Práctica 29

Datos generales:

Nombre de la Práctica	Grafos
Nombre de la carrera	Ingeniería de Software
Nombre de la materia	Laboratorio de Estructuras de Datos
Número y nombre de Unidad(es) temática(s)	IV. Estructuras no lineales.
Docente que imparte la materia	Aldonso Becerra Sánchez
Fecha de entrega para los alumnos	10-noviembre-2021
Fecha de entrega para los alumnos	11-noviembre-2021
Fecha de elaboración	10-noviembre-2021

Objetivo de la Práctica	Practicar con las operaciones sobre grafos.
Tiempo aproximado de realización	3 horas
Introducción	Los grafos son útiles en el manejo de mapas y planos donde se defina un amplia gama de posibilidades de conexión entre diferentes entidades con el fin de estar comunicadas o por el hecho de ser dependientes unas de otras.

Referencias que debe consultar el alumno (si se requieren):

Referencia 1:

1.Cairo, Osvaldo; Guardati, Silvia. Estructura de Datos, Tercera Edición. McGraw-Hill, México, Tercera Edición, 2006.

Referencia 2:

2.Mark Allen Weiss. Estructura de datos en Java. Ed. Addison Wesley.

Referencia 3:

3. Joyanes Aguilar, Luis. Fundamentos de Programación. Algoritmos y Estructuras de Datos. Tercera Edición, 2003. McGraw – Hill.

Actividades que debe realizar el alumno:

Actividad inicial:

Generar el reporte en formato IDC.

Actividad 1:

Primero genere la **Introducción**.

Actividad 2:

Complementar la clase GrafoMatriz con los métodos de:

- eliminarVertice(Object vertice). Elimina un vértice (dado su nombre) del grafo y todos los proceso que ello conlleva, tenga o no aristas. La idea es que no queden huecos en los lotes de almacenamiento.
- esAdyacente(Object origen, Object destino). Indica si hay adyacencia entre dos vértices proporcionados como argumento (en formato de nombres).
- eliminarArista(Object origen, Object destino). Este método elimina una arista ente un par de nodos proporcionados como "nombres".
- buscarVertice(Object vertice). Busque un nodo en un grafo. Si existe debe regresar la información en cuestión, en caso contrario null.
- esPseudografo(). Regresa verdadero si el grafo que tiene lazos o bucles.
- esMultigrafo(). Regresa verdadero si al menos dos de sus vértices están conectados entre sí por medio de dos aristas (aristas múltiples o paralelas).
- gradoVertice(Object vertice). Regresa el número de aristas que contiene el vértice. Si es 0, es nodo aislado.
- hayRuta(Object origen, Object destino). Es un proceso que usa ciclos para determinar si existe un camino válido entre dos vértices proporcionados como argumentos. Este proceso tiene que ir brincando de celda por celda de la matriz de acuerdo a las adyacencias existente, al final indicará si se pudo alcanzar el destino o no, que es donde terminará el proceso.
- esConexo(). Un grafo es conexo si desde cualquier vértice existe un camino hasta cualquier otro vértice del grafo. Puede apoyarse en el procedimiento explicado en clase que usa los recorridos (profundidad o anchura).

- hayCaminoCerrado(Object origen). Este método determina si tiene ciclos. Indica si el primero y último vértice son iguales en un camino o ruta definido (no cuenta brincar del origen al propio origen directamente como en un bucle, debe ser una ruta alterna). Va relacionado con el método de hayRuta(). Modifique el método de ordenación topológica de tal manera que con este método ya se pueda verificar si un grafo tiene ciclos o no. En caso que se pueda generar la ordenación topológica, hacerlo; en caso contrario, marcar el error.
- esCaminoSimple(Object origen, Object destino). Indica si el camino es simple, es decir, si todos sus nodos son distintos, excepto el primero y último, que pueden ser iguales (no cuenta que sea un bucle o ciclo).
- esDirigido(). Este método indica si el grafo es dirigido o no dirigido.
- esArbol(). Se dice que un grafo no dirigido es un árbol si es conexo, acíclico y donde cada vértice solo debe tener un padre. En el caso de un grafo dirigido, no debe tener ciclos y cada vértice solo debe tener un padre.
- listarAristas(). Este método muestra en el formato (origen, destino, peso), un listado de todas las aristas que tiene un grafo. Por ejemplo: [(A,B,12), (C,D,3), …, (G,H,9)].
- listarAristas(Object vertice). Lista las aristas del vértice proporcionado.
- listarVertices(). Muestra los vértices del grafo.

Complete la etapa de **desarrollo** del reporte con base a esta actividad.

Actividad 3:

Pruebe el funcionamiento del programa de las actividades con todo y sus capturas de pantalla.

Actividad 4:

Realice la sección de **Código agregado** (diagrama de clases UML).

Actividad 5:

Realice la sección de **Pre-evaluación** (use los lineamientos establecidos).

Actividad 6:

Finalmente haga las **Conclusiones**.

Actividad 7:

Enviar en http://ingsoftware.reduaz.mx/moodle

Archivo anexo que se requiere para esta tarea (opcional):

Dudas o comentarios: **a7donso@gmail.com**