UT9 – TA6

**Insertar:**

* **Descripción**: Agregar un nuevo elemento en el APO, y después, ajustar el árbol para mantener la propiedad de heap (el elemento más pequeño debe estar en la raíz).
* **Pasos básicos**:
  1. Añadir el nuevo elemento al final del heap.
  2. Realizar un "up-heap" (también llamado "heapify-up"): Comparar el nuevo elemento con su padre y, si es menor, intercambiarlos. Repetir este proceso hasta que el nuevo elemento esté en su posición correcta.

**suprimeMinimo:**

* **Descripción**: Eliminar el elemento mínimo (que está en la raíz) y reorganizar el heap para mantener su propiedad.
* **Pasos básicos**:
  1. Reemplazar la raíz con el último elemento del heap.
  2. Realizar un "down-heap". Comparar el nuevo elemento en la raíz con sus hijos, y si alguno de los hijos es menor, intercambiarlos. Repetir este proceso hasta que el elemento esté en su posición correcta.

**2)**

**Insertar:**

* **Orden de complejidad**: O(logN)
* **Justificación**: En un árbol binario balanceado, la altura del árbol es logN. En el peor de los casos, el nuevo elemento puede necesitar moverse desde una hoja hasta la raíz, lo que requiere logN comparaciones e intercambios.

**suprimeMinimo:**

* **Orden de complejidad**: O(logN)
* **Justificación**: Similar a la operación "insertar", en un árbol binario balanceado, al suprimir el elemento mínimo (la raíz) y mover el último elemento a la raíz, este nuevo elemento puede necesitar moverse desde la raíz hasta una hoja, lo que nuevamente requiere logN comparaciones e intercambios.

**3)**

**Para un nodo en la posición i**:

* + El índice de su padre es (i-1) / 2
  + Los índices de sus hijos son 2i + 1 + y 2i + 2.

**a) Operación "insertar":**

* **Orden de complejidad**: O(logN)
* **Justificación**: El nuevo elemento se inserta al final del vector y se ajusta hacia arriba. Cada ajuste (intercambio) se realiza en tiempo constante, y el número máximo de ajustes es proporcional a la altura del heap, que es logN.

**b) Operación "suprimeMinimo":**

* **Orden de complejidad**: O(logN)
* **Justificación**: Al eliminar el elemento mínimo (el primer elemento del vector), el último elemento se mueve a la raíz y se ajusta hacia abajo. Cada ajuste (intercambio) se realiza en tiempo constante, y el número máximo de ajustes es proporcional a la altura del heap, que es logN.