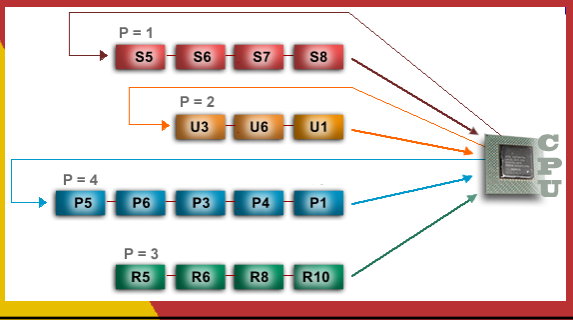
UNIVERSIDAD MAYOR

COLAS DE PRIORIDAD



Plan del MA08

PLAN DEL MÓDULO DE APRENDIZAJE 08

TAD Priority Queue

Aprendizajes esperados

Seleccionar la mejor estructura de datos para representar operaciones que involucren el concepto de cola de prioridad

1. *Aplicar TAD Pcola en la solución de un problema.*
2. *Implementar TAD Pcola genérico.*
3. *Evaluar las alternativas de implementación, considernado la eficiencia de los algoritmos.*

Contenido

- *Concpetos de árboles genéricos*
- *Conceptos de árboles binarios*
- *Características de un Heap*
- *Métodos de un Heap*
- *Implementación del TAD Pcola utilizando un Heap.*

UNIVERSIDAD MAYOR

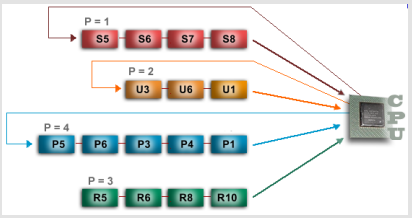
TAD Cola de Prioridad (pQueue)

✔ Una ***cola de Prioridad*** es un TAD, donde las operaciones de:

- ***Agregar*** se realiza según orden de llegada (Cola)
- ***Extraer*** se realiza según la prioridad del objeto

✔ **Operadores:**

- agregar(e)
- extraer()
- vacia()



UNIVERSIDAD MAYOR

Prioridad

✔ La prioridad es un valor numérico, asociado a cada elemento, que indica:

"Mayor importancia, menor valor"

✔ Ejemplo: Gestión de impresora:
Criterios de prioridad:

- Los trabajos más importantes primero
- Los trabajos más cortos primero

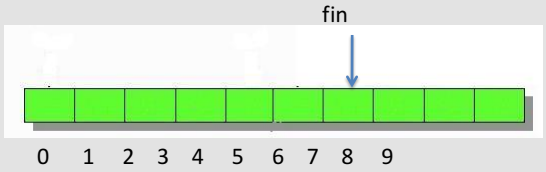
UNIVERSIDAD MAYOR

Alternativa de implementación

- ✔ Una lista ordenada en forma ascendente por la prioridad, permite:
  - extraer: Seleccionar el mínimo: *O(1).*
  - agregar: Requiere inserción, manteniendo en orden: *O*(n); en promedio debe efectuar un recorrido de *O*(n/2)
- ✔ *Una lista no ordenada:*
  - agregar: Tiene costo de inserción : *O(1)*
  - *extraer: Tiene costo de búsqueda: O(n).*

Alternativa de implementación

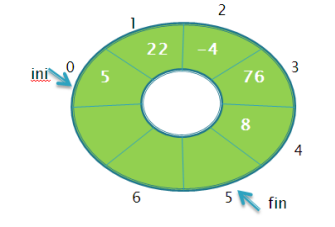
- ✔ Agregar un elemento a la cola de prioridad: O(cte)



- ✔ La "extracción" de un elemento de la cola de prioridad , implica:
  - La búsqueda del objeto de MAYOR prioridad. O(n)
  - Mantener ordenado el arreglo. $O(n^2)$

Alternativa de implementación

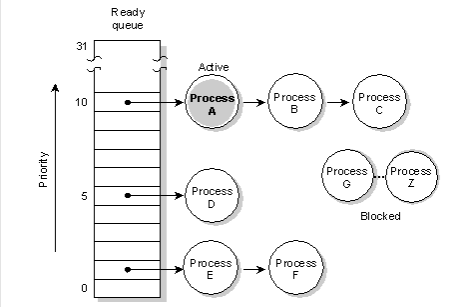
✔ Agregar un elemento a la cola de prioridad: O(cte)



✔ La "extracción" de un elemento de la cola de prioridad , implica la búsqueda del objeto de MAYOR prioridad. O(n)

Alternativa de implementación

agregar: ***O(p)***



UNIVERSIDAD MAYOR

Colas de prioridad

Heap

9

Alternativas de solución

E. Datos	buscarMin	eliminarMin	insertar
Lista	O(N)	O(N)	O(1)
Lista Ord	O(1)	O(1)	O(N)
ABB	O(log N)	O(log N)	O(log N)
Heaps	O(1)	O(log N)	O(log N)

Alternativa de implementación:

✔ Un ***heap*** es una estructura de datos que ***asegura:***
- La extracción del elemento de mayor prioridad en tiempo *O*(log n)
- La *agregación* de un nuevo elementos en tiempo *O*(log n).

✔ Gráficamente, un *heap,* es un **Árbol Binario completo o semi completo.** Es decir:
- Posee los niveles completos.
- Posee los niveles INTERNOS completos, a excepción quizás del último, en el cuál todas las hojas están situadas a la izquierda

UNIVERSIDAD MAYOR

HEAP

✔ Los elementos de un ***heap*** deben cumplir con el siguiente criterio o condición:
- ***La prioridad de un elemento "padre" es MAYOR que la de sus hijos.***

$$\forall\ i : 1 < i <= n : v[i/2] < v[i]$$

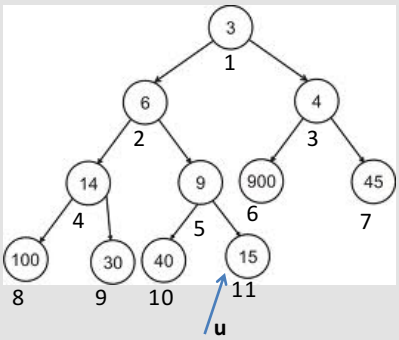
✔ Propiedades:
- Su altura es a lo sumo (log N)
- Admite una representación implícita sobre vector

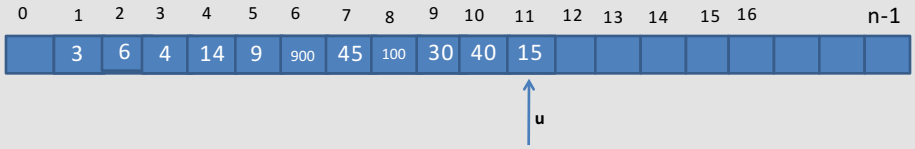
UNIVERSIDAD MAYOR

Heap

✔ Arreglo en el que se cumple:

Todo Padre tiene MAYOR prioridad que sus hijos





Clase PCola

```
public class Pcola<T>{
    private Elemento<T>[] v;
    private int u;
```

Clase Elemento

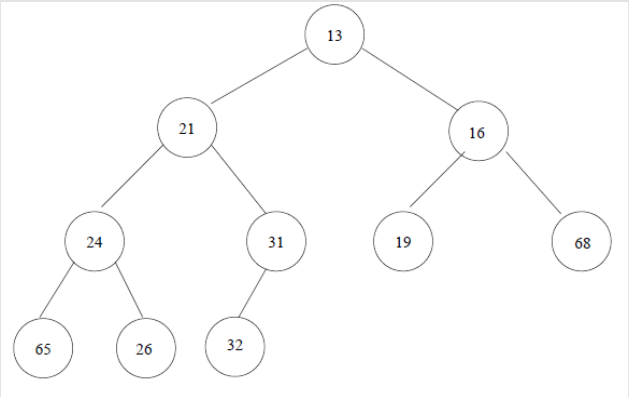
```
public class Elemento<T>{
    private int prioridad;
    private T info;
```

Condición del heap

- ✔ *Un elemento* cuya posición es **i**, tiene la ubicación (índice) del *padre en la posición :* _____
- ✔ Hijo IZQUIERDO de un elemento ubicado en la posición ***h:*** _____
- ✔ ¿Tiene hijo DERECHO un elemento ubicado en la posición ***h:*** _____
- ✔ Hijo DERECHO de un elemento ubicado en la posición ***h:*** _____
- ✔ Un elemento ubicado en la posición ***h,*** ¿Es padre?: _____

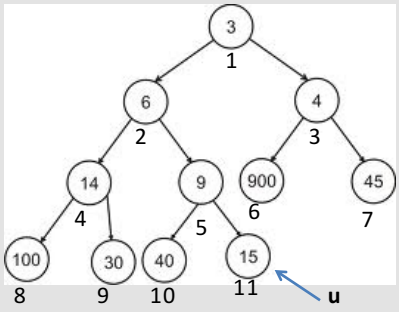
Ejercicio

- ✔ ¿Es un Heap?

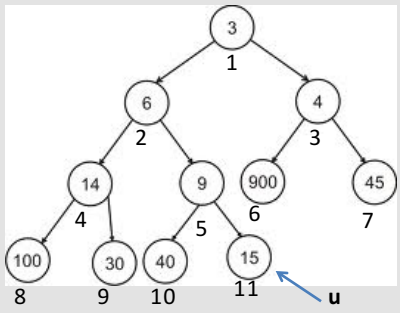


Agregar un elemento

- ✔ El Heap se "llena" de izquierda a derecha en el último nivel.
- ✔ Agrega al final del Heap : e=45
- ✔ Luego "**Restaura**" la condición del Heap.
- ✔ Se compara el NUEVO elemento con su posible padre. Se intercambian en caso que NO cumplan con la condición del Heap.



add (e)

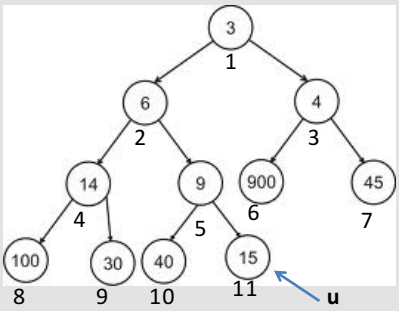



```
public void add(Elemento<T> e){
    u++;
    v[u]=e;
    subirHeap();
}
```

SubirHeap()

✔ A partir del Heap de la figura, agregue los elementos con las siguientes prioridades:

- 48
- 12
- 99
- 50



SubirHeap()

```
private void subirHeap(){
    int i, h;
    Elemento<T> aux;
    i=u;
    h=u/2;
    while(h>0 && v[h].getPrioridad()> v[i].getPrioridad()){

        aux=v[h];
        v[h]=v[i];
        v[i]=aux;

        i=h;
        h=i/2;
    }
}
```

Ejercicio

✔ Utilizando diagramas y las reglas de formación, generar un Heap utilizando los siguientes valores de prioridades, en la secuencia indicada:

60, 124, 30, 12, 50, 25, 42, 18, 37, 28, 55

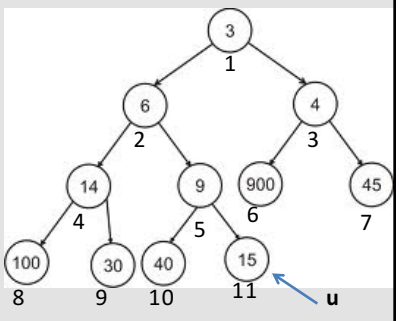
Extraer un elemento

✔ Se elimina el PRIMER elemento.

✔ El último elemento, se ubica en primera posición.

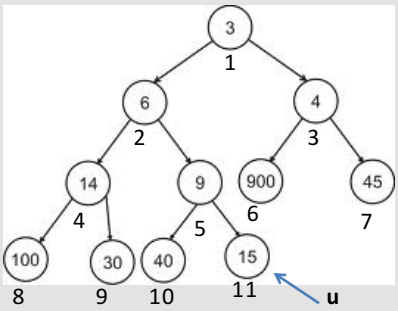
✔ Se restaura la condición del Heap, comparando la prioridad del padre con su hijo de mayor prioridad.

✔ Se intercambian, en caso de NO cumplir con la condición del heap.



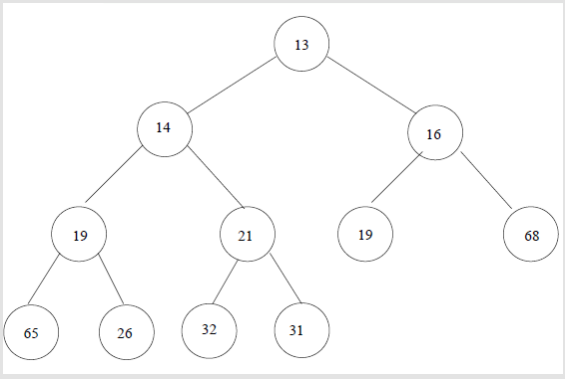
BajarHeap()

✔ Elimina 3 elementos del Heap del ejemplo:
- ¿Cuáles son?
- ¿Cómo queda el heap?

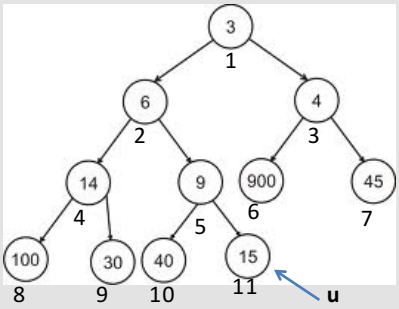


Ejercicio

✔Elimina el elemento de mayor prioridad



remove()




```
public Elemento<T> remove(){
    Elemento<T> e;
    e=v[1];
    v[1]=v[u];
    u--;
    bajarHeap();
    return e;
}
```


BajarHeap()

```
private void bajarHeap(){
    boolean ubicado=false;
    Elemento<T> e;
    int hijo,i, d;
    int k=1;

    while (k<=u/2 && !ubicado){
        i=2*k;
        hijo= i;
        if(i<u) {   //tiene hijo derecho. Elegir entre los dos hijos.
            d=2*k+1;
            if(v[i].getPrioridad() > v[d].getPrioridad())
               hijo=d;
        }

        if(v[k].getPrioridad()>v[hijo].getPrioridad()){
            e= v[k];
            v[k]=v[hijo];
            v[hijo]=e;
            k=hijo;
        }
        else{
            ubicado=true;
        }
    }
}
```

Ejercicios

En forma gráfica, represente los "árboles" resultantes al insertar los valores en la secuencia indicada, a partir de un Heap vacío:

6,4,15,2,10,11,8,1,13,7,9,12,5,3,14

UNIVERSIDAD MAYOR

PC- MA08

PROYECTO DE CÁTEDRA MA08

Atención ER

Descripción

La recepción de personas que llegan para ser atendidos en una unidad de emergencias médicas (Posta, ER, etc.) se realiza según la prioridad que la enfermera-recepcionista asigne a los pacientes luego de la primera revisión.
Cada Paciente está representado por los datos de una Persona más la prioridad asignada por la enfermera.

Implemente una aplicación que permita simular la llegada – a tención a pacientes en una ER.

Consideraciones

- Cuando un paciente llega a ER, la enfermera asigna una prioridad y el Paciente es "agregado" a la cola de espera.
- Asuma que en la sala de espera existen pacientes, cuyos datos están almacenados en el archivo pacientes.txt.
- Simule la atención del paciente de mayor prioridad, cada vez que se "haga un llamado" desde el box.
- Utilice la clase Enum para identificar prioridades.