PARTE 3: Ejercicio de programación JAVA

Duración 60 minutos

Este ejercicio comprende **2** actividades (no se especifica un orden de las mismas):

a) Desarrollo de la funcionalidad especificada más abajo
b) Desarrollo de los casos de prueba ("test case", al menos uno) para verificar la corrección de la funcionalidad implementada.

ESCENARIO

Una compañía de telecomunicaciones tiene una gran **red de datos** instalada en una cierta región. La red cuenta con "**servidores**" y "**enlaces**".

Para enviar un paquete de datos desde un servidor a otro de la red es posible utilizar diferentes rutas. Un problema típico a resolver es encontrar una ruta entre dos servidores cualesquiera, que tenga la menor cantidad de saltos posible, es decir la mínima cantidad de servidores intermedios.

Por ello, se le ha encargado al Departamento de Sistemas que desarrolle una aplicación para que la consulta sobre cuál es la ruta entre dos servidores con la menor cantidad de saltos posible, se resuelva de la forma más eficiente posible.

El Arquitecto de Sistemas de la empresa ha determinado que como modelo para representar esta red de datos puede usarse un **GRAFO NO DIRIGIDO** (donde los servidores se representan mediante los vértices, y los enlaces mediante las aristas), y te ha encargado entonces que desarrolles y pruebes un método que permita obtener la ruta con menos saltos que enlace un cierto servidor con otro indicado.

Nota: Cada servidor (vértice) cuenta con un atributo de tipo servidor (vértice), llamado "**predecesor**", que **puede** ser usado para cargar allí el predecesor (anterior) a él en la ruta que se busca.

Esta funcionalidad **DEBE SER IMPLEMENTADA CON EL MENOR ORDEN DEL TIEMPO DE EJECUCIÓN POSIBLE.**

Funcionalidad a desarrollar:

Descargar de la webasignatura el archivo "**Parcial2-2021.zip**" que contiene el Proyecto a ser completado.

Se desea:

1. Generar una estructura apropiada para representar el problema y cargarla a partir de los archivos de entrada indicados. Ver detalles de los archivos de entrada más abajo.

2. Desarrollar una funcionalidad que permita, dados dos servidores (que se han de indicar en la prueba), devolver la ruta con menor cantidad de enlaces (la ruta consiste en una lista de servidores, incluyendo los dos indicados como parámetros).

3. Emitir un archivo de salida "**rutas.txt**" con el resultado de la ejecución de las operación indicadas (**OBSERVAR EN EL PROGRAMA PRINCIPAL SE INDICAN 3 EJECUCIONES CONSECUTIVASDEL ALGORITMO CON DIFERENTES P, SIN DETENER EL PROGRAMA)**, con las etiquetas de los servidores de cada ruta solicitada por cada línea, dejando 2 líneas en blanco entre rutas.

4. Implementar las verificaciones básicas que aseguren que las funcionalidades anteriores se pueden ejecutar correctamente para el grafo representado.

De TGrafoRedDatos (que solamente extiende de TGrafoNoDirigido a efectos de esta operación)

LinkedList <TVertice> rutaMenosSaltos (Comparable origen, Comparable destino)

Test cases.

Implementa los **Casos de Prueba** necesarios para verificar el correcto funcionamiento de los métodos desarrollados.

ENTREGA

Subir a la webasignatura, en la tarea "**PARCIAL2-PARTE3**" el proyecto completo realizado, más el archivo de salida "**rutas.txt**".

ARCHIVOS DE ENTRADA

- "**servidores.txt**": lista de servidores de la Red de Datos
- "**enlaces.txt**": cada línea representa un enlace: servidor1, servidor2, 1 (cada enlace tiene costo 1).

NOTA IMPORTANTE: LOS ARCHIVOS **PUEDEN** CONTENER ERRORES. SE **DEBEN** IMPLEMENTAR LAS PRECAUCIONES BASICAS PARA EVITAR QUE EL SISTEMA COLAPSE ANTE ERRORES O INCONGRUENCIA DE DATOS.

RUBRICA DE CALIFICACIÓN: se utilizarán los siguientes criterios en la evaluación del trabajo remitido:

1. **EJECUCIÓN: 40%**
   - Correcta lectura de los archivos de datos y creación de las estructuras definidas.
   - Consideraciones de seguridad con respecto a los datos – chequeos de consistencia y corrección realizados
   - Ejecución completa y en tiempo razonable
   - Emisión de los resultados correctos
2. **DESARROLLO 40%**
   - Selección e implementación del método con un algoritmo que tenga el **MENOR ORDEN DEL TIEMPO DE EJECUCION POSIBLE**
   - Estructura de datos utilizada (pertinencia, eficiencia)
   - Cumplimiento de las interfaces publicadas
   - Corrección del método solicitado
   - Implementación de chequeos necesarios para cumplimiento de la funcionalidad requerida
   - Programa principal, generación correcta del archivo de salida.
3. **CALIDAD DEL CÓDIGO (5 %)**
   - Nombres de variables y métodos
   - Aplicación correcta y rigurosa del paradigma de programación orientada a objetos
   - Invocación racional y eficiente de métodos y funciones
   - Encapsulación, modularidad
   - Utilización apropiada de clases de colecciones y genéricos necesarios
4. **PRUEBAS DE UNIDAD 15%.**
   - Calidad de los tests desarrollados, todas las condiciones normales y de borde, se testean todos los métodos, uso del enfoque inductivo en los tests.