ESTRUCTURA DE DATOS

- GRADO EN INGENIERÍA EN INFORMÁTICA
- ENUNCIADOS DE LABORATORIOS (ACTUALIZACIÓN 3/11/2025)
- CURSO ACADÉMICO 2025/26
- PROFESORADO: SERGIO SAYAGO

Universitat de Lleida

Campus Universitari Igualada-UdL

2 LABORATORIOS

- Laboratorio I (análisis de algoritmos)
- Laboratorio II (programación orientada a objetos)
- Laboratorio III (estructuras de datos de acceso secuencial)
- **Laboratorio IV (estructuras de datos en forma de árbol)**
- Laboratorio V (tablas de dispersión)

3 LABORATORIO IV
IMPLEMENTAR UN HEAP GENÉRICO

- Una cola de prioridad (un HEAP) es una estructura de datos que nos permite acceder muy rápidamente - O(1) - al mínimo elemento, aquel que tiene la mayor prioridad
  - Elemento con prioridad 2 está por delante de elementos con prioridad 3 y superiores
- En este laboratorio implementaremos un HEAP genérico de elementos que serán Comparables
  - Haremos una simulación de una cola de impresión compartida de documentos de diferentes usuarios y prioridades

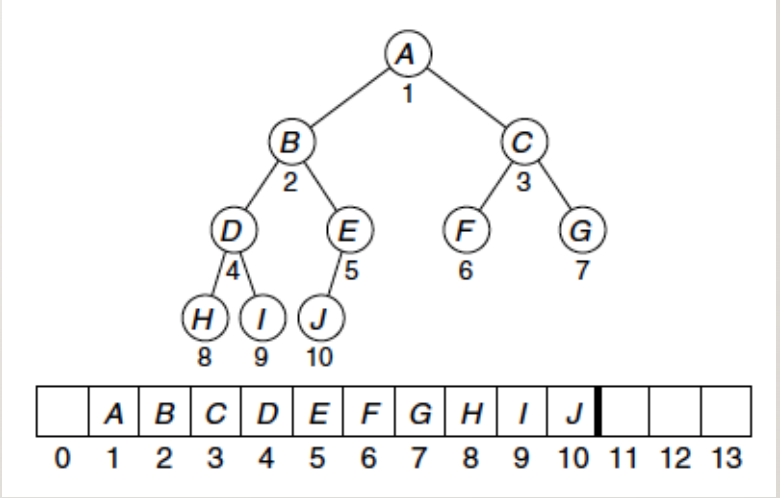
4 LABORATORIO IV

IMPLEMENTAR UN HEAP GENÉRICO – EN LA NUEVA PROGRAMACIÓN

- En esta práctica nos ayudaremos de la IAG (Inteligencia Artifical Generativa) para programar la solución
- Lo que espero que hagáis es:
  - Integrar la IAG como una herramienta más para vuestra educación
  - Escribir los prompts para que la IAG os escriba el código
  - Entender el código que os ha proporcionado la IAG (con su ayuda)
  - Integrar el código en un proyecto y que funcione
  - Discutir en la entrega (y en la entrevista) el uso de la IAG en la práctica

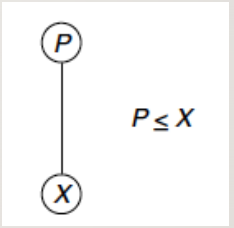
5 LABORATORIO IV
IMPLEMENTAR UN HEAP GENÉRICO CON VECTOR

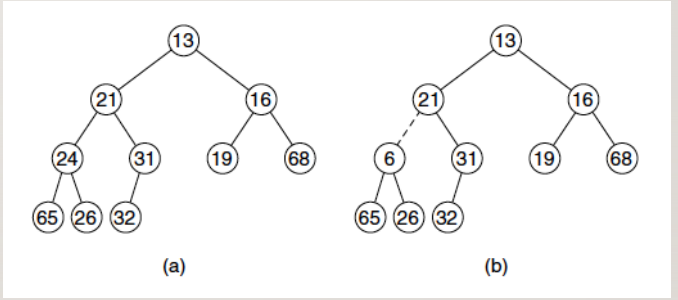
Implementaremos el HEAP como un árbol binario completo y equilabrado, utilizando un vector



6 LABORATORIO IV
IMPLEMENTAR UN HEAP GENÉRICO

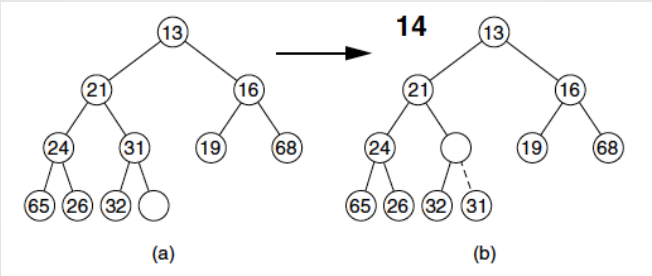
- **Propiedad del orden del montículo**
- El mínimo elemento (con mayor prioridad) siempre está en la raíz
- Aplicando la recursividad, cualquier nodo tiene que ser más pequeño o igual que todos sus descendientes

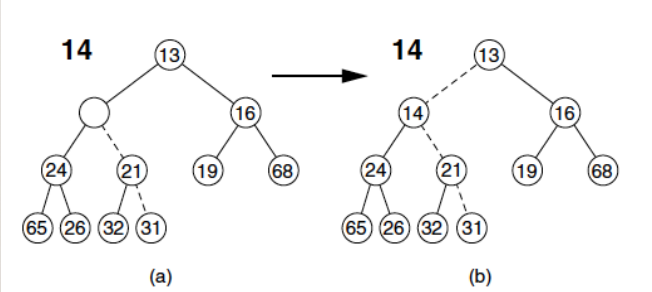




a) es un HEAP, b) no es un HEAP

7 LABORATORIO IV
IMPLEMENTAR UN HEAP GENÉRICO - INSERCIÓN

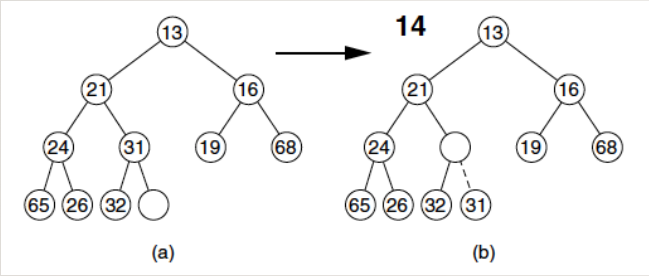


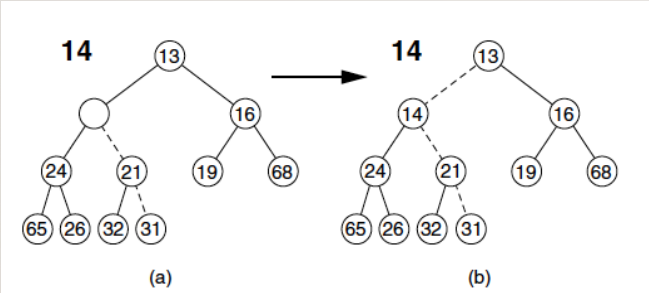


8 LABORATORIO IV
IMPLEMENTAR UN HEAP GENÉRICO - INSERCIÓN

Inserción. Para insertar un elemento X en el montículo, primero debemos añadir un nodo en el árbol. Esto lo haremos creando una posición en el vector, que será la siguiente posición a ocupar. Si no hacemos esto, el árbol no será completo. Si el elemento se puede insertar satisfaciendo la propiedad de orden, el padre es más pequeño que cualquiera de sus hijos, hemos acabado. De lo contrario, movemos el padre del nuevo nodo a dicho nodo, y repetimos el proceso – vamos subiendo hacia arriba - hasta que colocamos el nuevo nodo. El hecho de subir nos obliga a situar el elemento más pequeño lo más arriba posible.

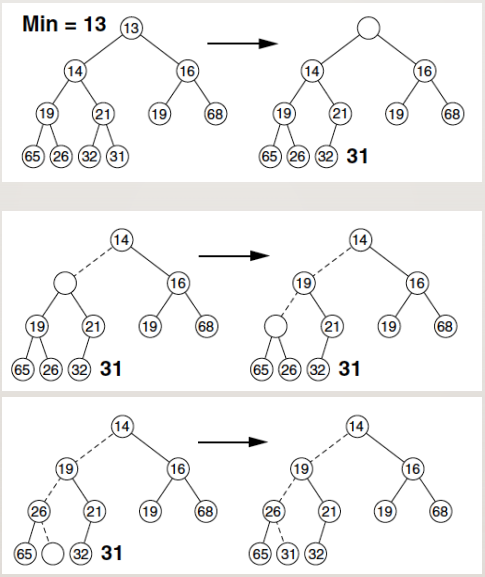
9 LABORATORIO IV
IMPLEMENTAR UN HEAP GENÉRICO - INSERCIÓN





En el siguiente montículo, queremos insertar el 14. Creamos un nuevo nodo, que tiene ser hijo derecho de 31 para que el árbol sea completo. El nodo 14 no lo podemos insertar en el nuevo nodo, porque su padre, 31, no es más pequeño que 14. Por tanto, bajamos 31 al nuevo nodo, y seguimos. El nodo 14 no lo podemos insertar como hijo derecho de 21, porque su padre, 21, sería más grande que uno de sus descendientes. Por tanto, bajamos 21. Ahora si podemos insertar 14, porque 13, su padre, es más pequeño.

10 LABORATORIO IV
IMPLEMENTAR UN HEAP GENÉRICO - ELIMINACIÓN

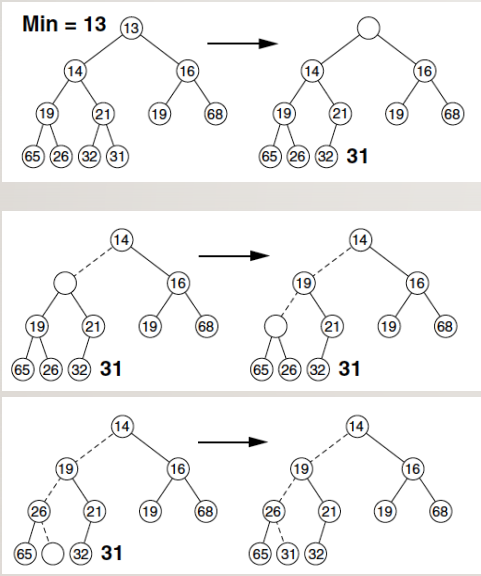


11 LABORATORIO IV

IMPLEMENTAR UN HEAP GENÉRICO - ELIMINACIÓN

Eliminación. Eliminaremos siempre el mínimo, que está en la raíz del árbol. Se implementa de forma similar a la operación de inserción. Cuando se elimina el mínimo, se crea un agujero en el montículo. Tenemos un elemento menos en el árbol, y esto significa que el último elemento se tiene que eliminar – es decir, poner en algún otro sitio. Para tapar el agujero, y reducir en 1 el número de nodos en el árbol, si pudiéramos poner el último elemento arriba del todo, ya estaríamos. Sin embargo, esto es imposible, porque se espera que los nodos en niveles inferiores sean más grandes que los superiores, pero lo tenemos que poner en algún sitio! Entonces lo que hacemos es poner algún elemento en el agujero, y nos movemos hacemos abajo. Concretamente, lo que hacemos es buscar el hijo más pequeño del nodo agujero. Si dicho nodo es más pequeño que el nodo que queremos poner en el agujero (el último), entonces movemos dicho hijo al nodo agujero. Esto crea un nuevo agujero, en un nivel inferior, y repetimos el proceso, hasta que el último nodo del árbol se encuentre en una posición correcta.

12 LABORATORIO IV
IMPLEMENTAR UN HEAP GENÉRICO - ELIMINACIÓN



En el siguiente montículo, eliminamos el 13. Esto crea un agujero. Fijamos el último nodo (31) para ver dónde ubicarlo. Buscamos el hijo menor de 13. Es 14. Como 14 < 31, lo colocamos en el agujero, y creamos el agujero en el nivel inferior. Buscamos hijo menor. Es 19. Como 19 < 31, lo colocamos en el agujero, y creamos un nuevo agujero, en un nivel inferior. Buscamos hijo menor. Es 26. Como 26 < 31, lo subimos, y tenemos el agujero en nodo hoja. Como no tiene hijos, 31 pasa a ser hijo derecho de 26.

13

LABORATORIO IV
IMPLEMENTAR UN HEAP GENÉRICO - TAREAS

- **Tarea 1. Implementar la clase Documento**
  - Tendrá dos atributos, nombre del autor del documento y prioridad
  - Es comparable (es decir, implementa la interfaz Comparable<T>)
  - La comparación de documentos se hará según el atributo prioridad

14 LABORATORIO IV
IMPLEMENTAR UN HEAP GENÉRICO - TAREAS

- **Tarea 2**. Implementa la cola de prioridad con un vector
  - public class MyPriorityQueue <E extends Comparable<? super E>>
  - public boolean add (E x) {…} Añade un elemento al montículo cumpliendo con su propiedad
  - public E remove() {…}
  - public String toString() {…} Imprimir el montículo (es decir, el contenido de sus elementos)
  - Podéis utilizar métodos auxiliares

15 LABORATORIO IV
IMPLEMENTAR UN HEAP GENÉRICO - TAREAS

- **Tarea 3**. Implementar juego de pruebas en el main

```
MyPriorityQueue<Document> pq = new MyPriorityQueue<~>();
pq.add(new Document ( a: "Sergio",  p: 13));
pq.add(new Document ( a: "Juanen",  p: 14));
pq.add(new Document ( a: "Gimeno",  p: 16));
pq.add(new Document ( a: "Toni",  p: 19));
pq.add(new Document ( a: "Marta",  p: 21));
pq.add(new Document ( a: "Montse",  p: 68));
pq.add(new Document ( a: "Merce",  p: 65));
pq.add(new Document ( a: "Manel",  p: 26));
pq.add(new Document ( a: "Grau",  p: 32));
pq.add(new Document ( a: "Adela",  p: 31));
System.out.println(pq.toString());
Document aux = pq.remove();
System.out.println("---");
System.out.println(aux.toString());
Document aux2 = pq.remove();
System.out.println("---");
System.out.println(aux2.toString());
```

```
Documento de: Adela con prioridad: 31
Documento de: Sergio con prioridad: 13
Documento de: Juanen con prioridad: 14
Documento de: Gimeno con prioridad: 16
Documento de: Toni con prioridad: 19
Documento de: Marta con prioridad: 21
Documento de: Montse con prioridad: 68
Documento de: Merce con prioridad: 65
Documento de: Manel con prioridad: 26
Documento de: Grau con prioridad: 32
Documento de: Adela con prioridad: 31

---
Documento de: Sergio con prioridad: 13
---
Documento de: Juanen con prioridad: 14
```

LABORATORIO IV
IMPLEMENTAR UN HEAP GENÉRICO - TAREAS

- **Tarea 4**. Implementar un método auxiliar que, dado un heap implementado como en la práctica, compruebe que realmente lo es, en O(N).

0	1	2	3	4	5	6	7	8	9	10	11
	13	14	16	19	21	19	68	65	26	32	31

Este vector cumple con la propiedad del montículo vista en el enunciado.
Por tanto, es un heap (concretamente, un min-heap)

LABORATORIO IV
IMPLEMENTAR UN HEAP GENÉRICO - TAREAS

- **Tarea 5.** Utilizar la IAG. Concretamente:
  - Integrar la IAG como una herramienta más para vuestra educación
  - Escribir los prompts para que la IAG os escriba el código
  - Entender el código que os ha proporcionado la IAG (con su ayuda)
  - Integrar el código en un proyecto y que funcione
  - Discutir en la entrega (y en la entrevista) el uso de la IAG en la práctica

18 LABORATORIO III
ENTREGA

- En la actividad correspondiente del Campus Virtual
  - No hay estudio previo
  - Un único fichero ZIP con los siguientes ficheros:
  - Uso de la IAG (en PDF o DOCX)
    - Los prompts utilizados para realizar la práctica
    - Vuestra explicación del código generado por la IAG
    - El documento debe tener el nombre de los estudiantes y de la asignatura
  - Proyecto IntelliJ IDEA en ZIP con el código del *heap*
    - Nombres de los integrantes en el nombre del fichero, por ej. SergioJordi.zip

LABORATORIO III
CRITERIOS GENERALES DE EVALUACIÓN

- Práctica no entregada o entregada fuera de plazo = 0
- Entrega parcial, <= 3 puntos
- Entrega completa, entre 4 y 10 puntos, según:
  - Uso de la IAG
  - Entrevista
  - Código

20 LABORATORIOS

- Laboratorio I (análisis de algoritmos)
- Laboratorio II (programación orientada a objetos)
- Laboratorio III (estructuras de datos secuenciales)
- Laboratorio IV (estructuras de datos en forma de árbol)
- **Laboratorio V (tablas de dispersión)**