Examen de Algorítmica y Complejidad (Plan 2014) — **8 de noviembre de 2019**

Nº matrícula:_____ **Nombre:** _____

Apellidos: _____

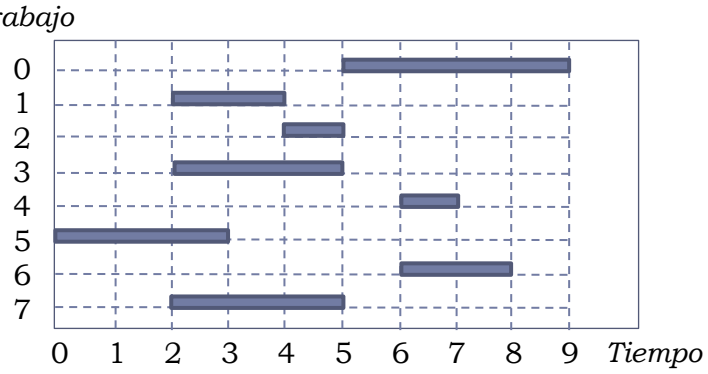
Problema. Se dispone de una lista de trabajos que tienen asociado un instante de comienzo ci y un instante de finalización fi, donde ci<fi, de forma que cada trabajo debe realizarse durante el periodo [ci,fi). Durante la realización de un trabajo es necesario hacer uso de un recurso único de manera exclusiva, por lo que dos trabajos i, j son compatibles si los intervalos [ci,fi) y [cj,fj) no se superponen. Se quiere obtener el conjunto de trabajos mutuamente compatibles que maximice el tiempo que está en uso el recurso. Ejemplo:

Comienzo

0	1	2	3	4	5	6	7
5	2	4	2	6	0	6	2

Fin

0	1	2	3	4	5	6	7
9	4	5	5	7	3	8	5



Los trabajos compatibles que maximizan el uso del recurso son el 0, el 2 y el 5, que en total hacen uso del recurso durante 8 unidades de tiempo. **Implementar un algoritmo en Java**, basado en el **esquema de Selección Optima** que ofrezca esta funcionalidad. El algoritmo a implementar deberá tener la siguiente cabecera:

boolean[] maxUsoRecurso(**int**[] comienzo, **int**[] fin)

donde los parámetros comienzo y fin contienen el instante de comienzo y finalización de todos los trabajos. El método deberá devolver un vector de boolean, con valor true en las posiciones de los trabajos seleccionados por el algoritmo. Se podrán implementar todos los métodos privados y las clases Java que se consideren necesarios.

```
boolean[] maxUsoRecurso(int[] comienzo, int[] fin){
    boolean[] seleccionados = new boolean[comienzo.length];
    int usoRecurso=0;
    boolean[] mejorSeleccionados = new boolean[comienzo.length];
    Entero mejorUsoRecurso = new Entero(0);
    for (int i=0; i<seleccionados.length; i++) seleccionados[i]=false;

    maxUsoRecursoAux(comienzo, fin, 0, seleccionados, usoRecurso,
                     mejorSeleccionados, mejorUsoRecurso);

    return mejorSeleccionados;
}
```

```
void maxUsoRecursoAux(int[] comienzo, int[] fin, int tarea,
                      boolean[] seleccionados, int usoRecurso,
                      boolean[] mejorSeleccionados, Entero mejorUsoRecurso){
    if (tarea==comienzo.length){
        if (usoRecurso > mejorUsoRecurso.getValor()){
            mejorUsoRecurso.setValor(usoRecurso);
            for (int i=0; i< seleccionados.length; i++)
                mejorSeleccionados[i] = seleccionados[i];
        }
    }
    else{
        for (int c=0; c<2; c++){
            // c=0 no se selecciona la tarea "tarea"
            // c=1 sí se selecciona la tarea "tarea"
            if ((c==0) || compatible(comienzo, fin, seleccionados, tarea)){
                seleccionados[tarea] = (c==1);
                usoRecurso = usoRecurso + (fin[tarea] - comienzo[tarea]) * c;
                tarea++;
                maxUsoRecursoAux(comienzo, fin, tarea, seleccionados, usoRecurso,
                                 mejorSeleccionados, mejorUsoRecurso);
                tarea--;
                usoRecurso = usoRecurso - (fin[tarea] - comienzo[tarea]) * c;
                seleccionados[tarea]=false;
            }
        }
    }
}


boolean compatible(int[] comienzo, int[] fin, boolean[] seleccionados, int tarea){
    boolean ok=true;
    int i=0;
    while (ok && (i<tarea)){
        if (seleccionados[i])
            ok = ((comienzo[i]>= fin[tarea]) || (comienzo[tarea]>=fin[i]));
        i++;
    }
    return ok;
}
```