Práctica 2

Temas 4, 5 y 6

Objetivo

Resolver de forma **individual** y **original** los ejercicios planteados, para demostrar que se alcanza los siguientes resultados de aprendizaje de la asignatura:

- **RAE1.** Ser capaz de calcular la complejidad teórica de un algoritmo y su orden de magnitud, para poder argumentar la elección de una solución frente a otra.
- **RAE2.** Entender estrategias algorítmicas clásicas (divide y vencerás, avance rápido y vuelta atrás), y ser capaz de utilizarlas a la hora de implementar soluciones a problemas concretos.
- **RAE4.** Comprender las principales estructuras de datos jerárquicas (árboles), su utilidad, su funcionamiento y su implementación. Ser capaz de usarlas para la resolución de problemas concretos y como mecanismo para plantear soluciones más óptimas.
- **RAE5.** Comprender las principales estructuras de datos relacionales (grafos), su utilidad, su funcionamiento y su implementación. Ser capaz de emplearlas para la resolución de problemas concretos y como mecanismo para plantear soluciones más óptimas.

Ejercicio 1. Árboles. Traza.

Genera el árbol AVL en el que, partiendo de un árbol vacío, se insertan (en el orden indicado) los siguientes elementos: {1,2,3,4,5,6,7,15,14,13,12,11,10,9,8}.
Indica cuál es el árbol AVL final y cuáles han sido las rotaciones resultantes que has necesitado aplicar.

Ejercicio 2. Árboles. Código.

Diseñar (pseudocódigo) e implementar en Java una función que reciba como entrada un árbol binario de letras, y que escriba por pantalla los caracteres almacenados en el último nivel. Utilizar para ello las operaciones del TAD ArbolBin visto en la asignatura y la implementación proporcionada. Pueden codificarse las funciones auxiliares que se deseen. Calcular de manera razonada su eficiencia, justificando las complejidades de las operaciones y funciones utilizadas

Ejercicio 3. Grafos.

3.1 Codifica el algoritmo de Kruskal.

Utilizando la implementación de Lista, Par y Grafo proporcionados, crea una clase AlgorimoKruskalApellido. Dicha clase debe contener:

- un método algoritmoKruskalAR que respete la cabecera siguiente:

```
public Lista <Par< Clave>> AlgoritmoKruskalAR (Grafo <Clave,
InfoVertice, Coste>,
```

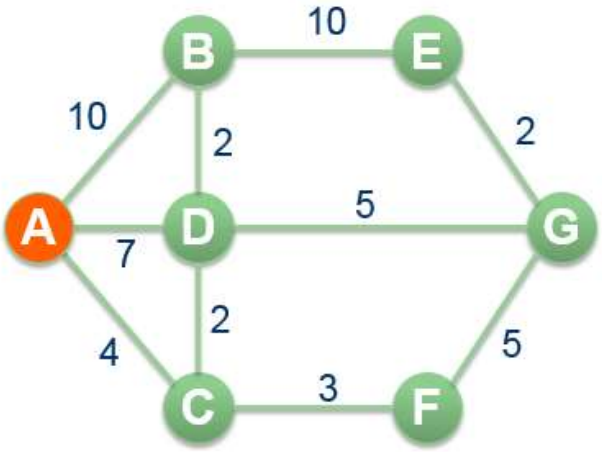
- un método main en el que se construya un grafo, se aplique método anterior para calcular el árbol de recubrimiento mínimo de dicho grafo, y una vez obtenido, mostrar por pantalla las aristas que forman dicho árbol, y el coste total.

Para ello pueden añadirse los métodos auxiliares (privados) que se consideren oportunos.

NOTA: Para mayor simplicidad, consideraremos como precondición que el **grafo no dirigido** de entrada debe ser **conexo**. Ten en cuenta que el grafo proporcionado es la implementación de un grafo dirigido.

3.2. Calcular razonadamente la complejidad de la implementación del (algoritmoKruskalAR), detallando las reducciones o sumatorios necesarios para su desarrollo.

3.3. Codifica en tu programa el siguiente grafo y aplícale el algoritmo de Kruskal para construir el árbol de recubrimiento mínimo y calcular el coste total.



Normas de entrega

La entrega consistirá en 1 único fichero comprimido que contendrá a su vez los siguientes archivos:

- Un documento PDF con la descripción de todo el trabajo y resolución de los ejercicios.
- AlgoritmoEscribeCaracteres.java
- AlgoritmoKruskalApellido.java

Se comprimirán en **formato ZIP** (no RAR, ni 7Z u otros). No respetar los formatos descritos en este apartado supondrán la no aceptación de la entrega.

Los diferentes ejercicios de entregarán de la siguiente manera:

- **Ejercicio 1:**
  - Se incluirá en el PDF.
  - Deben razonarse, desarrollarse e ilustrarse (dibujarse) todos los pasos de manera clara, paso a paso. Puede utilizarse una herramienta para generar los gráficos, o bien optarse por fotografiar/escanear los pasos realizados a mano alzada. En este último caso, las imágenes deben ser nítidas y perfectamente legibles, como por el ejemplo el dibujo utilizado en el enunciado de este ejercicio.
  - Debe indicarse al final el listado de rotaciones que se han tenido que realizar

- **Ejercicio 2:**
  - Se incluirán en el PDF los apartados:
    - Idea de resolución
    - Pseudocódigo
    - Código de la función (“copiar y pegar” del .java)
    - Cálculo de la complejidad de la solución propuesta.
  - Se entregará en el propio fichero AlgoritmoEscribeCaracteresApellido.java (que incluirá el primer apellido del estudiante). Es condición necesaria que el código compile y esté comentado.

- **Ejercicio 3:**
  - Se incluirán en el PDF los apartados:
    - Código de la función solicitada (“copiar y pegar” del .java)
    - Cálculo razonado de la complejidad de la solución propuesta, detallando las reducciones o sumatorios necesarios para su desarrollo.
    - Coste total del árbol de recubrimiento mínimo generado, y listado de aristas que lo componen.
  - Se entregará en el propio fichero AlgoritmoKruskalApellido.java (que incluirá el primer apellido del estudiante). Es condición necesaria que el código compile y esté comentado. El fichero Java debe mantener la estructura del proporcionado inicialmente.

Antes de subir la entrega, seguir los siguientes cuatro pasos:

1. Crear una carpeta con el siguiente formato de nombre:

Apellido1_Apellido2_Nombre.TPA.**P2**

2. En dicha carpeta, incluir los tres ficheros indicados:
    a. AlgoritmoImprimeCaracteresApellido.java
    b. AlgoritmoKruskalApellido.java
    c. El **PDF** con todos los detalles de la actividad. Este documento seguirá el siguiente formato a la hora de ser nombrado:

Apellido1_Apellido2_Nombre.TPA.P2.**PDF**

3. Comprimir en formato **ZIP** (no RAR, no 7Z, etc.) la carpeta creada en el punto 1, lo que generará un fichero que seguirá el siguiente formato:

Apellido1_Apellido2_Nombre.TPA.P1.**ZIP**

4. Entregar el fichero ZIP **antes de las 23:55h** de la fecha marcada como límite: **13 de junio de 2021.**