Estructuras de Datos 2021/22 – Segundo Parcial
20 de enero de 2022
15:00 – 17:00
Aulas: P2.01, P2.04, P2.09 y P2.11

Nombre, Apellidos y DNI:

Pregunta 1. (3 puntos)

Dado el siguiente max-heap:

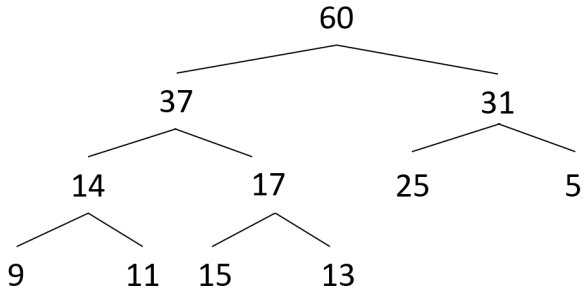

Figura 1. Max-heap de la Pregunta 1

Realiza las siguientes operaciones:

- *Remove* sobre el max-heap original.
- *Insert del elemento 65* sobre el max-heap original.

Explica en cada parte de esta pregunta (*remove* e *insert*) las situaciones que se dan en cada momento. Debes explicar el proceso seguido, así como los pasos y resultados intermedios para llegar al resultado final.

Pregunta 2. (3,5 puntos)

Dadas dos tablas hash vacías (**tabla 1** y **tabla 2**), como la de la Figura 2-a, dibuja paso a paso, cómo quedan las tablas con las operaciones indicadas en la Figura 2-b. Es necesario explicar brevemente qué ocurre y cómo se actúa en cada nueva situación, es decir, la primera vez que ocurre una nueva situación, deberá ser explicada pero no en las siguientes veces que ocurra.

Información común de ambas tablas:

- Función de dispersión a utilizar:
  $h(k) = ((\sum \text{dígitos de } k) + k + 1) \bmod 10$

  Por ejemplo:
  $h(43) = ((\sum \text{dígitos de } 43) + 43 + 1) \% 10 = (7 + 43 + 1) \% 10 = 51 \% 10 = 1$

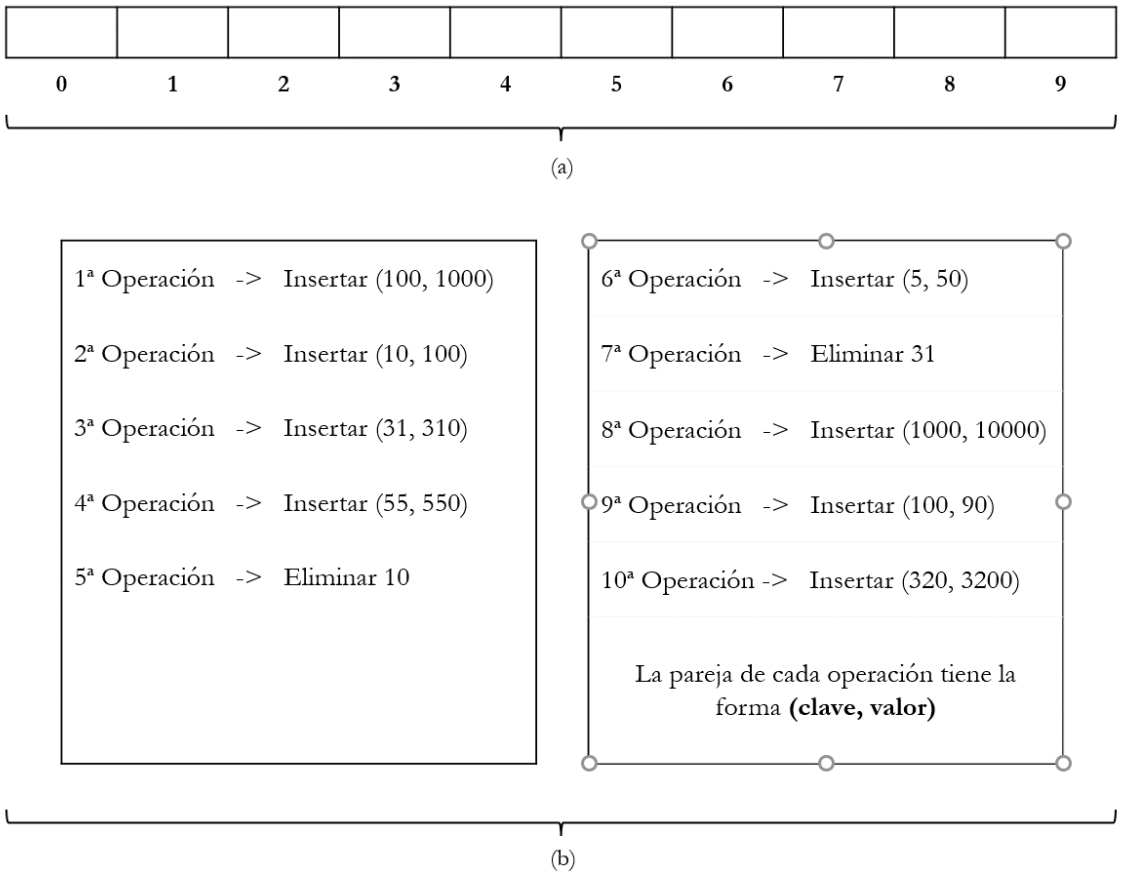

Figura 2. Tabla (a) y operaciones (b) de la pregunta 2

Información sobre la **tabla 1**:

- La tabla de dispersión se representa por medio de un vector de nodos enlazados de parejas (clave, valor).
- Estrategia de dispersión utilizada: dispersión abierta.

Información sobre la **tabla 2**:

- La tabla de dispersión se representa por medio de un vector de tripletas (clave, valor, marca).
- Estrategia de dispersión utilizada: dispersión cerrada.

  $h_i(k) = (h(k) + i^2 + 1) \bmod 10$

  h(k) es la función de dispersión previamente comentada

  i es el número de intentos para buscar una nueva posición libre (0, 1, 2, ...)

  - Por ejemplo, para la pareja con clave 43 la secuencia de posiciones en las que buscar una posición libre será la siguiente:
    1. $h_0(43) = (h(43) + 0^2 + 1) \% 10 = (1 + 0 + 1) \% 10 = 2 \% 10 = 2$
    2. $h_1(43) = (h(43) + 1^2 + 1) \% 10 = (1 + 1 + 1) \% 10 = 3 \% 10 = 3$
    3. $h_2(43) = (h(43) + 2^2 + 1) \% 10 = (1 + 4 + 1) \% 10 = 6 \% 10 = 6$
    4. …

Pregunta 3. (3,5 puntos)

Implementa un método, llamado *leavesMinusInternal,* que reciba un *BinaryTree<Integer>* y devuelva el resultado de restar la suma de los nodos interiores a la suma de las hojas. Si el árbol es vacío, devolverá 0. A continuación, en la Figura 3, se muestran dos ejemplos.

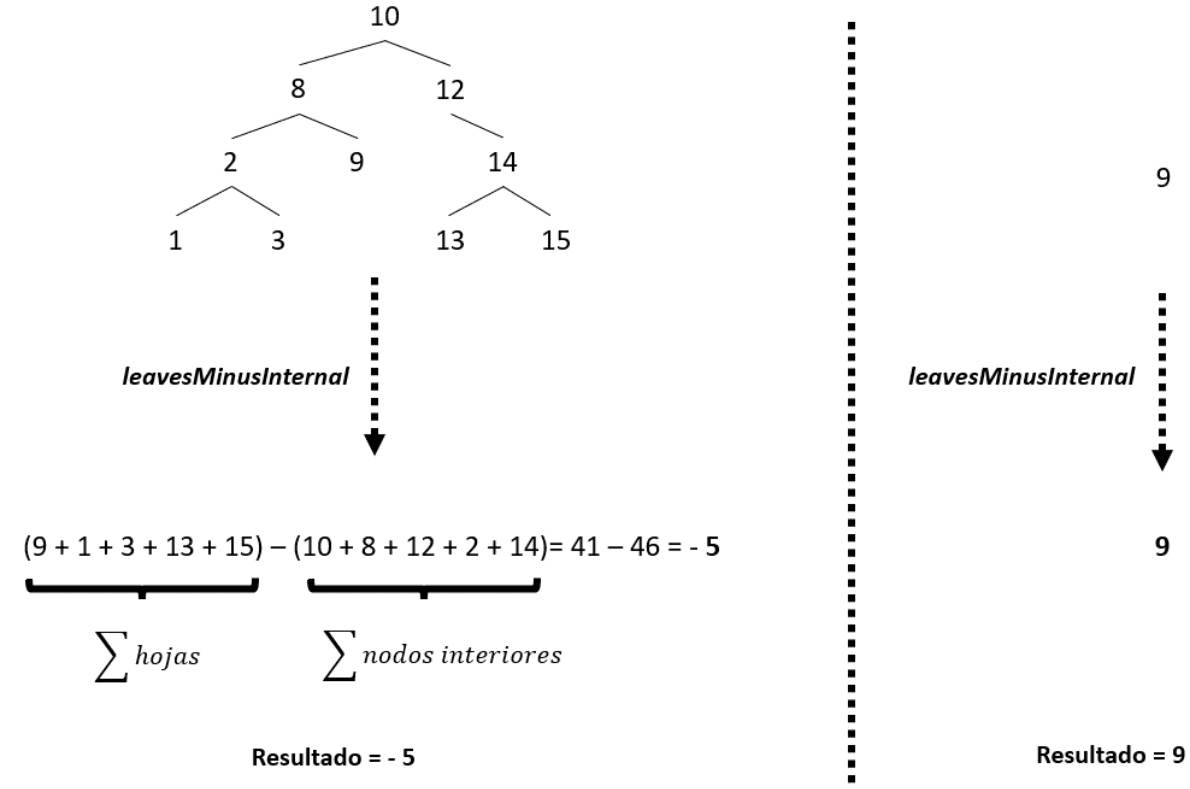

Figura 3. Ejemplos de la Pregunta 3

Información adicional

```
public interface BinaryTree<E> extends Collection<E> {
    BinaryTree<E> left();
    BinaryTree<E> right();
    E root();
    void removeLeft();
    void removeRight();
    int height();
    iterator<E> iterator();
    BinaryTreeIterator<E> iteratorPre();
    BinaryTreeIterator<E> iteratorIn();
    BinaryTreeIterator<E> iteratorPost();
    BinaryTreeIterator<E> iteratorLevels();
    void clear();
    boolean contains(Object o);
    boolean containsAll(Collection<?> c);
    boolean equals(Object o);
    int size();
    Object[] toArray();
    boolean isEmpty();
}
```