# ALGORITMOS Y ESTRUCTURA DE DATOS 2

## Documentación - Obligatorio

 

María Eugenia Cremona

196172

Claudio Pellegrini

194777

Grupo

Francisco Bouza

23 de noviembre de 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)

# 

**Obligatorio :: Gestión de Data-Centers**

Una empresa de tecnología que tiene varios data-centers (DC) en américa está interesada en contar con un sistema que registre sus DC y los de la competencia, así como las conexiones entre éstos. Un DC tiene una empresa propietaria, una ubicación, y una capacidad de análisis de información. Si un DC necesita procesar información (o simplemente almacenarla) y no tiene la capacidad necesaria, puede enviarla a otro DC para que lo procese. El DC al que se envíe la información para procesar será elegido dependiendo de: - El costo de transmisión de la información. - Que el DC de destino tenga esa capacidad - El costo de procesamiento del DC donde irá. Un DC de la misma empresa no tendrá costos, pero sí tendrá costo adicional si el procesamiento lo hace un DC de la competencia. El sistema deberá contar con un mapa en donde se podrán ver las principales ciudades del continente y los diferentes DC. Además, será el encargado de decidir a qué DC serán enviados los pedidos de procesamiento siguiendo las reglas especificadas. Finalmente, cada cierto tiempo será necesario hacer mantenimiento sobre los DC y las conexiones entre estos, por lo que el sistema deberá dar un reporte de cuáles son los DC y las conexiones mínimas que deben permanecer funcionando para poder mantener la conexión entre todos los puntos con un costo total mínimo.

# Interfaz Sistema: Pre y post condiciones

//Precondiciones: Tipo de Datos validados

//Postcondiciones: Crea Sistema con cantPuntos

Retorno inicializarSistema (int cantPuntos);

//Precondiciones:

//Postcondiciones: Destruye el sistema y todos sus elementos liberando la memoria utilizada.

Retorno destruirSistema();

//Precondiciones: Tipo de Datos validados

//Postcondiciones: Crea una nueva Empresa, lo inserta en árbol ABBEmpresa

Retorno registrarEmpresa(String nombre, String direccion, String

pais, String email\_contacto, String color);

//Precondiciones: Tipo de Datos validados

//Postcondiciones: Crea una nueva Ciudad, la inserta en grafo mapa y en arraylist ciudades

Retorno registrarCiudad(String nombre, Double coordX, Double

coordY);

//Precondiciones: Tipo de Datos validados

//Postcondiciones: Crea un nuevo DC, lo inserta en grafo mapa y en arraylist datacenters

Retorno registrarDC(String nombre, Double coordX, Double coordY,

String empresa, int capacidadCPUenHoras, int costoCPUporHora);

//Precondiciones: Tipo de Datos validados

//Postcondiciones: Registra un nuevo tramo en grafo mapa

Retorno registrarTramo(Double coordXi, Double coordYi, Double

coordXf, Double coordYf, int peso);

//Precondiciones: Tipo de Datos validados

//Postcondiciones: Elimina un tramo en grafo mapa

Retorno eliminarTramo(Double coordXi, Double coordYi, Double

coordXf, Double coordYf);

//Precondiciones: Tipo de Datos validados

//Postcondiciones: Elimina un punto y sus tramos asociados en grafo mapa

Retorno eliminarPunto(Double coordX, Double coordY);

//Precondiciones:

//Postcondiciones: Abre ventana de navegador y muestra en un mapa de Google Maps todos los puntos registrados en el sistema

Retorno mapaEstado();

//Precondiciones: Tipo de Datos validados

//Postcondiciones: Calcula y retorna cuál es el DC que puede para procesar información

Retorno procesarInformacion (Double coordX, Double coordY, int

esfuerzoCPUrequeridoEnHoras);

//Precondiciones:

//Postcondiciones: Retorna listado de los tramos mínimos de conexión de la red (grafo)

Retorno listadoRedMinima();

//Precondiciones:

//Postcondiciones: Retorna listado de empresas ordenadas por nombre en manera ascendente

Retorno listadoEmpresas();

# Solución escogida

## Diagrama de la estructura de datos

## 

## Justificación

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

|  |  |
| --- | --- |
| **ABBEmpresas** | |
| Estructura | Árbol Binario de Búsqueda |
| Justificación | Por requerimiento de la letra, la inserción de Empresas debía implementarse en orden (log n) promedio. Además nos permitía solucionar el listado de empresas ordenado con una simple recorrida IN-ORDEN |

|  |  |
| --- | --- |
| **mapa** | |
| Estructura | Grafo |
| Justificación | Para contemplar la propuesta de la letra (donde se necesita un sistema de información geográfica), y considerando que es una de las aplicaciones probables del grafo, es que lo elegimos para representar estructuralmente el flujo de esta red (grafo ponderado). |

|  |  |
| --- | --- |
| **ciudades** | |
| Estructura | ArrayList |
| Justificación | Al no tener condicionantes por requerimientos del sistema, se eligió esta estructura. |

|  |  |
| --- | --- |
| **datacenters** | |
| Estructura | ArrayList |
| Justificación | Al no tener condicionantes por requerimientos del sistema, se eligió esta estructura. |

## Testing

Al juego de testeo provisto por el docente (TestDocente) incluído en el proyecto, se le agrega un nuevo JUnitTest llamado: **TestPropio**.

Este es el resumen de las pruebas realizadas y sus resultados:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SISTEMA \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Test

public void testInicializarSistemaCero();

Prueba: Intento de inicializar sistema con 0 puntos

Resultado: Error1

@Test

public void testInicializarSistema();

Prueba: Inicializar Sistema

Resultado: Ok

@Test

public void testDestruirSistema();

Prueba: Destruir sistema

Resultado: Ok

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* EMPRESA \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Test

public void testRegistroEmpresa();

Prueba: Registrar empresa

Resultado: Ok

@Test

public void testRegistroEmpresaMailMal();

Prueba: Intento de registrar Empresa con un mail inválido

Resultado: Error1

@Test

public void testRegistroEmpresaNombreDuplicado();

Prueba: Intento de registrar Empresa con nombre duplicado

Resultado: Error2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CIUDAD \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Test

public void testRegistroCiudad() ;

Prueba: Registrar Ciudad

Resultado: Ok

@Test

public void testRegistroCiudadCoordExiste() ;

Prueba: Intento de registrar Ciudad con coordenadas ya registradas en el Sistema

Resultado: Error2

@Test

public void testRegistroCiudadMaxPuntos();

Prueba: Intento de registrar Ciudad con el máximo de cantidad de puntos ya superados

Resultado: Error1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* DATA-CENTER \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Test

public void testRegistroDC();

Prueba: Registrar DC

Resultado: Ok

@Test

public void testRegistroDCMaxPuntos();

Prueba: Intento de registrar DC con el máximo de cantidad de puntos ya superados

Resultado: Error1

@Test

public void testRegistroDCCapCero();

Prueba: Intento de registrar DC con capacidad igual o menor a 0

Resultado: Error2

@Test

public void testRegistroDCCoordExiste() ;

Prueba: Intento de registrar DC con coordenadas que ya están registradas en Sistema

Resultado: Error3

@Test

public void testRegistroDCEmpresaNoExiste() ;

Prueba: Intento de registrar DC con empresa no registrada

Resultado: Error4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* TRAMOS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Test

public void testRegistroTramo();

Prueba: Registrar Tramo

Resultado: Ok

@Test

public void testRegistroTramoPesoCero();

Prueba: Intento de registrar tramo con peso menor o igual a 0

Resultado: Error1

@Test

public void testRegistroTramoCoordNoExiste() ;

Prueba: Intento de registrar tramo con coordenadas no registradas en el Sistema

Resultado: Error2

@Test

public void testRegistroTramoYaExiste();

Prueba: Intento registrar tramo con coordenadas que ya están registradas en el Sistema

Resultado: Error3

@Test

public void testEliminarTramo();

Prueba: Eliminar Tramo

Resultado: Ok

@Test

public void testEliminarTramoCoordNoExiste();

Prueba: Intento eliminar tramo con coordenadas que NO están registradas en el Sistema

Resultado: Error1

@Test

public void testEliminarTramoNoExiste();

Prueba: Intento eliminar tramo que NO está registrado en el Sistema

Resultado: Error2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PUNTO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Test

public void testEliminarPunto();

Prueba: Eliminar Tramo

Resultado: Ok

@Test

public void testEliminarPuntoNoExiste();

Prueba: Intento eliminar Punto que NO está registrado en el Sistema

Resultado: Error1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAPA \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Test

public void testMostrarMapa();

Prueba: Mostrar mapa en navegador

Resultado: Ok

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PROCESAR INFORMACIÓN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Test

public void testProcesarInfo();

Prueba: Procesar información en un Data-Center

Resultado: Ok

@Test

public void testProcesarInfoCoordNoExiste();

Prueba: Intento procesar información en un Data-Center con coordenadas que NO están registradas en el Sistema

Resultado: Error1

@Test

public void testProcesarInfoCapSuperadaTodos();

Prueba: Intento procesar información con esfuerzo mayor a la capacidad de todos los DC

Resultado: Error2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Red Mínima \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Test

public void testListadoRedMinima();

Prueba: Genera el listado de red mínima para mantener la conexión entre DC con el menor costo

Resultado: Ok

**Nota:**

Además de estas pruebas, en la clase main Alg2Obligatorio, se fueron realizando otras pruebas (similares a las realizadas en JUnitTest), que sirvieron de guía durante el desarrollo.