# ALGORITMOS Y ESTRUCTURA DE DATOS 2

## Documentación - Obligatorio





Bruno DIAZ

203056

Federico SPERONI

165357

Grupo: M4A

Nombre del docente: Sebastián GRATTAROLA

Fecha: 23/11/2016

Índice

[ALGORITMOS Y ESTRUCTURA DE DATOS 2 1](#_Toc467664421)

[Documentación - Obligatorio 1](#_Toc467664422)

[Generales: 3](#_Toc467664423)

[Específicas de cada método: 3](#_Toc467664424)

[Solución escogida 5](#_Toc467664425)

[Diagrama de la estructura de datos 5](#_Toc467664426)

[Justificación 5](#_Toc467664427)

[Testing 7](#_Toc467664428)

[Resultado de las pruebas realizadas con la clase Prueba utilizada desde el Main 7](#_Toc467664429)

[Mapa utilizado para las pruebas 7](#_Toc467664430)

[Link para llegar al mapa 8](#_Toc467664431)

[Resultado de las pruebas realizadas con el JUnit brindado en Aulas 8](#_Toc467664432)

Interfaz Sistema: Pre y post condiciones

## Generales:

//Precondiciones: Respetar los Tipos de Dato de los parámetros de entrada

## Específicas de cada método:

//Precondiciones: No posee

//Postcondiciones: Genera las estructuras necesarias para el funcionamiento del sistema, limitando la cantidad puntos en cantidad igual a *cantPuntos*.

Retorno **inicializarSistema**(int cantPuntos);

//Precondiciones: No posee

//Postcondiciones: Destruye las estructuras creadas y libera la memoria utilizada.

Retorno **destruirSistema**();

//Precondiciones: No posee.

//Postcondiciones: Registra la empresa de nombre *nombre* y le asocia el color *color* para su representación en el mapa.

Retorno **registrarEmpresa**(String nombre, String direccion, String pais,

String email\_contacto, String color);

//Precondiciones: No posee.

//Postcondiciones: Registra en el mapa la ciudad de nombre *nombre*.

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

//Precondiciones: No posee.

//Postcondiciones: Registra en el mapa el DataCenter de nombre *nombre*perteneciente a la empresa *empresa* con una capacidad igual a *capacidadCPUenHoras* y un costo por hora de procesamiento igual a *costoCPUPorHora*.

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

String empresa,int capacidadCPUenHoras, int costoCPUporHora);

//Precondiciones: No posee.

//Postcondiciones: Registra un tramo que va desde el punto (Ciudad o DataCenter) con coordenadas *coordXi, coordYi* hasta el punto con coordenadas *coordXf, coordYf*, así como el tramo inverso, ambos con peso igual a *peso.*

Retorno **registrarTramo**(Double coordXi, Double coordYi, Double coordXf,

Double coordYf, int peso);

//Precondiciones: No posee.

//Postcondiciones: Elimina un tramo que va desde el punto (Ciudad o DataCenter) con coordenadas *coordXi, coordYi* hasta el punto con coordenadas *coordXf, coordYf*, así como el tramo inverso.

Retorno **eliminarTramo**(Double coordXi, Double coordYi, Double coordXf,

Double coordYf);

//Precondiciones: No posee.

//Postcondiciones: Elimina el punto (Ciudad o DataCenter) con coordenadas *coordX, coordY.* Así como todos los tramos en los que esté involucrado.

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

//Precondiciones: Se debe tener conexión a internet.

//Postcondiciones: Abre una ventana del navegador seleccionado por defecto, donde se representan las ciudades en color Amarillo, los DataCenter en los colotes de su empresa, y los tramos en rojo. En caso de no tener puntos guardado, muestra el mapa de Uruguay.

Retorno **mapaEstado**();

//Precondiciones: El grafo debe ser Conexo (de cualquier punto del mapa seleccionado, se debe poder llegar a todos los demás)

//Postcondiciones: En caso de haber un DataCenter que pueda procesar la información solicitada, le resta *esfuerzoCPUrequeridoEnHoras* a su capacidad actual, y retorna el nombre del DataCenter seleccionado y el costo total para realizar la operación.

Retorno **procesarInformación**(Double coordX, Double coordY,

int esfuerzoCPUrequeridoEnHoras);

//Precondiciones: El grafo debe ser Conexo (de cualquier punto del mapa seleccionado, se debe poder llegar a todos los demás)

//Postcondiciones: Devuelve un String donde se detallan los tramos que pertenecen al Arbol de Cubrimiento de menor costo total.

Retorno **listadoRedMinima**();

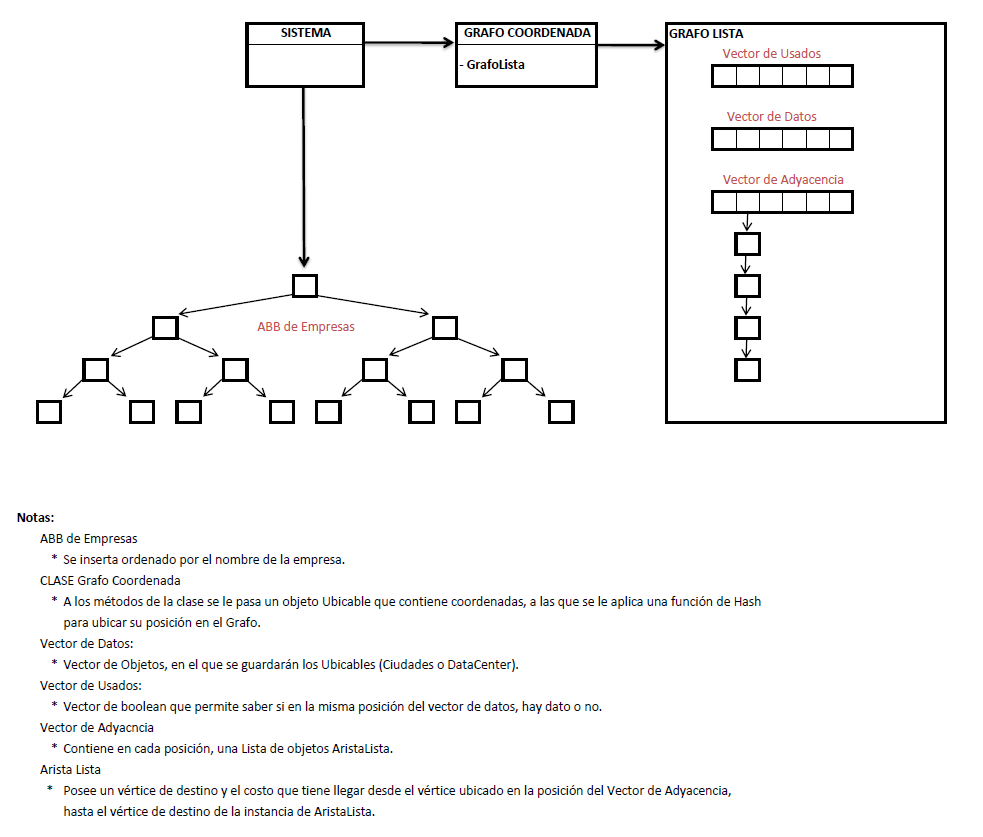
//Precondiciones: No posee.

//Postcondiciones: Devuelve un listado de todas las empresas registradas, con su nombre y si email.

Retorno **listadoEmpresas**();

# Solución escogida

## Diagrama de la estructura de datos



## Justificación

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

|  |  |
| --- | --- |
| **EMPRESAS** | |
| Estructura | ABB |
| Justificación | Se solicita que el ingreso de DataCenter sea en orden promedio Log n, por lo que para insertarlo, se debe buscar la empresa ya guardada, siendo ese órden el promedio de la estructura seleccionada. |

|  |  |
| --- | --- |
| **UBICABLES (DataCenter y Ciudades)** | |
| Estructura | Grafo |
| Justificación | Se debe tener una estructura que permita ingresar Objetos y conexiones entre los mismos. Para poder cumplir con los requisitos del sistema, tales como encontrar el camino mínimo entere 2 puntos, o todas las conexiones que dejen a todos los puntos conectados con el menor costo posible, debemos utilizar los algoritmos de Dijkstra y de Prim respectivamente, los cuales se aplican a un Grafo. |

## Testing

Nota: Se realizaron las pruebas de JUnit provistas en Aulas, y pruebas utilizando la clase Prueba provista por el docente Rafael COHEN para el obligatorio de Algoritmos 1.

## Resultado de las pruebas realizadas con la clase Prueba utilizada desde el Main

Nota: Las clases Prueba y Main están en el obligatorio para que se pueda ver su código.

+------------------------------+

RESULTADOS DE LA PRUEBA

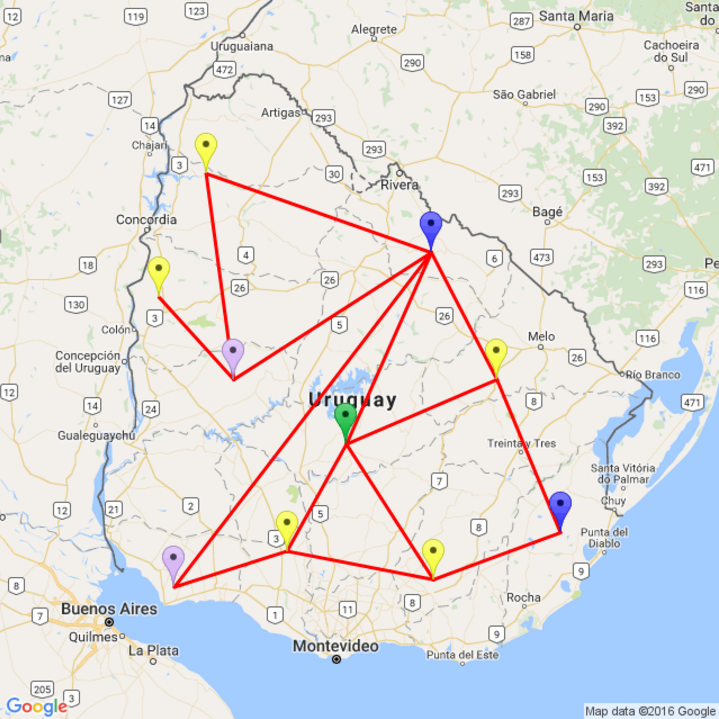
Pruebas Correctas: 72

Pruebas Incorrectas: 0

Pruebas NI: 0

+------------------------------+

### Mapa utilizado para las pruebas



### Link para llegar al mapa

http://maps.googleapis.com/maps/api/staticmap?size=1200x800&maptype=roadmap&sensor=false&markers=color:0x2E2EFE|-33.88,%20-53.98&markers=color:0xF7FE2E|-34.26,%20-55.22&markers=color:0xF7FE2E|-30.92,%20-57.44&markers=color:0xF7FE2E|-34.03,%20-56.65&markers=color:0xF7FE2E|-31.95,%20-57.9&markers=color:0x04B431|-33.16,%20-56.07&markers=color:0xD0A9F5|-32.63,%20-57.17&markers=color:0xD0A9F5|-34.32,%20-57.75&markers=color:0xF7FE2E|-32.63,%20-54.6&markers=color:0x2E2EFE|-31.58,%20-55.24&path=color:0xff0000ff|weight:3|-33.88,%20-53.98|-34.26,%20-55.22&path=color:0xff0000ff|weight:3|-33.88,%20-53.98|-32.63,%20-54.6&path=color:0xff0000ff|weight:3|-34.26,%20-55.22|-34.03,%20-56.65&path=color:0xff0000ff|weight:3|-34.26,%20-55.22|-33.16,%20-56.07&path=color:0xff0000ff|weight:3|-30.92,%20-57.44|-32.63,%20-57.17&path=color:0xff0000ff|weight:3|-30.92,%20-57.44|-31.58,%20-55.24&path=color:0xff0000ff|weight:3|-34.03,%20-56.65|-34.32,%20-57.75&path=color:0xff0000ff|weight:3|-34.03,%20-56.65|-33.16,%20-56.07&path=color:0xff0000ff|weight:3|-31.95,%20-57.9|-32.63,%20-57.17&path=color:0xff0000ff|weight:3|-33.16,%20-56.07|-31.58,%20-55.24&path=color:0xff0000ff|weight:3|-33.16,%20-56.07|-32.63,%20-54.6&path=color:0xff0000ff|weight:3|-32.63,%20-57.17|-31.58,%20-55.24&path=color:0xff0000ff|weight:3|-34.32,%20-57.75|-31.58,%20-55.24&path=color:0xff0000ff|weight:3|-32.63,%20-54.6|-31.58,%20-55.24

## Resultado de las pruebas realizadas con el JUnit brindado en Aulas

