Actividad 4. Heap y HeapSort

Objetivo

Conocer las propiedades de los montículos binarios (*Heap*) y saber implementar esta estructura de datos.

Procedimiento

1. Ver el video o leer la presentación sobre árboles binarios que están disponibles en Moodle, Tema 1/ Sección 1.3 Heap/ Recursos didácticos.
2. Resolver los ejercicios que se proponen en esta actividad.
3. Para probar el correcto funcionamiento de los métodos se puede hacer uso del test disponible en Moodle, Tema 1/ Sección 1.3 Heap/ Actividades Grupo Reducido.

Evaluación

Estos contenidos serán evaluados mediante una prueba individual el 21 de octubre de 2022.

Tiempo estimado

6 horas

Ejercicios

1. Dada la especificación del TAD Heap (disponible en el Anexo) escribir un proyecto en java para implementar dicho TAD.

   ```
   public class Heap<E extends Comparable<E>>
   ```

2. Se han añadido dos métodos nuevos a la clase Heap:
   - Método **introducir**(): añade un objeto, pero no garantiza que se mantenga la propiedad de ordenación del heap.
   - Método **arreglarHeap**(): restablece el orden en el montículo. Debido a que es costoso, su uso está justificado si se realizan muchas operaciones *introducir*() entre dos accesos al elemento de mayor prioridad. Este método llama al método *hundir* (utilizado al suprimir) sobre cada nodo en sentido inverso al recorrido por niveles; cuando se realice la llamada con el nodo i se habrán procesado todos los descendientes del nodo i con una llamada a hundir. No hace falta ejecutar hundir sobre las hojas, por lo que se comienza con el nodo de mayor índice que no sea una hoja.

   Añade estos dos nuevos métodos a la clase Heap y haciendo uso de ellos implementa el **algoritmo de ordenación HeapSort.** Una implementación eficiente de este algoritmo sería (i) introducir() cada elemento en un montículo binario, (ii) llamar a arreglarHeap() y (iii) llamar a suprimirMax() tantas veces como elementos haya en el montículo. Los elementos saldrán del montículo en orden.

Anexo

```
public class Heap<E extends Comparable<E>>{
        public Heap();
        public boolean esVacio();
        public E recuperarMax() throws HeapVacioExcepcion;
        public E suprimirMax() throws HeapVacioExcepcion;
        public void insertar(E e) throws IllegalArgumentException;
        public void anular();
}
```