Actividad 4. Heap y HeapSort

Objetivo

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

Procedimiento

1. Leer y comprender los apuntes sobre heaps que están disponibles en **Tema**, sección **Documentos e Ligazóns** / **Teoría** / **Heaps**.
2. Resolver los ejercicios que se indica en esta actividad, utilizando el lenguaje java. Para probar el correcto funcionamiento del ejercicio 1 se puede hacer uso de los tests disponibles en Tema, sección Documentos e Ligazóns / Actividades / Test / **HeapTest.java.**

Evaluación

Estos contenidos serán evaluados mediante una prueba individual el 22 de octubre de 2019.

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 haciendo uso de un array.

   ```
   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

     ```
     public void arreglarHeap(){
       //Modifica: this
       //Produce: un heap a partir de un árbol completo sin orden
       for(int i = tamAct/2; i > 0 ; i--)
          hundir(i);
       }
     ```

     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 suprimir() 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 recuperar() throws HeapVacioExcepcion;
        public E suprimir() throws HeapVacioExcepcion;
        public void insertar(E e) throws IllegalArgumentException;
        public void anular();
}
```