

Algoritmos y Estructuras de Datos II

Práctica 4 – Diseño: Elección de estructuras de datos

Notas preliminares

- En esta práctica se utilizan los términos abreviados *Rep* y *Abs* para referirse al invariante de representación y a la función de abstracción, respectivamente.
- Por “diseñar” se entiende el hecho de elegir una estructura de representación adecuada, esquematizar *Rep*, *Abs* y (en pseudocódigo) los algoritmos más importantes y justificar que se cumplen los requerimientos pedidos.

Ejercicio 1

Considere el TAD `MULTICONJUNTO(NAT)`.

1. Diseñelo utilizando listas enlazadas de naturales. Calcule el orden de complejidad (de peor caso) de las operaciones de inserción, borrado y cálculo de la cantidad de repeticiones de un elemento.¹
2. Piense en otras estructuras –más allá de las listas– y estime los órdenes de complejidad de peor caso que tendrían las operaciones mencionadas.

Ejercicio 2 ★

Escriba algoritmos para las operaciones \cap y \cup , considerando las siguientes representaciones para conjuntos:

1. Sobre listas enlazadas (ordenadas y no ordenadas).
2. Sobre árboles binarios de búsqueda.
3. Sobre *heaps*.

En cada inciso, calcule los órdenes de complejidad de peor caso de las operaciones.

Ejercicio 3

Confeccione una tabla comparativa de las complejidades de peor caso de las operaciones de pertenencia, inserción, borrado, búsqueda del mínimo y borrado del mínimo para conjuntos de naturales sobre las siguientes estructuras:

- | | |
|--------------------------------|---|
| 1. Lista enlazada. | 4. Árbol AVL. |
| 2. Lista enlazada ordenada. | 5. <i>heap</i> . |
| 3. Árbol binarios de búsqueda. | 6. <i>trie</i> (usar la expresión binaria del natural). |

Ejercicio 4 ★

El TAD `MATRIZ INFINITA DE BOOLEANOS` tiene las siguientes operaciones:

- *Crear*, que crea una matriz donde todos los valores son falsos.
- *Asignar*, que toma una matriz, dos naturales (fila y columna) y un booleano, y asigna a este último en esa coordenada. (Como la matriz es infinita, no hay restricciones sobre la magnitud de fila y columna.)
- *Ver*, que dadas una matriz, una fila y una columna devuelve el valor de esa coordenada. (Idem.)
- *Complementar*, que invierte todos los valores de la matriz.
- *Intersecar*, que toma dos matrices y realiza la conjunción coordenada a coordenada.

¹Note que cuando se expresa el orden de complejidad, se hace en base a cierto n que representa el “tamaño” de la estructura en algún sentido. ¿Qué representa n en los incisos a) y b)?

Diseñe este TAD de modo que las operaciones *Crear*, *Ver* y *Complementar* tomen $O(1)$ tiempo en peor caso.

Ejercicio 5 ★

Una matriz finita posee las siguientes operaciones:

- *Crear*, con la cantidad de filas y columnas que albergará la matriz.
- *Definir*, que permite definir el valor para una posición válida.
- *#Filas*, que retorna la cantidad de filas de la matriz.
- *#Columnas*, que retorna la cantidad de columnas de la matriz.
- *Obtener*, que devuelve el valor de una posición válida de la matriz (si nunca se definió la matriz en la posición solicitada devuelve cero).
- *SumarMatrices*, que permite sumar dos matrices de iguales dimensiones.

Diseñe un módulo para el TAD MATRIZ FINITA de modo tal que dadas dos matrices finitas A y B ,

- * *Definir* y *Obtener* aplicadas a A se realicen cada una en $\Theta(n)$ en peor caso, y
- * *SumarMatrices* aplicada a A y B se realice en $\Theta(n + m)$ en peor caso,

donde n y m son la cantidad de elementos **no nulos** de A y B , respectivamente.

Ejercicio 6

Considere el TAD DICCIONARIO CON HISTORIA, cuya especificación es la siguiente:

TAD DICCIONARIO CON HISTORIA(CLAVE, SIGNIFICADO)

parámetros formales

géneros κ, σ

$\bullet = \bullet : \kappa \times \kappa \rightarrow \text{bool}$

(relación de equivalencia)

igualdad observacional

$$(\forall d, d' : \text{diccHist}(\kappa, \sigma)) \left(d =_{\text{obs}} d' \iff \left((\forall k : \kappa) (\text{Definido?}(k, d) =_{\text{obs}} \text{Definido?}(k, d') \wedge_{\text{L}} \right. \right. \\ \left. \left. (\text{Definido?}(k, d) \Rightarrow_{\text{L}} | \right. \right. \\ \left. \left. \text{Significado}(k, d) =_{\text{obs}} \text{Significado}(k, d') \wedge \right. \right. \\ \left. \left. \text{BorrarSignificado}(k, d) =_{\text{obs}} \text{BorrarSignificado}(k, d') \right) \right)$$

observadores básicos

Definido? : $\kappa \times \text{diccHist} \rightarrow \text{bool}$

Significado : $\kappa \times \text{diccHist } dh \rightarrow \sigma$

{*Definido?*(c, dh)}

BorrarSignificado : $\kappa \times \text{diccHist } dh \rightarrow \text{diccHist}$

{*Definido?*(c, dh)}

generadores

Vacío : $\rightarrow \text{diccHist}$

Definir : $\kappa \times \sigma \times \text{diccHist} \rightarrow \text{diccHist}$

otras operaciones

CantSignificados : $\kappa \times \text{diccHist } dh \rightarrow \text{nat}$

{*Definido?*(c, dh)}

axiomas $\forall dh : \text{diccHist} \forall c, c' : \kappa \forall s : \sigma$

Definido?($c, \text{Vacío}$) $\equiv \text{false}$

Definido?($c, \text{Definir}(c', s, dh)$) $\equiv (c = c') \vee \text{Definido?}(c, dh)$

Significado($c, \text{Definir}(c', s, dh)$) $\equiv \text{if } c = c' \text{ then } s \text{ else } \text{Significado}(c, dh) \text{ fi}$

BorrarSignificado($c, \text{Definir}(c', s, dh)$) $\equiv \text{if } c = c' \text{ then}$

dh

else

Definir($c', s, \text{BorrarSignificado}(c, dh)$)

fi

CantSignificados($c, \text{Definir}(c', s, dh)$) $\equiv \text{if } c = c' \text{ then } 1 \text{ else } 0 \text{ fi} +$

if Definido?(c, dh) **then** *CantSignificados*(c, dh) **else** 0 **fi**

Fin TAD

Se debe diseñar este TAD respetando los siguientes órdenes de ejecución en el peor caso:

- | | | |
|--------------------|----------------------|----------------------------|
| ■ Definido? $O(n)$ | ■ Significado $O(n)$ | ■ BorrarSignificado $O(n)$ |
| ■ Vacío $O(1)$ | ■ Definir $O(n)$ | ■ CantSignificados $O(n)$ |

donde n es la cantidad de claves que están definidas en el diccionario.

Ejercicio 7

1. Diseñe el TAD $\text{ÁRBOL BINARIO}(\alpha)$, teniendo en cuenta que las operaciones *nil*, *bin*, *nil?*, *izq*, *der* y *raíz* deben tener orden de complejidad $O(1)$ en peor caso.
2. Especifique la precondition, postcondición y el aliasing de las operaciones mencionadas.
3. Dar algoritmos para resolver las siguientes operaciones, y calcular su orden de ejecución en peor caso:
 - a) NivelCompleto? : $\text{nat } n \times \text{ab}(\alpha) a \longrightarrow \text{bool} \quad \{n \leq h(a)\}$
 - b) Completo? : $\text{ab}(\alpha) \longrightarrow \text{bool}$

Ejercicio 8

Diseñe el TAD COLA DE PRIORIDAD, escribiendo los algoritmos e indicando sus órdenes de complejidad de peor caso sobre las siguientes estructuras:

1. Árbol binario con invariante de *heap*.
2. Arreglo con invariante de *heap*.

Ejercicio 9 ★

Diseñar un conjunto de naturales que satisfaga los siguientes requerimientos de complejidad temporal:

- Ag debe costar tiempo $O(1)$ si el elemento a agregar es menor que el mínimo elemento del conjunto hasta el momento o mayor que el máximo, y tiempo lineal en la cantidad de elementos del conjunto en cualquier otro caso.
 - \cup y \cap deben tomar tiempo $O(1)$ si cada elemento de uno de los conjuntos pasados como parámetro es mayor que todos los del otro, y tiempo $O(n + m)$ en cualquier otro caso, siendo n y m las cardinalidades de los dos conjuntos pasados como parámetros.
1. Elegir una estructura adecuada, justificando la decisión, y escribir los algoritmos para las tres operaciones anteriores. Documentar claramente los aspectos de aliasing.
 2. ¿Puede usarse este mismo diseño para conjuntos que no sean de naturales? ¿Qué condición debe cumplir el tipo al que pertenecen los elementos para poder usar este módulo?

Ejercicio 10 ★

Diseñar las relaciones binarias entre números naturales. Para una relación \mathcal{R} , se quiere preguntar:

- Dado un número a , cuáles son los números b tales que $a\mathcal{R}b$.
- Dado un número a , cuáles son los números b tales que $b\mathcal{R}a$.

Las relaciones se construyen por extensión, es decir, agregando y borrando parejas de la relación.

1. Suponiendo que las relaciones son entre números de 0 a 100, elegir estructuras de datos adecuadas. Justificar. Dar el invariante de representación y la función de abstracción de las estructuras. Implementar las operaciones sobre las estructuras elegidas. Indicar el orden de complejidad de las operaciones.
2. Elegir las estructuras de datos adecuadas si el rango sobre el cual se una relación será fijado al momento de la creación de la misma. Justificar.

- Elegir las estructuras de datos adecuadas si las relaciones son entre naturales cualesquiera (i.e., no hay un rango prefijado), de manera que ninguna operación tenga un costo demasiado alto. Justificar.

Ejercicio 11 ★

Se desea diseñar un sistema de estadísticas para la cantidad de personas que ingresan a un banco. Al final del día, un empleado del banco ingresa en el sistema el total de ingresantes para ese día. Se desea saber, en cualquier intervalo de días, la cantidad total de personas que ingresaron al banco. La siguiente es una especificación del problema.

TAD INGRESOSALBANCO

observadores básicos

$\text{totDias} : \text{iab} \rightarrow \text{nat}$

$\text{cantPersonas} : \text{iab } i \times \text{nat } d \times \text{nat } h \rightarrow \text{nat} \quad \{1 \leq d \wedge d \leq h \wedge h \leq \text{totDias}(i)\}$

generadores

$\text{Comenzar} : \rightarrow \text{iab}$

$\text{TerminaDia} : \text{iab} \times \text{nat} \rightarrow \text{iab}$

axiomas ...

$\text{totDias}(\text{Comenzar}) \equiv 0$

$\text{totDias}(\text{TerminaDia}(i, n)) \equiv 1 + \text{totDias}(i)$

$\text{cantPersonas}(\text{TerminaDia}(i, n), d, h) \equiv \text{if } \text{totDias}(i) < h \text{ then } n \text{ else } 0 \text{ fi} + \text{if } \text{totDias}(i) < d \text{ then } 0 \text{ else } \text{cantPersonas}(i, d, \text{mín}(h, \text{totDias}(i))) \text{ fi}$

Fin TAD

- Dar una estructura de representación que permita que la función cantPersonas tome $O(1)$.
- Calcular cuánta memoria usa la estructura, en función de la cantidad de días que pasaron n .
- Si el cálculo del punto anterior fue una función que no es $O(n)$, piense otra estructura que permita resolver el problema utilizando $O(n)$ memoria.
- Agregue al diseño del punto anterior una operación *mediana* que devuelva el último (mayor) día d tal que $\text{cantPersonas}(i, 1, d) \leq \text{cantPersonas}(i, d+1, \text{totDias}(i))$, restringiendo la operación a los casos donde dicho día existe. Si este ítem no sale, se recomienda dejarlo pendiente hasta avanzada la práctica de dividir y conquistar. No olvidar retomarlo luego.

Ejercicio 12

El servicio meteorológico ha decidido generar un sistema que registre el nivel de smog instante a instante, durante un año. Cuenta con diversos empleados que van ingresando las distintas mediciones. Se desea que el ingreso de las mediciones sea muy eficiente. También se desea obtener el promedio mensual. Además, se está estudiando premiar al empleado más productivo de cada mes (el que más mediciones registre), por lo que es necesario conocerlo. La especificación es la siguiente:

TAD SERVICIO METEOROLÓGICO

generadores

$\text{CrearSM} : \text{conj}(\text{empleado}) \times \text{timestamp} \rightarrow \text{sm}$

$\text{IngresarMedición} : \text{sm } s \times \text{empleado } e \times \text{timestamp } t \times \text{nat } n \rightarrow \text{sm} \quad \{e \in \text{Empleados}(s) \wedge \text{Mes}(t) = \text{MesActual}(s) \wedge \neg \text{HayValor?}(s, t)\}$

$\text{CerrarMes} : \text{sm} \rightarrow \text{sm}$

observadores básicos

$\text{MesActual} : \text{sm} \rightarrow \text{mes}$

$\text{Empleados} : \text{sm} \rightarrow \text{conj}(\text{empleado})$

$\text{HayValor?} : \text{sm} \times \text{timestamp} \rightarrow \text{bool}$

Valor : $\text{sm } s \times \text{timestamp } t \rightarrow \text{nat}$ $\{\text{HayValor?}(s, t)\}$
 #MedicionesDelEmpleado : $\text{sm } s \times \text{empleado } e \times \text{mes } m \rightarrow \text{nat}$ $\{e \in \text{Empleados}(s)\}$
otras operaciones
 PromedioMensual : $\text{sm } s \times \text{mes } m \rightarrow \text{nat}$ $\{\sum_{e \in \text{Empleados}(s)} \# \text{MedicionesDelEmpleado}(s, e, m) > 0\}$
 MejorEmpleado : $\text{sm } s \times \text{mes } m \rightarrow \text{empleado}$ $\{(\exists e: \text{empleado})(e \in \text{Empleados}(s) \wedge \# \text{MedicionesDelEmpleado}(s, e, m) > 0)\}$
axiomas ...

Fin TAD

EMPLEADO ES STRING[20], MES ES NAT (entre 1 y 12) y TIMESTAMP ES NAT (con precisión arbitraria y proyectores para ver día, mes, etc.).

Diseñar un sistema eficiente, que responda a las siguientes complejidades temporales en peor caso:

- IngresarMedición: $O(1)$.
- Valor: $O(n \log(m' + n))$, siendo n la cantidad de mediciones ingresadas desde la última invocación a la función y m' la máxima cantidad de mediciones correspondientes a una misma hora (es decir, las 9 del día 1, las 10 del día 1, ..., las 9 del día 2, etc.),
- MejorEmpleado y PromedioMensual: $O(1)$.

Pista: no se olvide de *CerrarMes*.

Ejercicio 13 ★

Considere el siguiente TAD que describe la estructura de un texto:

TAD TEXTO

observadores básicos
 cantPalabras : $\text{txt} \rightarrow \text{nat}$
 enPos : $\text{txt } t \times \text{nat } n \rightarrow \text{palabra}$ $\{1 \leq n \leq \text{cantPalabras}(t)\}$
generadores
 enBlanco : $\rightarrow \text{txt}$
 agPalabra : $\text{txt} \times \text{palabra} \rightarrow \text{txt}$
otras operaciones
 cambiarPalabra : $\text{txt} \times \text{palabra} \times \text{palabra} \rightarrow \text{txt}$
 posiciones : $\text{txt} \times \text{palabra} \rightarrow \text{conj}(\text{nat})$
 subtexto : $\text{txt } t \times \text{nat } ini \times \text{nat } fin \rightarrow \text{txt}$ $\{1 \leq ini \leq fin \leq \text{cantPalabras}(t)\}$
 masRepetidas : $\text{txt} \rightarrow \text{conj}(\text{palabra})$ $\{\text{cantPalabras}(t) > 0\}$
otras operaciones (no exportadas)
 maxRepeticiones : $\text{txt} \rightarrow \text{nat}$
axiomas

 cantPalabras(enBlanco) $\equiv 0$
 cantPalabras(agPalabra(t, p)) $\equiv 1 + \text{cantPalabras}(t)$

 enPos(agPalabra(t, p), n) \equiv **if** cantPalabras(t) = $n - 1$ **then** p **else** enPos(t, n) **fi**

 cambiarPalabra(enBlanco, p, p') \equiv enBlanco
 cambiarPalabra(agPalabra(t, p''), p, p') \equiv agPalabra(cambiarPalabra(t, p, p'),
if $p = p''$ **then** p' **else** p'' **fi**)

 posiciones(enBlanco, p') $\equiv \emptyset$
 posiciones(agPalabra(t, p), p') \equiv **if** $p = p'$ **then**
Ag(cantPalabras(t)+1, posiciones(t, p')))
else
posiciones(t, p')
fi

```

subtexto( $t, i, f$ )  $\equiv$  if  $i = f$  then
    agPalabra(enPos( $t, i$ ), enBlanco)
else
    agPalabra(enPos( $t, f$ ), subtexto( $t, i, f - 1$ ))
fi

masRepetidas(agPalabra( $t, p$ ))  $\equiv$  if cantPalabras( $t$ ) = 0  $\vee$  #posiciones( $t, p$ ) = maxRepeticiones( $t$ ) then
    Ag( $p, \emptyset$ )
else
    if #posiciones( $t, p$ ) + 1 = maxRepeticiones( $t$ ) then
        Ag( $p, masRepetidas(t)$ )
    else
        masRepetidas( $t$ )
    fi
fi

maxRepeticiones( $t$ )  $\equiv$  if cantPalabras( $t$ ) = 0 then
    0
else
    #posiciones( $t, dameUno(masRepetidas(t))$ )
fi

```

Fin TAD

Se desea diseñar el módulo correspondiente al TAD texto. En particular, asumimos que trabajaremos sólo con textos en español, y por lo tanto podemos dar una cota para la longitud de la palabra más larga que puede aparecer en el texto.

Se requiere que las operaciones que se listan a continuación cumplan con la complejidad temporal indicada:

- *subtexto*(*in inicio: nat, in fin: nat, in t: txt*) \rightarrow *txt*

Devuelve el texto correspondiente al fragmento de t que comienza en la posición *inicio* y finaliza en la posición *fin*.

$O(\text{fin} - \text{inicio})$ en el peor caso

- *cambiarPalabra*(*in anterior: palabra, in nueva: palabra, inout t: text*)

Cambia todas las ocurrencias en el texto de la palabra anterior por la nueva.

$O(k)$ en el peor caso, donde k es la cantidad de veces que se repite la palabra a cambiar.

- *palabrasMasRepetidas* (*in t: txt*) \rightarrow *conj(palabras)* Devuelve el conjunto de palabras que más se repiten en el texto.

$O(1)$ en el peor caso. Puede generar aliasing.

- a) Describir la estructura a utilizar, documentando claramente cómo la misma resuelve el problema y cómo cumple con los requerimientos de eficiencia. El diseño debe incluir sólo la estructura de nivel superior. Para justificar los órdenes de complejidad, describa las estructuras soporte. **Importante:** si alguna de las estructuras utilizadas requiere que sus elementos posean una función especial (por ejemplo, comparación) deberá describirla.
- b) Escribir una versión en lenguaje imperativo del algoritmo *cambiarPalabra*. Justifique la complejidad sobre el código.

Ejercicio 14 ★

Se desea diseñar un sistema para monitorear una planta industrial que cuenta con un conjunto de alarmas asociadas a distintos sensores. Cada sensor está asociado a una única alarma pero cada una de estas puede ser activada por distintos sensores. Una alarma está activa cuando la medición de al menos uno de sus sensores asociados supera un valor umbral definido para ese sensor. Los sensores y las alarmas se identifican con un código alfanumérico (para los sensores puede asumir que este es de longitud acotada, mientras que para las alarmas no)

El siguiente TAD es una especificación para este problema.

TAD PLANTA**observadores básicos**

$\text{esAlarma} : \text{planta} \times \text{alarma} \rightarrow \text{bool}$
 $\text{esSensor} : \text{planta} \times \text{sensor} \rightarrow \text{bool}$
 $\text{alarmaSensor} : \text{planta } p \times \text{sensor } s \rightarrow \text{alarma} \quad \{\text{esSensor}(p,s)\}$
 $\text{umbral} : \text{planta } p \times \text{sensor } s \rightarrow \text{nat} \quad \{\text{esSensor}(p,s)\}$
 $\text{medicion} : \text{planta } p \times \text{sensor } s \rightarrow \text{nat} \quad \{\text{esSensor}(p,s)\}$

generadores

$\text{crear} : \text{conj}(\text{alarma}) \rightarrow \text{planta}$
 $\text{agSensor} : \text{planta } p \times \text{sensor } s \times \text{nat } u \times \text{alarma } a \rightarrow \text{planta} \quad \{\neg \text{esSensor}(p,s) \wedge \text{esAlarma}(p,a) \wedge u > 0\}$
 $\text{nuevaMedicion} : \text{planta } p \times \text{sensor } s \times \text{nat} \rightarrow \text{planta} \quad \{\text{esSensor}(p,s)\}$

otras operaciones

$\text{encendidosPorAlarma} : \text{planta } p \times \text{alarma } a \rightarrow \text{conj}(\text{sensor}) \quad \{\text{esAlarma}(p,a)\}$
 $\text{encendida} : \text{planta } p \times \text{alarma } a \rightarrow \text{bool} \quad \{\text{esAlarma}(p,a)\}$

axiomas

$\text{esAlarma}(\text{crear}(c), a) \equiv a \in c$
 $\text{esAlarma}(\text{agSensor}(p, s, u, a'), a) \equiv \text{esAlarma}(p, a)$
 $\text{esAlarma}(\text{nuevaMedicion}(p, s, n), a) \equiv \text{esAlarma}(p, a)$

 $\text{esSensor}(\text{crear}(c), s) \equiv \text{false}$
 $\text{esSensor}(\text{agSensor}(p, s', u, a), s) \equiv s = s' \vee \text{esSensor}(p, s)$
 $\text{esSensor}(\text{nuevaMedicion}(p, s', n), s) \equiv \text{esSensor}(p, s)$

 $\text{alarmaSensor}(\text{agSensor}(p, s', u, a), s) \equiv \text{if } s = s' \text{ then } a \text{ else } \text{alarmaSensor}(p, a) \text{ fi}$
 $\text{alarmaSensor}(\text{nuevaMedicion}(p, s', n), s) \equiv \text{alarmaSensor}(p, s)$

 $\text{umbral}(\text{agSensor}(p, s', u, a), s) \equiv \text{if } s = s' \text{ then } u \text{ else } \text{umbral}(p, a) \text{ fi}$
 $\text{umbral}(\text{nuevaMedicion}(p, s', n), s) \equiv \text{umbral}(p, s)$

 $\text{medicion}(\text{agSensor}(p, s', u, a), s) \equiv \text{if } s = s' \text{ then } 0 \text{ else } \text{medicion}(p, a) \text{ fi}$
 $\text{medicion}(\text{nuevaMedicion}(p, s', n), s) \equiv \text{if } s = s' \text{ then } n \text{ else } \text{medicion}(p, a) \text{ fi}$

 $\text{encendidosPorAlarma}(\text{crear}(c), a) \equiv \emptyset$
 $\text{encendidosPorAlarma}(\text{agSensor}(p, s, u, a'), a) \equiv \text{encendidosPorAlarma}(p, a)$
 $\text{encendidosPorAlarma}(\text{nuevaMedicion}(p, s, n), a) \equiv \text{if } \text{alarmaSensor}(p, s) = a \wedge n > \text{umbral}(p, s) \text{ then}$
 $\quad \text{Ag}(s, \text{encendidosPorAlarma}(p, a))$
 $\quad \text{else}$
 $\quad \text{encendidosPorAlarma}(p, a) - \{s\}$
 $\quad \text{fi}$

 $\text{encendida}(p, a) \equiv \#\text{encendidosPorAlarma}(p, a) > 0$

Fin TAD

Se desea diseñar el sistema propuesto, teniendo en cuenta que la operación *nuevaMedicion* debe realizarse con complejidad temporal $O(1)$ y *encendida* con complejidad temporal $O(\ell a)$ en el peor caso, donde a es la cantidad de alarmas de la planta y ℓ es la longitud del identificador de alarma más largo.

Se pide:

- Describir la estructura a utilizar, documentando claramente cómo la estructura resuelve el problema y cómo cumple con los requerimientos de eficiencia. El diseño debe incluir sólo la estructura de nivel superior. (De ser necesario para justificar los órdenes de complejidad, describa las estructuras soporte.)
- Escribir una versión en lenguaje imperativo del algoritmo *nuevaMedición*, indicando la complejidad de cada uno de los pasos. Tener en cuenta que esta operación puede activar una alarma (cuando la medición supera el umbral) o desactivarla (cuando la medición pasa a ser menor que el umbral y no hay otros sensores activándola).

$$(\forall s, s' : \text{sueco}) \left(s =_{\text{obs}} s' \iff \left((\forall d:\text{nat})(\forall h:\text{nat}) \begin{array}{l} \#Intervalo(s, d, h) =_{\text{obs}} \#Intervalo(s', d, h) \\ plataIntervalo(s, d, h) =_{\text{obs}} plataIntervalo(s', d, h) \end{array} \wedge \right) \right)$$

observadores básicos

$\#Intervalo : \text{sueco} \times \text{nat} \times \text{nat} \longrightarrow \text{nat}$
 $plataIntervalo : \text{sueco} \times \text{nat} \times \text{nat} \longrightarrow \text{nat}$

generadores

$iniciarDia : \longrightarrow \text{sueco}$
 $agregarBoleto : \text{sueco} \times \text{nat} \times \text{nat } p \longrightarrow \text{sueco} \quad \{0 < p\}$

axiomas

$\#Intervalo(iniciarDia, d, h) \equiv 0$
 $\#Intervalo(agregarBoleto(s, t, p), d, h) \equiv \#Intervalo(s, d, h) + \text{if } d \leq t \wedge t < h \text{ then } 1 \text{ else } 0 \text{ fi}$
 $plataIntervalo(iniciarDia, d, h) \equiv 0$
 $plataIntervalo(agregarBoleto(s, t, p), d, h) \equiv plataIntervalo(s, d, h) + \text{if } d \leq t \wedge t < h \text{ then } p \text{ else } 0 \text{ fi}$

Fin TAD

Se pide dar una estructura de representación que permita realizar la operación *agregarBoleto* en $O(n)$, y *#Intervalo* en $O(\log n)$, donde n es la cantidad de boletos agregados hasta ese momento.

- Describir la estructura a utilizar.
- Para ambas operaciones, explicar como se usa la estructura de manera de cumplir con la complejidad pedida (no hace falta dar pseudocódigo de cada operación, pero se puede darlo si resulta conveniente para la explicación).
- Agregar lo que sea necesario para proveer la operación *plataIntervalo* también en $O(\log n)$. Describa las modificaciones a la estructura, a las operaciones del punto anterior que precisen modificaciones, y la forma de implementar *plataIntervalo*.