PARTE 2: Ejercicios de seudocódigo

Duración: 60 minutos

CONSIDERACIONES IMPORTANTES

- Se deben respetar las normas publicadas sobre la escritura de seudocódigo (lenguaje natural, pre y post condiciones, seudocódigo detallado sobre esta implementación del TDA.
- Desarrollar en lenguaje natural los casos de prueba que se estimen convenientes.
- Se deben desarrollar completamente todas las operaciones invocadas.
- Analizar el orden del tiempo de ejecución

EJERCICIO 1

CONTEXTO

Los árboles binarios de búsqueda son muy utilizados para almacenamiento de información y búsquedas sobre la misma.

Supongamos que, en una estructura de este tipo, se han almacenado los datos de todos los alumnos de una Facultad. Cada alumno está representado mediante un identificador único (su CI, numérico, sin código de verificación), su nombre, apellido y carrera en la que está matriculado (estos tres son alfanuméricos).

Una de las tareas típicas que se suele realizar es consultar por los alumnos de una cierta carrera. Para evitar realizar una búsqueda extensa sobre todos los alumnos de la facultad, se decide generar otras estructuras que permitan hacerlo en forma más eficiente.

Se desea entonces, dado un árbol binario de búsqueda (TDA TABB) que contenga todos los alumnos de la facultad (con los datos indicados más arriba, habiendo sido insertados de acuerdo a su número de CI), y dado el nombre de una cierta carrera, crear un nuevo árbol que indexe la información de los alumnos correspondientes, por Apellido (se desea poder imprimir la lista de alumnos de la carrera ordenada por Apellido en la forma más eficiente).

TABBU -Tipo ArbolUniversidad- **// TABBU es un Arbol Binario de Búsqueda específico que contiene nodos //de tipo NodoAlumnoAB**

De Tipo NodoAlumnoAB Raíz

Tipo NodoAlumnoAB

De Tipo comparable Etiqueta
De Tipo Alumno Dato
De Tipo NodoAlumnoAB HijoIzq, HijoDer

Tipo Alumno

De Tipo numérico CI
De Tipo alfanumérico Apellido
De Tipo alfanumérico Carrera

De Tipo TABBU armarIndiceCarrrera (de tipo alfanumérico unaCarrera) devuelve TABBU
De Tipo NodoAlumnoAB indizar (de tipo TABBU indicePorCarrera; de tipo alfanumérico unaCarrera)

NOTAS IMPORTANTE:

- Se deben desarrollar los métodos de Árbol y de Nodo necesarios.
- Se debe evitar duplicar información

Ejemplo de invocación:

TABBU universidadCatolica // contiene todos los alumnos de la universidad, por CI
TABBU ingenieriaInformatica = universidadCatolica.armarIndiceCarrera(“Ingeniería Informática”)

EJERCICIO 2

A menudo se desea, para un cierto conjunto de elementos, procesar solamente un subconjunto de los mismos, cuyas claves se encuentren en un cierto rango determinado.

Supongamos que, en un Árbol Binario de Búsqueda, tenemos almacenado un conjunto de elementos que representan productos. Estos productos tienen un código o clave que es la que se utiliza como clave de inserción / búsqueda en el árbol. El producto también tiene asociado un campo "valor", numérico, siempre mayor que cero. La gerencia de la empresa nos ha solicitado que le indiquemos cuál es el producto que tiene el mayor valor, entre un cierto rango de claves.

Este algoritmo debe visitar la menor cantidad posible de nodos del árbol.

Tipo Arbol Binario de Busqueda

Raiz de Tipo Elemento Arbol Binario

mayorValor (claveMenor, claveMayor) devuelve el **código** del producto con mayor valor en árbol, dentro del rango, o 0 si no hay ningún producto en el rango indicado

Tipo Elemento Arbol Binario

Clave: numero
Datos: Tipo Producto
HijoIzquierdo, HijoDerecho: Tipo Elemento Arbol Binario

mayorValor (claveMenor, claveMayor) devuelve un número positivo o cero si no hay ningún producto en el rango indicado

Tipo Producto

Codigo: numero
Valor: numero.

Al invocar al método del árbol para el ejemplo:

- mayorValor (3,8) devuelve 24
- mayorValor (1,3) devuelve 5
- mayorValor (19,55) devuelve 0