

Disclaimer: estas son las notas hechas en la clase TM, puede ser que alguna cosa no se entienda si no participaron de la clase

```

proc nuevoSistema() : Sistema  $O(1)$ 
proc registrarMateria(inout s : Sistema, in m : materia)  $O(\log m)$ 
proc registrarNota(inout s : Sistema, in m : materia, in a : alumno, in n : nota)  $O(\log n + \log m)$ 
proc notaDeAlumno(in s : Sistema, in a : alumno, m : materia) : nota  $O(\log n + \log m)$ 
proc cantAlumnosConNota(in s : Sistema, in m : materia, n : nota) :  $\mathbb{Z}$   $O(\log m)$ 
proc cantAlumnosAprobados(in s : Sistema, in m : materia) :  $\mathbb{Z}$   $O(\log m)$ 

```

$m = \text{cant materias}$, $n = \text{cant alumnos}$

① estructura

var $\Rightarrow dm = \text{Dicc Log} \langle \text{materia}, \text{dicc Log} \langle \text{alumno}, \text{nota} \rangle \rangle$; cumple las comp pedidas?

nuevo Sistema $\leadsto dm = \text{diccionarioVacio}(\text{mat}, \text{diccLog} \langle \text{alum}, \text{nota} \rangle)$
 $O(1)$ $O(1)$

Registrar Materia: insertar m : materia en dm : diccionarioLog $\leadsto O(\log m)$ ✓

Registrar Nota:

buscar materia + buscar (si a : alumno esta) en da : dicLog
 insertar (si a no está)
 $\leadsto O(\log m) + O(\log n)$ ✓

Nota De Alumno:

buscar m en dm + buscar a en da $\leadsto \log(m) + \log(n)$ ✓

Cant Alumnos Con Nota

buscar m en dm + para cada a ver la nota (recorrer todo da)
 $O(\log m) + O(n) = \max\{O(\log m), O(n)\} = O(n)$ X

asumiendo que hay más alumnos que materias ($m \leq n$)

venos que con solo dm no alcanza, proponemos también:

var dn : diccLog $\langle \text{materias}, \text{Notas: array}[11] \rangle$

en cada pos tengo la cant de alumes con esa nota

cant Alumnos Con Nota: buscar m materia + ver una pos de un array
 $\leadsto O(\log m) + O(1) = O(\log m)$ ✓

cant Alumnos Aprobados buscar m materia + contar la pos 7, 8, 9 y 10 del array
 $\leadsto O(\log m) + O(1) = O(\log m)$ ✓

veamos si las anteriores siguen OK:

(anterior)

RegistrarMateria en $O(\log m)$ \rightarrow insertar m materia en cm $\rightarrow O(\log m) + O(\log m)$ ✓

RegistrarNota en $O(\log n + \log m)$ \rightarrow buscar m en cm + modificar nota

NotaDeAlumno en $O(\log n + \log m)$ $O(\log n) + O(1) + O(\log n + \log m)$ ✓

Modulo sistema implementa (TAD) Sistema
 var : $dm = \text{diccLog}(\text{materia}, \text{diccLog}(\text{alumno}, \text{nota}))$
 $dn = \text{diccLog}(\text{materia}, \text{array}[11]:\mathbb{Z})$

3 y 4

→

Proc nuevo Sistema() $O(1)$ tipos

$\left[\begin{array}{l} dm = \text{new diccLog} < \text{materia}, \text{diccLog}(\text{alumno}, \text{nota}) > \quad O(1) \\ dn = \text{new diccLog} < \text{materia}, \text{array}[11] > \quad O(1) \end{array} \right.$

$\left[\begin{array}{l} dm = \text{diccionarioVacio}() \quad O(1) \\ dn = \text{diccionarioVacio}() \quad O(1) \end{array} \right.$

Proc registrar Materia (inout s: sistema, in m: materia)

→ s.dm. definir (m, new diccLog (alumno, nota)) $O(\log m) + o(1)$

→ s.dn. definir (m, new array [11]) $O(\log m) + o(1)$

$O(\log m) + o(1) + O(\log m) + o(1) =$

$= \max \{ O(\log m), o(1), O(\log m), o(1) \} = O(\log m)$

Proc registrar Nota (inout s: sistema, in m: materia, in n: nota, in a: alumno)

s.dm. $\overset{O(\log m)}{\text{obtener}(m)}$. $\overset{O(\log n)}{\text{definir}(a, n)}$ $O(\log m) + O(\log n) = O(\log m + \log n)$

aux = s.dn. obtener(m) $O(\log m) + o(1)$

aux[n] = aux[n] + 1 $O(1)$

s.dn. definir (m, aux) $O(\log m)$

$\max \{ O(\log m + \log n), O(\log m) + o(1), o(1), O(\log m) \} = O(\log m + \log n)$

Proc NotaAlumno (in s: sistema, in m: materia, in a: alumno): \mathbb{Z}

int res = s.dm. obtener(m). obtener(a) $O(\log m + \log n)$

return res $O(1)$

$\max \{ o(1), O(\log m + \log n) \} = O(\log m + \log n)$

Proc cant AlumnosConNota (in s: sistema, in m: materia, in n: nota): \mathbb{Z}

int res = s.dn. obtener (m) [n] $O(\log n) + o(1)$

return res $O(1)$

$\max \{ o(1), O(\log n) \} = O(\log n)$

Proc CantAlumnosAprobados (in S: Sistema, in m: materia) : \mathbb{Z}

aux[] = S.dn.obtener(m)

int res = aux[7] + aux[8] + aux[9] + aux[10]

return res

→ ¿cómo se relacionan las var del módulo?

2. Su invariante de representación y su función de abstracción

¿cómo se relacionan las var del módulo con los obs del TAD?

Módulo sistema

$\begin{cases} dm = \text{diccLog}(\text{materia}, \text{diccLog}(\text{alumno}, \text{nota})) \\ dn = \text{diccLog}(\text{materia}, \text{array}[11]:\mathbb{Z}) \end{cases}$

TAD Sistema {

obs notas: dict⟨materia, dict⟨alumno, \mathbb{Z} ⟩⟩

pred invRep (S: Sistema)

1° claves(dm) = claves(dn) ←

2° $\forall m:\text{materia} (\forall n:\text{nota} (0 \leq n \leq 10 \wedge m \in \text{claves}(dm)$

$\Rightarrow \exists c:\text{conj(alumnos)} (\forall a:\text{alumno}) (a \in c \Rightarrow dm[m][a] = n$

$\wedge \text{tamaño}(c) = dn[m][n]$

pred funAbs ($\frac{\text{módulo}}{S:\text{Sistema}}, \frac{\text{TAD}}{S':\text{Sistema}}):$

S.dm = S'.notas

Ejercicio 4. Considerar la siguiente extensión sobre el TAD Conjunto(\mathbb{Z})

TAD Conjunto(\mathbb{Z}) {

obs elems: conj(\mathbb{Z})

proc cantElementosEnRango(in c : Conjunto(\mathbb{Z}), in desde : \mathbb{Z} , in hasta : \mathbb{Z}) : \mathbb{Z}

requiere { desde \leq hasta }

asegura { $(\exists c' : \text{conj}(\mathbb{Z})) ((\forall e : \mathbb{Z}) (e \in c' \iff e \in c \wedge \text{desde} \leq e < \text{hasta}) \wedge \text{res} = |c'|) \}$ }

1. Escriba el algoritmo para el cantElementosEnRango si el TAD Conjunto(\mathbb{Z}) está implementado sobre ABB. ¿Qué complejidad tiene?

```
private class Nodo {
```

```
→ T valor;
```

```
→ Nodo izq;
```

```
→ Nodo der;
```

```
→ Nodo padre;
```

```
Nodo(T v) {
```

```
valor = v;
```

```
izq = null;
```

```
der = null;
```

```
padre = null;
```

```
}
```

```
}
```

```
public class ABB<T extends Comparable<T>> implements Conjunto<T> {
```

```
private Nodo _raiz;
```

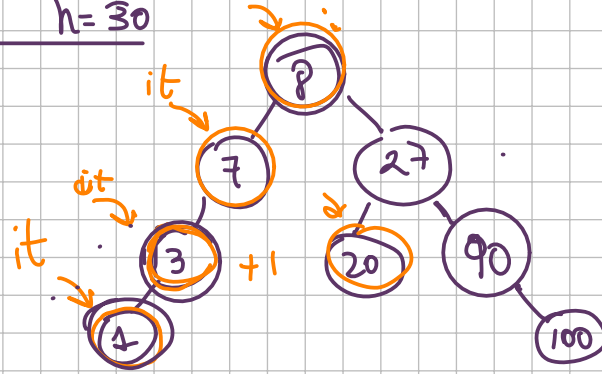
```
public ABB() {
```

```
_raiz = null;
```

```
}
```

```
}
```

d = 2 h = 30



Proc CantElemEnRango (in c:conj(\mathbb{Z}), d:int, h:int)

int cont = 0

it = c.iterator()

while (it.haySiguiente() && it.valor < d)

it = it.siguiente

→ while (it.haySiguiente() && it.valor < h)

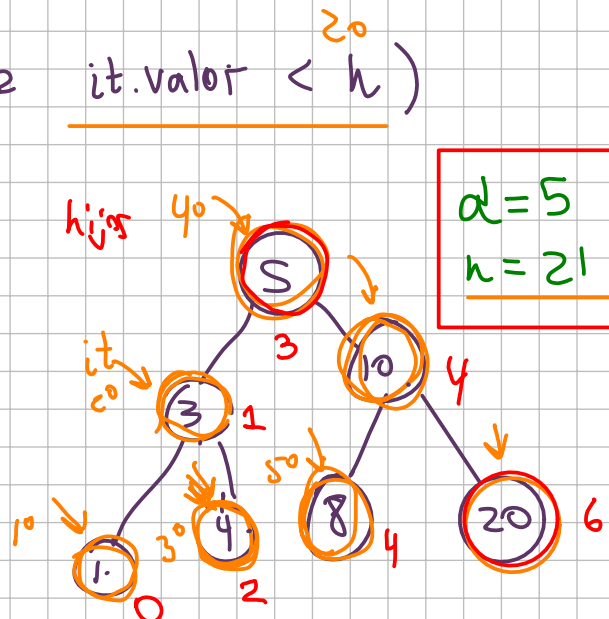
→ cont ++

→ it.siguiente()

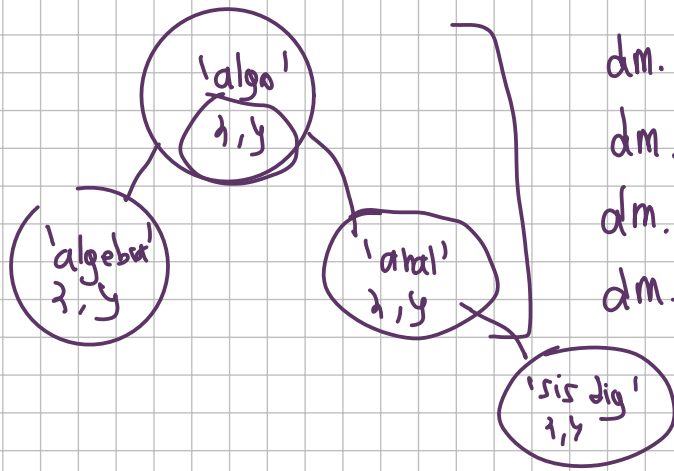
return cont

IteradorBidireccional

proc	complejidad
haySiguiente	$O(\log n)$
hayAnterior	$O(\log n)$
siguiente	$O(\log n)$
anterior	$O(\log n)$



AVL dm: < materia, da >



dm. definir ('algo', 3, 4)

dm. definir ('algebra', 3, 4)

dm. definir ('anal', 3, 4)

dm. definir ('sis dig', 3, 4)