

ALGORITMOS Y ESTRUCTURAS DE DATOS
SEGUNDO PARCIAL - SEGUNDO SEMESTRE 2024
PARTE 3: Ejercicios de Programación (Duración: 60 minutos)

Ejercicio 1 (Grafos)

Escenario

En una ciudad ficticia, se está diseñando un sistema de transporte público basado en estaciones conectadas por rutas directas. Cada estación está identificada por un nombre único, y cada ruta entre estaciones tiene un tiempo estimado de recorrido en minutos. Además, algunas rutas tienen restricciones de uso (por ejemplo, pueden estar cerradas temporalmente).

Modelado del escenario

```
class SistemaTransporte {
    private Map<String, List<Ruta>> grafo;
    // Clase interna para representar una ruta
    private static class Ruta {
        String destino;
        int tiempo;
    }
    // Clase interna para representar una estación en la consulta de tiempo mínimo
    private static class Estacion {
        String nombre;
    }
    // Clase interna para representar una ruta conectada en el árbol generador mínimo
    public static class RutaConectada {
        String origen;
        String destino;
        int tiempo;
    }
}
```

Problemas por resolver: debes elegir solo uno

Problema 1: Tiempo Mínimo de Viaje.

Dadas dos estaciones de origen y destino, encuentra el tiempo mínimo necesario para viajar entre ellas, teniendo en cuenta los tiempos de las rutas disponibles.

- Si no existe un camino que conecte las dos estaciones, debe retornarse una señal clara indicando esta situación.

Firma de la función

```
/**
 * Encuentra el tiempo mínimo de viaje entre dos estaciones.
 * @param origen Nombre de la estación de origen.
 * @param destino Nombre de la estación de destino.
 * @return Tiempo mínimo necesario para viajar entre las estaciones, o -1 si no hay camino.
```

```
*/
public int consultaTiempoMinimo(String origen, String destino) { ... }
```

Problema 2: Diseño de una Red de Mantenimiento.

Encuentra un subconjunto mínimo de rutas (es decir, un subgrafo) que conecte todas las estaciones con el menor tiempo total posible. Este diseño ayudará a determinar qué rutas priorizar para mantenimiento preventivo.

Firma de la función

```
/**
 * Encuentra un subconjunto mínimo de rutas para conectar todas las estaciones.
 * @return Lista de rutas seleccionadas en el formato (origen, destino, tiempo).
 */
public List<RutaConectada> redDeMantenimiento() { ... }
```

Ejemplos de uso (ambos problemas)

```
public class Main {
    public static void main(String[] args) {
        SistemaTransporte sistema = new SistemaTransporte();
        sistema.agregarRuta("A", "B", 5);
        sistema.agregarRuta("A", "C", 10);
        sistema.agregarRuta("B", "C", 2);
        sistema.agregarRuta("B", "D", 4);
        sistema.agregarRuta("C", "D", 1);

        // Problema 1: Tiempo mínimo de viaje
        int tiempoMinimo = sistema.consultaTiempoMinimo("A", "D");
        System.out.println("Tiempo mínimo entre A y D: " + tiempoMinimo);
    }

    // Resultado esperado:
    // Tiempo mínimo entre A y D: 8

    // Problema 2: Red de mantenimiento
    List<SistemaTransporte.RutaConectada> red = sistema.redDeMantenimiento();
    System.out.println("Red de mantenimiento:");
    for (SistemaTransporte.RutaConectada ruta : red) {
        System.out.println(ruta.origen + " - " + ruta.destino + ": " + ruta.tiempo + " minutos");
    }

    // Resultado esperado:
    // Red de mantenimiento:
    // A - B: 5 minutos
    // B - C: 2 minutos
    // C - D: 1 minutos
}
}
```

CONSIDERACIONES IMPORTANTES DEL PARCIAL

- Se tendrá en cuenta la eficiencia de los algoritmos desarrollados, calificando mejor aquellos que tengan mejor tiempo de ejecución o mejor performance general.