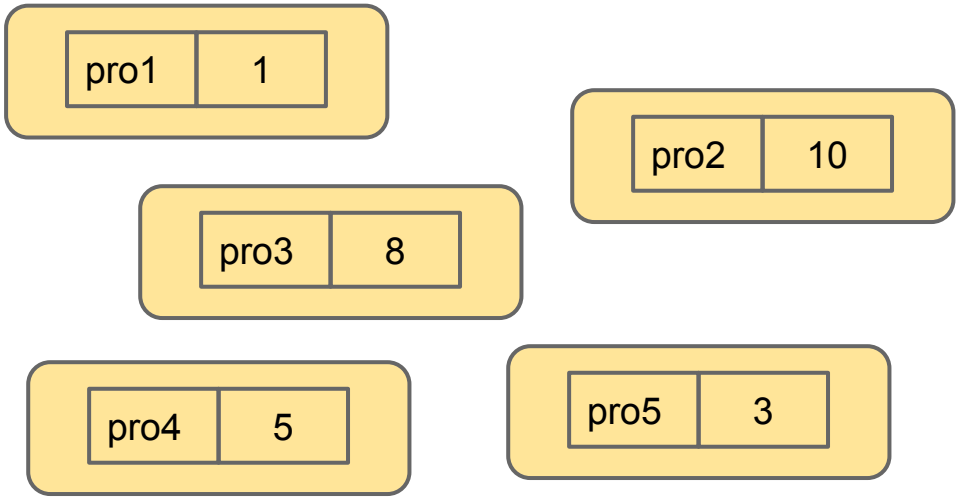
Algoritmos y Estructuras de Datos 2015

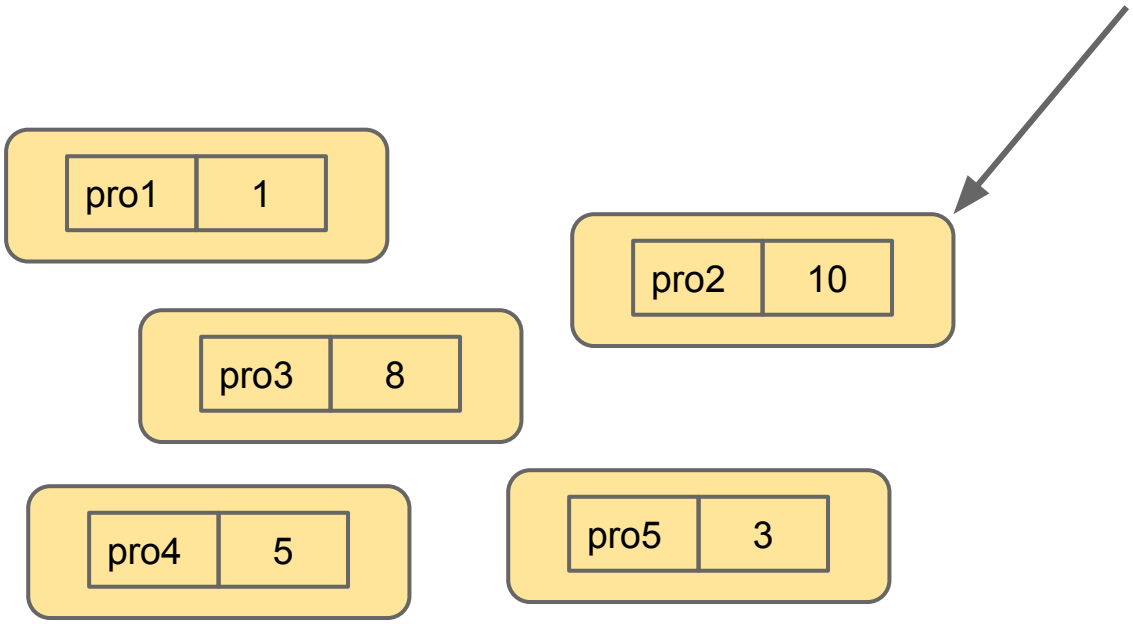
Heap

Enunciado

Un sistema operativo administra sus procesos por prioridades, es decir, de un conjunto de procesos selecciona el de mayor prioridad para ejecutar. A continuación se muestran procesos y su prioridad. ¿Cuál es el de mayor prioridad?



Enunciado



Enunciado

Defina una clase ConjuntoDeProcesos e implemente los métodos:

void agregarProceso (String id, int prioridad)

String retornarMayorPrioridad () (retorna y elimina)

void incrementarPrioridad (String id)

¿Qué estructura utilizar?

- lista?
- array?
- árbol general?
- árbol binario?
- árbol binario de búsqueda?
- avl?
- heap?

¿Qué estructura utilizar?

- lista?

Podría ser… el problema consiste en tener un conjunto de elementos y buscar el mayor.

- array?

Idem anterior.

- árbol general?

No tiene mucho sentido utilizar una estructura que enfatiza en una organización jerárquica

¿Qué estructura utilizar?

- árbol binario?

Idem árbol general.

- árbol binario de búsqueda?

Podría permitir recuperar el elemento mayor de forma eficiente.

- AVL?

Idem árbol binario de búsqueda

- Heap?

Idem árbol binario de búsqueda

Enunciado

Tenga en cuenta que las operaciones deben ser lo más eficientes posibles.

void agregarProceso (String id, int prioridad)

String retornarMayorPrioridad ()

void incrementarPrioridad (String id)

¿Que estructuras permiten la mejor eficiencia?

- agregarProceso (String id, int prioridad)

Lista, Array, ABB, AVL, Heap?

- String retornarMayorPrioridad ()

Lista, Array, ABB, AVL, Heap?

- incrementarPrioridad (String id)

Lista, Array, ABB, AVL, Heap?

¿Que estructuras permiten la mejor eficiencia?

- void agregarProceso (String id, int prioridad)

Lista, Array -> si no nos preocupamos por mantener el orden, O(1)

Lista, Array -> si nos preocupamos por mantener el orden, O(n)

ABB -> O(n)

AVL -> O(Log(n))

Heap -> O(Log(n))

¿Que estructuras permiten la mejor eficiencia?

- String retornarMayorPrioridad ()

Lista, Array -> si están ordenados O(1)

Lista, Array -> si no están ordenados O(n)

ABB -> puede degenerar en una lista O(n)

AVL -> O(Log(n))

Heap -> O(Log(n))

¿Que estructuras permiten la mejor eficiencia?

- incrementarPrioridad (String id)

Lista, Array-> hay que buscar y actualizar O(n)

ABB -> hay que buscar y reordenar O(n). Se ordena por prioridad y se busca por id.

AVL-> Idem ABB

Heap -> Idem ABB

¿Que estructuras permiten la mejor eficiencia?

	Lista desorden	Lista ordenada	Array desorden	Array ordenado	ABB	AVL	Heap
agregar	O(1)	O(n)	O(1)	O(n)	O(n)	O(Log(n))	O(Log(n))
retornar	O(n)	O(1)	O(n)	O(1)	O(n)	O(Log(n))	O(Log(n))
increment	O(n)	O(n)	O(n)	O(n)	O(n)	O(n)	O(n)

¿Qué estructura utilizamos?

- ¿Es posible que haya procesos con prioridad repetida?
- ¿Qué sucede con claves repetidas en un AVL?
- ¿Qué sucede con prioridades repetidas en una HEAP?
- ¿Qué estructura en promedio posee mejor eficiencia para el problema planteado?

¿Qué estructura utilizamos?

- ¿Es posible que haya procesos con prioridad repetida? -> **Si**
- ¿Qué sucede con claves repetidas en un AVL? **-> no funciona**
- ¿Qué sucede con prioridades repetidas en una HEAP? -> **funciona (aunque no necesariamente respeta el orden de ingreso)**
- ¿Qué estructura en promedio posee mejor eficiencia para el problema planteado? -> **una Heap, al brindar un orden parcial (y no completo)**

¿Cómo utilizamos la Heap? ¿Subclasificamos o componemos?

¿ConjuntoDeProcesos debe ser **subclase** de Heap?

- Beneficios:

Se reutiliza la funcionalidad de Heap

- Desventajas:

Conceptualmente no cumple la relación “es-un”. Es decir, al cambiar la restricción de eficiencia de las operaciones, puede cambiar la implementación y por consiguiente la subclasificación.

¿Cómo utilizamos la Heap? ¿Subclasificamos o componemos?

¿ConjuntoDeProcesos debe usar Heap por **composicion**?

- Beneficios:

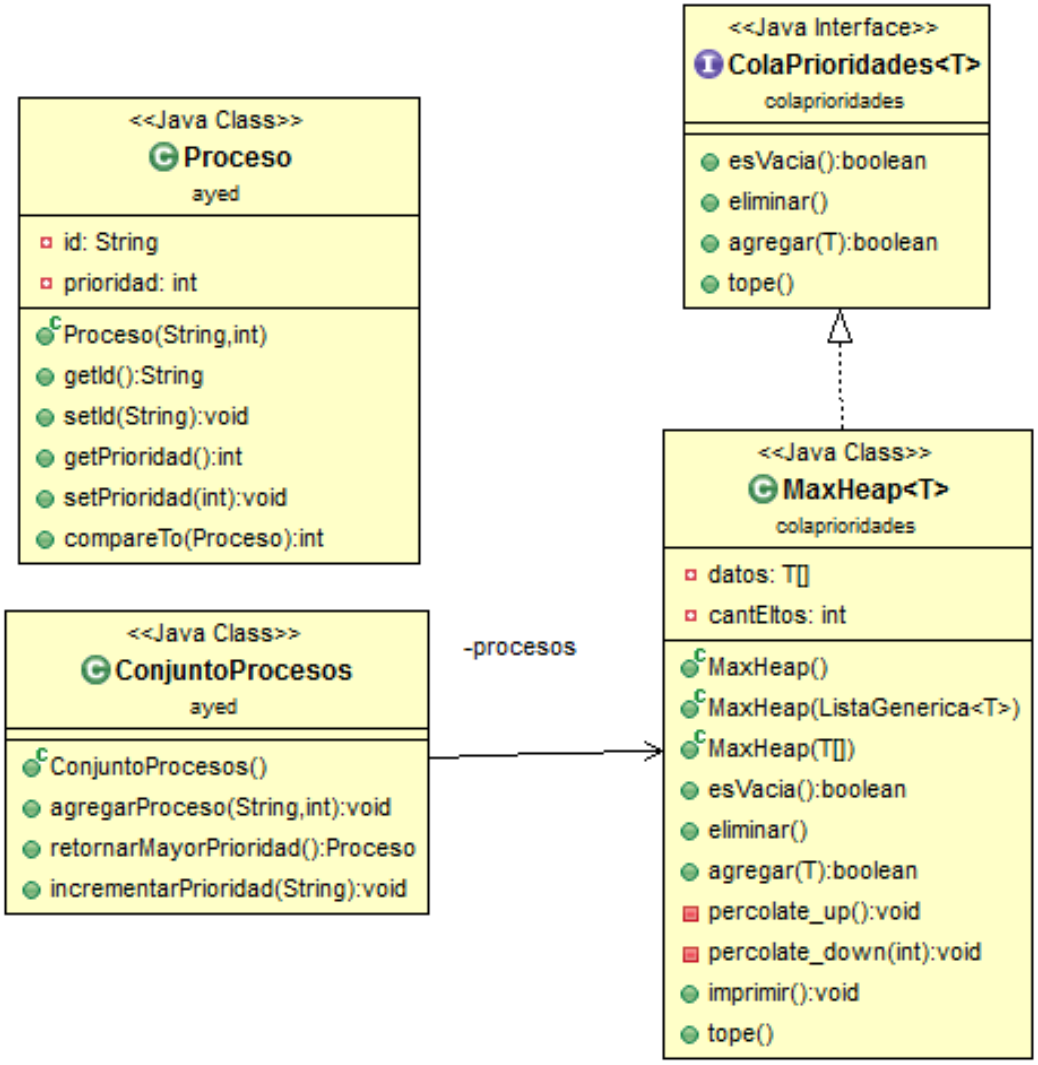
Se reutiliza la funcionalidad de Heap

Al cambiar la restricción de eficiencia de las operaciones, se puede cambiar la Heap por otra estructura

¿Cómo utilizamos la Heap?

- Por composición

Diseño de la solución (por composición)



Implementación de la solución

```
package ayed;
import colaprioridades.*;
public class ConjuntoProcesos {
        private MaxHeap<Proceso> procesos = new MaxHeap<Proceso>();

        public void agregarProceso(String id, int prioridad) {
                procesos.agregar(new Proceso(id, prioridad));}

        public Proceso retornarMayorPrioridad() {
                Proceso procesoMayorPrioridad = procesos.eliminar();
                return procesoMayorPrioridad;}

        public void incrementarPrioridad(String id) {
                // opcion 1: extrae de la Heap los procesos de a uno, hasta encontrar el proceso buscado,
modificar la prioridad y finalmente volver a insertar todos los procesos extraídos.
                // opción 2: la Heap exporta alguna operacion increaseKey (id), la cual busca en la heap el
        proceso de key=id y una vez encontrado hace un percolate_Up()
        }
}
```