GUIA De ejercicios – DyV – Arboles – Hash - Monticulos

Ejercicio 01

Calcular la eficiencia de los siguientes algoritmos:

a.

```
1. i = 1
2. Mientras ( i <= n)
3.      j = 1
4.      mientras ( j <= n)
5.              j = j * 2
6.      fin_mientras
7.      i = i + 1
8. Fin_mientras
```

b,.

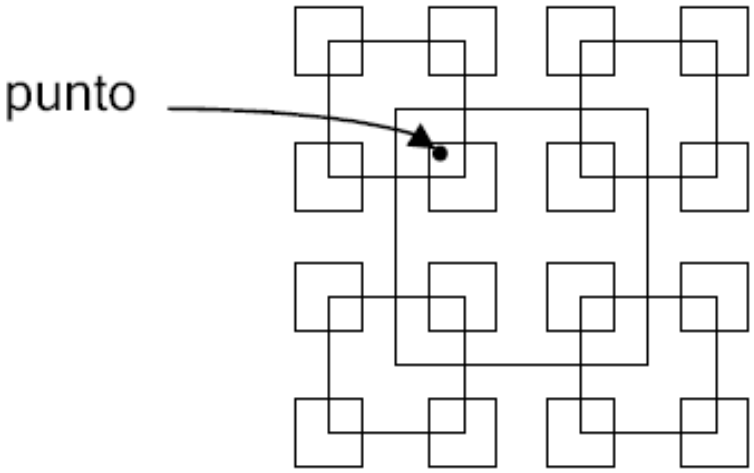
```
1. i = 1
2. Mientras ( i <= n)
3.      j = 1
4.      mientras ( j <= i)
5.              j = j +1
6.      fin_mientras
7.      i = i + 1
8. Fin_mientras
```

c.

```
1. i = 1
2. Mientras ( i <= 10)
3.      j = 1
4.      mientras ( j <= 10)
5.              j = j + 1
6.      fin_mientras
7.      i = i + 2
8. Fin_mientras
```

Ejercicio 02: DyV - Fractales

Determine cuantos cuadrados rodean a determinado punto. Los cuadrados se generan a través de un patrón recursivo, basado en un número k, que será el lado del cuadrado que a su vez tendrá en cada vértice un cuadrado con lado k/2. El mínimo valor de k es 2.



En el cuadrado del ejemplo, si tomamos en cuenta solo los cuadrados que se ven representados, la respuesta sería que tres cuadrados rodean al punto.

- Diseñar una clase patrón que responda a un método int rodean (int x, int y) que resuelva el problema de forma recursiva.
- Calcular la complejidad del método en términos de la notación O-Grande.

Ejercicio 03: DyV - Encontrar el piso de un arreglo

- Dado un arreglo ordenado de enteros, y un número entero X, se determina como piso del arreglo, al mayor número del mismo que es menor que X.
- Por ejemplo, en el arreglo
  - {2,2,3,5,6,8,12,45,67,89,112}
  - El piso de 4 es 3
  - El piso de 40 es 12
  - El piso de 1 es -1
  - El piso de 150 es 112
- Implementar un método recursivo para resolver este algoritmo teniendo en cuenta que recibe como argumentos el arreglo, y el entero X que determina el piso.

Ejercicio 04: Encontrar la mediana

- Dado un arreglo de enteros, **no ordenado**, implementar un algoritmo **recursivo** que encuentre la mediana (No deberá **ordenar el arreglo)**:
  - El valor central de una serie de datos, a cuyos lados queda la misma cantidad de datos.

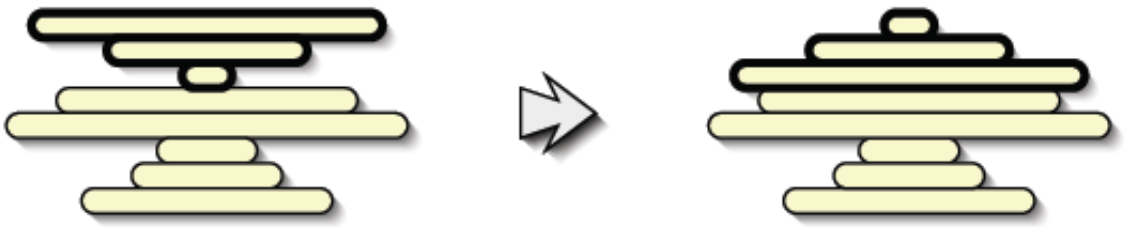
- En caso de ser par elegir como mediana el valor que se encuentra en N/2 exactamente.
- 1 2 3 5 6 6 7 8 9 → N = 9 ➔ mediana redondeo.abajo(9/2) =4
- 1 2 3 5 6 6 7 8 9 9 → N = 10 ➔ mediana 10/2 = 5

Ejercicio 05: Pancake sort

Suponga que tiene una pila con “panqueques” de diferente tamaño.

Se desea ordenar la pila, de forma tal que el más ancho esté en la parte inferior y el más angosto en la parte superior.

La única operación que puede realizar es dar vuelta con una espátula N panqueques (como mínimo 1 panqueque como máximo N).



Ejercicio 06 – Estructuras de datos

a) Dibuje como se insertarían los siguientes valores 93, 75, 83, 35, 54,26,12,37 en una tabla hash de 10 posiciones usando exploración lineal y luego inserte los mismos valores empleando exploración cuadrática.
b) Dibuje como se insertarían los siguientes valores en un montículo binario, si vienen en el siguiente orden:77, 23, 38, 48, 20, 19 , 29,26,31, 54 36. Luego muestre cómo evolucionará este montículo binario si remuevo 2 veces el menor elemento. **** cada vez que se inserta un elemento en el montículo debe reorganizarlo para que se cumplan sus propiedades en caso de que se violen***
c) Insertar los siguientes valores en un árbol Binario de Búsqueda.
    a. Valores a insertar: 145,49,22,198,38,164,225,14,201, 172.
    b. Luego eliminar el 14 y luego el 49.
    c. Repita el mismo ejercicio pero en un árbol equilibrado AVL y muestre cómo evoluciona, e indique, de ser necesario, si realiza rotaciones y el tipo de rotación que realiza

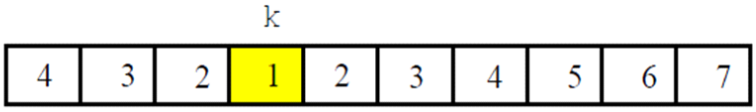
Ejercicio 07 – Divide y venceras

Sea ***v*** un vector de componentes "Integer" positivas que se ajustan al perfil de **una curva cóncava**, es decir, que existe una única posición ***k*** en el vector tal que:

* Los elementos a la izquierda de ***k*** están ordenados descendentemente.
* Los elementos a la derecha de ***k*** están ordenados ascendentemente.

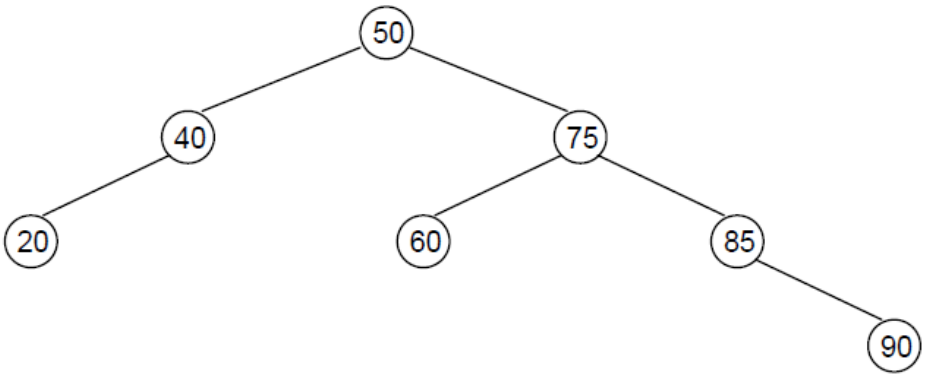
Diseñar un algoritmo DyV para encontrar dicho elemento (asumiendo que existe en un vector y es único)

Ejemplo:



Ejercicio 08 – AVL

Dado el siguiente árbol AVL, que almacena Valores Enteros.



Indique el factor de equilibrio de cada nodo del árbol:

Nodo	FE
50	
40	
10	
75	
60	
85	
90	

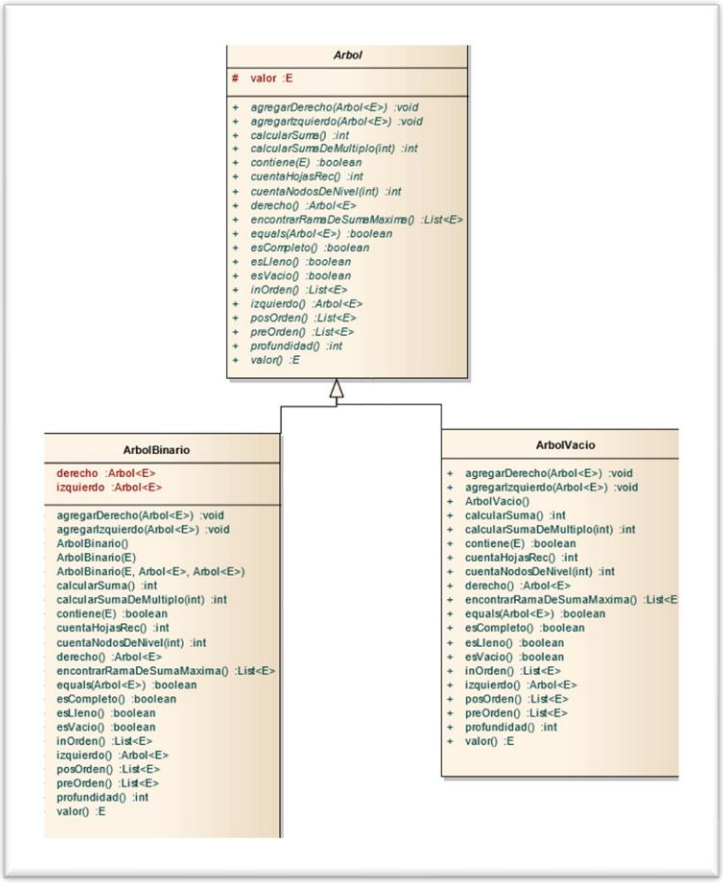
a) Dibuje el mismo árbol luego de realizar cada una de las siguientes operaciones (siempre comienza con el árbol de la figura, no con el que resulta de acumular las operaciones):
    1. Insertar la clave 10.
    2. Insertar la clave 95
    3. Insertar la clave 80 y luego la clave 77.
    4. Insertar la clave 80 y luego la clave 83
    5. Insertar la clave 45
    6. Insertar la clave 14 y luego borrar la clave 14.
    7. Insertar la clave 30 y luego borrar la clave 30.
    8. Insertar la clave 88 y luego borrar la clave 88.
    9. Insertar la clave 93 y luego borrar la clave 93.
b) ¿Luego de ejecutar las ultimas 4 operaciones (items 6, 7, 7 , 9) el árbol queda de la misma manera que antes de ejecutarlas?

Ejercicio 09 – AVL - HASH y MONTICULO

a) Dada la secuencia: 5 -10 – 15- 20 -23- 28 – 30 – 40
    a. Muestre el árbol binario de búsqueda correspondiente
    b. Muestre el árbol binario AVL correspondiente
    c. Muestre el montículo que se formaría y elimine 3 veces el mínimo.
    d. Muestre una tabla hash con 13 elementos usando exploración cuadrática
b) Dada la secuencia 4 19 -7 49 100 0 22 12
    a. Muestre el árbol binario de búsqueda correspondiente
    b. Muestre el árbol AVL correspondiente.
    c. Muestre el montículo que se formaría y e inserte 2 veces un valor menor que la raíz.
c) Dibujar la estructura del árbol que se produce luego de insertar en el orden en que aparecen los valores: 14,6,24,35,17,21,32,4,7,15,22.
d) Al árbol del punto c) eliminarle el nodo raíz. Hacerlo tantas veces como sea necesario hasta que se desequilibre un nodo y se deba aplicar rotación simple.
e) Mostrar un ejemplo donde la misma secuencia de valores ingresados, pero en distinto orden, genere dos árboles AVL distintos.

Ejercicio 10 – Arboles binarios de búsqueda

Usando el siguiente modelo recursivo de Árbol Binario de Búsqueda

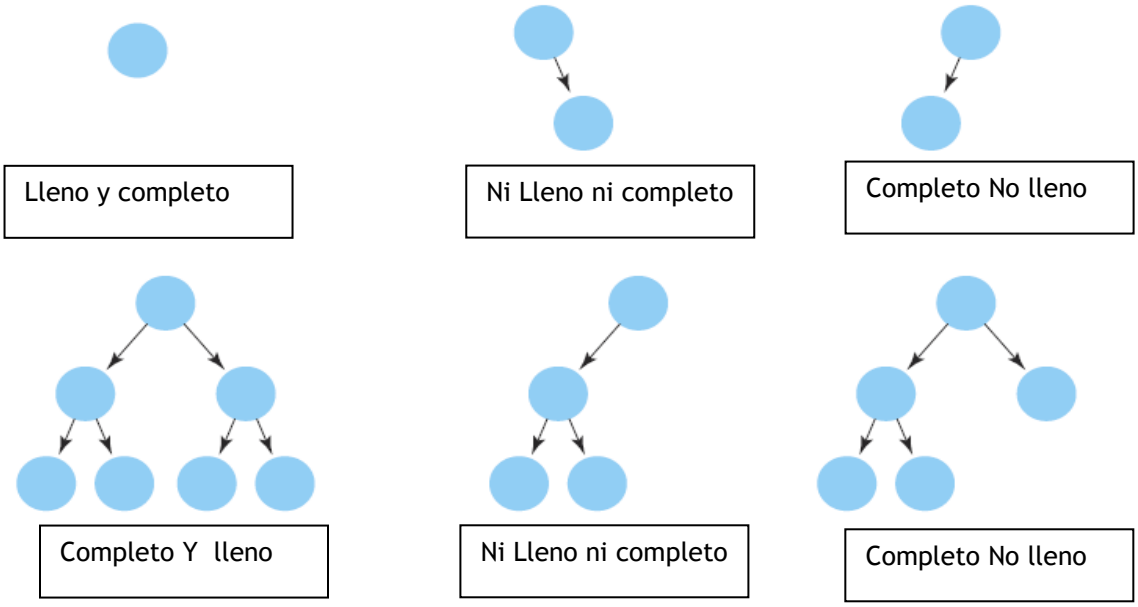


Implemente en las subclases de **Arbol, ArbolVacio y ArbolBinario** los siguientes métodos recursivos para resolver lo que se le indica

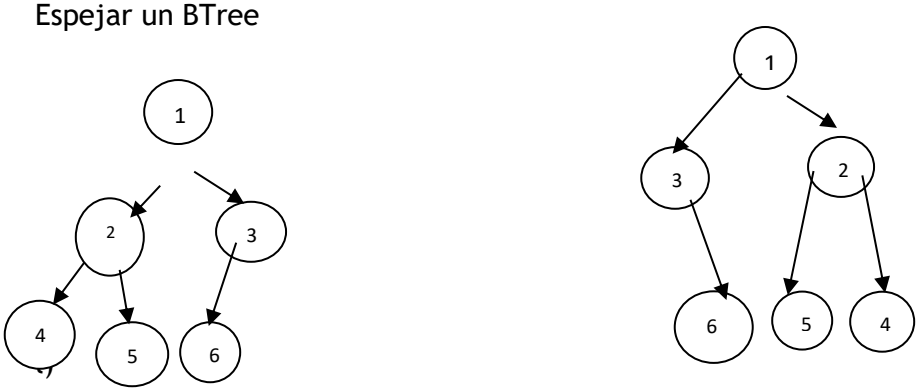
a) `public abstract boolean contiene(E unValor)` → retorna true si un elemento existe en el árbol.
b) `public abstract boolean equals(Arbol2<E> unArbol)` :método recursivo que retorna true si un árbol binario es idéntico a un recibido como parámetro.
c) `public abstract int profundidad()`:método recursivo que cuenta las altura de un árbol binario.
d) `public int cuentaHojasRec()`:método recursivo que cuente las hojas de un árbol binario.
e) `public int cuentaNodosDeNivel(int nivel)` : método que determina el número de nodos que se encuentran en un nivel N de un árbol.
f) `public boolean esLleno()`:método que determina si el árbol binario es lleno[1]

[1] Un árbol binario de nivel N es **lleno** cuando el máximo número de nodos permitidos en cada uno de los niveles..

g) `public boolean esCompleto()`: método que determina si el árbol binario es completo[2]
h) `public int calcularSuma()`: método que retorna la suma de todos los nodos del árbol (suponiendo que todos los nodos son de tipo entero)
i) `public List<E> camino(E v1,E v2)`: retorna el camino entre v1 y v2, si existe o null si no existe.
j) `public Arbol espejar()`: retorna el mismo árbol binario pero intercambia los hijos a izquierda y derecha (ver ejemplo).
k) `public Boolean esZurdo()`: Un árbol binario es zurdo si
   a. Es el árbol vacío
   b. Es una hoja
   c. Es un árbol donde más de la mitad de sus descendientes están en el hijo izquierdo y además tanto sus hijos izquierdos y como derecho son zurdos (es decir o son un árbol vacio, o son un nodo hoja, o cumplen el punto c)



Espejar un BTree



[2] Un árbol binario de nivel N es **completo** cuando para cada nivel desde el nivel 0 al nivel n-1, tiene un conjunto lleno de nodos (es decir tiene el máximo número de nodos permitidos para ese nivel), y en el nivel n, todos los nodos hoja ocupan las posiciones más a la izquierda del árbol.