artigas", -31.08333, -56.01667, "4.624.303-2", 20).getResultado();

s.registrarTambo("Tambo Soriano", -33.25583, -58.01917, "4.624.303-3", 10).getResultado();

s.registrarTambo("Tambo Santa Clara", -32.91667, -54.96667, "4.624.303-3", 60).getResultado();

s.registrarTambo("Tambo Flores", -33.53889, -56.88861, "4.624.303-2", 15).getResultado();

s.registrarTramo(-31.08333, -56.01667, -33.41306, -56.50056, 10); //tambo artigas a tambo central

s.registrarTramo(-33.41306, -56.50056, -33.53889, -56.88861, 8); //tambo central a tambo flores

s.registrarTramo(-32.91667, -54.96667, -34.09556, -56.21417, 5); //tambo santa clara a centro florida

s.registrarTramo(-34.09556, -56.21417, -34.66667, -54.16667, 5); //centro florida a la paloma

s.registrarTramo(-34.58944, -56.25278, -34.09556, -56.21417, 20); //centro joanico a centro florida

s.registrarTramo(-31.08333, -56.01667, -34.58944, -56.25278, 30); //tambo artigas centro joanico

s.registrarTramo(-34.83346, -56.16735, -31.08333, -56.01667, 60); //montevideo tambo artigas

s.registrarTramo(-33.53889, -56.88861, -32.91667, -54.96667, 2); //tambo flores tambo santa clara

s.registrarTramo(-34.66667, -54.16667, -34.86819, -55.27436, 2); //la paloma a priapolis

s.registrarTramo(-34.86819, -55.27436, -34.63333, -56.61917, 12); //priapolis libertad

s.registrarTramo(-34.63333, -56.61917, -34.83346, -56.16735, 4); //libertad a montevideo

s.registrarTramo(-34.63333, -56.61917, -32.91667, -54.96667, 6); //libertad a santa clara

s.registrarTramo(-33.53889, -56.88861, -34.83346, -56.16735, 19); //tambo flores a montevideo

s.registrarTramo(-33.25583, -58.01917, -34.83346, -56.16735, 30); //tambo soriano a montevideo

assertEquals("-33.41306;-56.50056|-33.53889;-56.88861|-32.91667;-54.96667|-34.09556;-56.21417|", s.rutaACentroMasCercano(-33.41306, -56.50056).getValorString());

assertEquals(Sistema.TipoError.OK, s.rutaACentroMasCercano(-33.41306, -56.50056).getResultado());

}

@Test

public void listadoDeTambosEnCiudad() {

s.inicializarSistema(11);

s.rutaACentroMasCercano(-33.41306, -56.50056);

s.registrarCiudad("Montevideo", -34.83346, -56.16735).getResultado();

s.registrarCiudad("Libertad", -34.63333, -56.61917).getResultado();

s.registrarCiudad("Piriapolis", -34.86819, -55.27436).getResultado();

s.registrarCiudad("La Paloma", -34.66667, -54.16667).getResultado();

s.registrarCentro("Joanico", -34.58944, -56.25278, 18).getResultado();

s.registrarCentro("Florida", -34.09556, -56.21417, 14).getResultado();

s.registrarTambo("Tambo Central", -33.41306, -56.50056, "4.624.303-4", 15).getResultado();

s.registrarTambo("Tambo Artigas", -31.08333, -56.01667, "4.624.303-2", 20).getResultado();

s.registrarTambo("Tambo Soriano", -33.25583, -58.01917, "4.624.303-3", 10).getResultado();

s.registrarTambo("Tambo Santa Clara", -32.91667, -54.96667, "4.624.303-3", 60).getResultado();

s.registrarTambo("Tambo Flores", -33.53889, -56.88861, "4.624.303-2", 15).getResultado();

s.registrarTramo(-31.08333, -56.01667, -33.41306, -56.50056, 10); //tambo artigas a tambo central

s.registrarTramo(-33.41306, -56.50056, -33.53889, -56.88861, 8); //tambo central a tambo flores

s.registrarTramo(-32.91667, -54.96667, -34.09556, -56.21417, 5); //tambo santa clara a centro florida

s.registrarTramo(-34.09556, -56.21417, -34.66667, -54.16667, 5); //centro florida a la paloma

s.registrarTramo(-34.58944, -56.25278, -34.09556, -56.21417, 20); //centro joanico a centro florida

s.registrarTramo(-31.08333, -56.01667, -34.58944, -56.25278, 30); //tambo artigas centro joanico

s.registrarTramo(-34.83346, -56.16735, -31.08333, -56.01667, 60); //montevideo tambo artigas

s.registrarTramo(-33.53889, -56.88861, -32.91667, -54.96667, 2); //tambo flores tambo santa clara

s.registrarTramo(-34.66667, -54.16667, -34.86819, -55.27436, 2); //la paloma a priapolis

s.registrarTramo(-34.86819, -55.27436, -34.63333, -56.61917, 12); //priapolis libertad

s.registrarTramo(-34.63333, -56.61917, -34.83346, -56.16735, 4); //libertad a montevideo

s.registrarTramo(-34.63333, -56.61917, -32.91667, -54.96667, 6); //libertad a santa clara

s.registrarTramo(-33.53889, -56.88861, -34.83346, -56.16735, 19); //tambo flores a montevideo

s.registrarTramo(-33.25583, -58.01917, -34.83346, -56.16735, 30); //tambo soriano a montevideo

assertEquals(Sistema.TipoError.OK, s.listadoDeTambosEnCiudad(-34.83346, -56.16735).getResultado());

//devuelve los tambos a 20 km o menos por los tramos prestablecidos, Tambos central, flores, y santa clara.

assertEquals("-33.41306;-56.50056|-33.53889;-56.88861|-32.91667;-54.96667|", s.listadoDeTambosEnCiudad(-34.83346, -56.16735).getValorString());

}

//TEST LISTADO DE PRODUCTORES

@Test

public void listadoProductores() {

s.inicializarSistema(8);

//Cargo primero los productores para poder listarlos

assertEquals(Sistema.TipoError.OK, s.registrarProductor("4.624.303-4", "Sebastian", "Gral Rivera 722", "srodriguez@gmail.com", "099123456").getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarProductor("4.624.303-2", "Juan", "18 de Julio 221", "juan@gmail.com", "099123456").getResultado());

assertEquals(Sistema.TipoError.OK, s.registrarProductor("4.624.303-3", "Luis", "25 de Mayo", "luis@gmail.com", "098123456").getResultado());

String lista = "4.624.303-2;Juan;099123456|4.624.303-3;Luis;098123456|4.624.303-4;Sebastian;099123456|";

assertEquals(Sistema.TipoError.OK, s.listadoProductores().getResultado());

assertEquals(lista, s.listadoProductores().getValorString());

}