ALGORITMOS Y ESTRUCTURAS DE DATOS

PRIMER PARCIAL (SEGUNDO SEMESTRE 2024)

PARTE 2: Ejercicios de seudocódigo

Duración: 60 minutos

Ejercicio 1

En genealogía, un árbol genealógico es una representación de las relaciones familiares a lo largo de generaciones. Tradicionalmente, los **hijos** se encuentran en los niveles inferiores y los **padres** en los superiores. **Sin embargo, en esta variación del problema, invertimos el árbol: los padres están en los niveles inferiores, y sus descendientes directos están en los superiores**. De este modo, cada nodo tiene a lo sumo dos hijos, que representan al padre y la madre, y las ramas suben a través de las generaciones.

Este tipo de estructura invertida es útil para analizar la ascendencia de una persona. En lugar de ver cómo se expande la familia hacia abajo, nos enfocamos en cómo se consolida hacia arriba, identificando el linaje que conecta a cada persona con sus padres, abuelos, bisabuelos, y así sucesivamente.

Problema por resolver: Cálculo de los grados de parentesco

El problema consiste en calcular los **grados de parentesco** entre dos personas en un árbol genealógico invertido, diferenciando si el parentesco es por **consanguinidad** o **político**. En genealogía, el grado de parentesco mide cuán cercana es la relación entre dos individuos en función de su ascendencia, y puede estar vinculado ya sea por lazos de sangre (consanguinidad) o por matrimonio (parentesco político).

El árbol genealógico invertido sigue la misma estructura que hemos planteado antes, donde cada persona tiene hasta dos padres (representados como nodos hijos). El objetivo es rastrear estas conexiones para determinar la cercanía de dos personas y el tipo de parentesco.

Definiciones clave

- **Parentesco por consanguinidad**: Relación directa de sangre entre dos personas (padres, abuelos, bisabuelos, etc.).
- **Parentesco por afinidad o político**: Relación establecida por matrimonio (ej., suegros, cuñados, etc.).

Firma de la función:

calcularParentesco(raiz: IElementoAB<Persona>, persona1: IElementoAB<Persona>, persona2: IElementoAB<Persona>) -> (int, str)

Parámetros:

- **raíz**: El nodo raíz del árbol genealógico invertido, que representa a la persona más reciente en el linaje.
- **persona1 y persona2: Los nodos con la información de las dos personas entre los que se calculará el parentesco.**

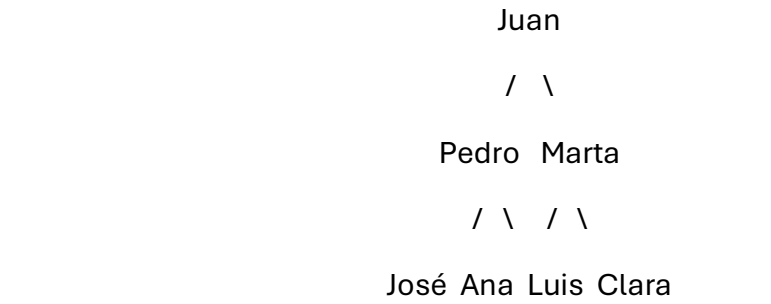
Retorno:

- La función devuelve una tupla con dos valores:
  - **int**: El número de grados de parentesco entre las dos personas.
  - **str**: El tipo de parentesco, que puede ser "consanguinidad" o "político".

Ejemplos del problema

1. **Parentesco por consanguinidad (relación directa de sangre)**:

Considera el siguiente árbol genealógico invertido:



Si se quiere calcular el parentesco entre **Juan** y **Pedro**:

- **Juan** es hijo de **Pedro**, por lo que el parentesco es de **1 grado de consanguinidad**.

Si se desea calcular el parentesco entre **José** y **Ana**:

- Ambos son padres de **Pedro**, así que hay **2 grados de parentesco político** entre ellos.

2. **Parentesco político (afinidad)**:

Siguiendo el mismo árbol:

- Si se desea calcular el parentesco entre **Pedro** y **Clara** (esposa de Luis):
  - La relación entre **Pedro** y **Clara** es **política** (yerno y suegra), ya que están relacionados por matrimonio a través de sus cónyuges. En este caso, se encontrarían **3 grados de parentesco político**.

Ejercicio 2

Imagina una empresa de desarrollo de software que maneja múltiples proyectos simultáneamente. Cada proyecto tiene un conjunto de tareas que deben completarse para cumplir con los plazos de entrega. Para asegurarse de que las tareas más críticas se aborden primero, se implementa un sistema de gestión de tareas basado en una **lista de prioridades**.

Descripción del Sistema:

1. **Tareas con Prioridades**: Cada tarea en el sistema tiene un nombre, una descripción, una fecha de vencimiento y una prioridad asignada. Las prioridades se representan con números enteros, donde un número más bajo indica mayor urgencia (por ejemplo, 1 = alta prioridad, 2 = media prioridad, 3 = baja prioridad).
2. **Lista de Prioridades**: Se utiliza una lista de prioridades (o heap) para gestionar y ordenar las tareas en función de su prioridad. Esto permite que el sistema siempre pueda acceder y ejecutar la tarea más urgente de manera eficiente.

Problema por resolver: implementar un TDA de ListaDePrioridades

Se conoce la existencia de la siguiente clase:

Clase Tarea

- Atributos:
    - nombre: str
    - descripcion: str
    - prioridad: int

Se debe implementar el TDA solicitado, con base en la implementación de alguna lista vista en clase, eligiendo la mejor correspondiente a este problema, teniendo que indicar sobrescribir los métodos que se deben sobrescribir para cumplir con el objetivo de la clase solicitada, teniendo que indicar su lenguaje natural, precondiciones y postcondiciones únicamente.

CONSIDERACIONES IMPORTANTES DEL PARCIAL

- Deben desarrollarse en detalle todos los métodos que se invoquen, salvo aquellos que correspondan a los métodos estándares de las estructuras vistas en clase o donde se indique que esto no es necesario.
- Se tendrá en cuenta la eficiencia de los algoritmos desarrollados, calificando mejor aquellos que tengan mejor tiempo de ejecución o mejor performance general.
- Se deben cumplir todos los pasos estándar de desarrollo de algoritmos, Lenguaje Natural, Pre y Post Condiciones, Pseudocódigo y Análisis del Orden de Tiempo de Ejecución.