

E1. Complejidad [20 pts]

Responder, justificando adecuadamente su respuesta:

- Decimos que la complejidad de insertar en una lista enlazada ordenada es $O(n)$ en el peor caso, pero eso sólo es cierto si podemos comparar dos elementos en $O(1)$ y lo mismo para la operación de copia. Calcule la complejidad de insertar en una lista enlazada ordenada *sin aliasing* (es decir, copiando en la lista el elemento a insertar y no haciendo una referencia al mismo) si comparar es $O(\log n)$ y copiar es $O(n)$.
- Si la complejidad en el peor caso de un algoritmo es $\Omega(n)$, ¿es verdad que la complejidad de mejor caso no puede ser $O(1)$?
- Indique si la siguiente expresión es verdadera o falsa y justifique: Si $O(f(n)) \cap \Omega(g(n)) = \emptyset$, entonces $O(g(n)) \cap \Omega(f(n)) = \emptyset$.

a) En peor caso, lo compara con todos.

COMP INSERCIÓN

Al insertar la $O(n)$ y Comparar $O(1)$ hacerlo x todo en $O(\overbrace{m+m}) = O(m)$

Otro, si Comparan en $\log m$ y Copian m enti

$$O(\overbrace{m + m \cdot \log(m)}) = O(m \cdot \log(m))$$

COPIAN COMPARAN CON TODOS

b) PEOR CASO EN $\Omega(m)$

c) $\underbrace{O(f(m)) \cap \Omega(g(m))}_{\text{BUSCO CASO V}} = \emptyset \Rightarrow O(f(m)) \cap \Omega(g(m)) = \emptyset ?$

$$f(m) = m^2 \quad g(m) = m^3$$

$$\underbrace{O(m^2) \cap \Omega(m^3)}_{\{m, m^2\} \cap \{m^3, m^4\}} = \emptyset \nrightarrow \underbrace{O(m^3) \cap \Omega(m^2)}_{\{m^3, m^2\} \cap \{m^2, m^3\}} = \{m^2, m^3\}$$

Luego es falso y habremos continuado.

E3. Estructuras [20 pts]

¿Qué estructura elegiría para implementar cada uno de los siguientes conjuntos? Justifique su respuesta indicando qué ventajas tiene la estructura que eligió.

- a) un conjunto de a lo sumo 100 números de 64 bits (entre 1 y 18446744073709551615)
 b) un conjunto de mediciones en el que se agregan nuevas mediciones muy frecuentemente y se consulta muy poco
 c) un conjunto de palabras sobre un alfabeto no acotado en el que se inserta poco y se consulta mucho

↑
 ¿un conjunto? ¿Alguno consideran el conjunto de que dice O NO?

- a) Como tiene órdenes para 100 y los Múltiples podría elegir en ARRAY.
 El conjunto se obtiene 1 más específicas o bien \exists dentro de agregar
Algunas todos 0/1 porque los primeros los definió con anterioridad.

VENTAJAS DE ARRAY ACOTADO:

- Recorrer ARRAY $O(1)$
- Inserción y/o modificación $O(1)$.

- b) Si se agrega más y se comprueba por necesario crear lugar
 Que no sea largo el array solviendo que no sea acotado.

TRIE: NO. Ya se formación palabras o combinaciones
ARRAY: NO. Es largo para completar para seguirán REDIMENSIONAR.

HEAP: NO. No nos sabemos el índice MAX para desordenar.

AVL: NO. No formamos múltiples, la complejidad es log(n).

LF: SÍ. Si tenemos el primero en el índice NO, el conjunto
 De inserción es $O(1)$.

ABBT/AVL: NO. No me importa el orden.

CONJUNTO LINEAL: Dependiente. Si la idea es obtener el índice no nos
permite formar el conjunto todos nos dicen que en el índice 0 no.

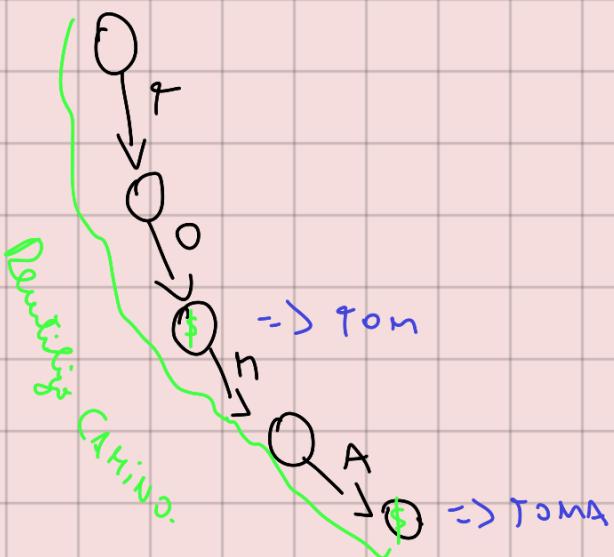
CONJUNTO LOG: Same CONJUNTO LINEAL.

DICCIONARIO LINEAL/LOG: NO. No el mon formación que los DA's son K,V.

RTA: L.F. Con PUNTOS AL FINAL $O(1)$ si no es, Con CONTINUO $O(n)$.

- c) Ol Roberto de palabras, en may índice pero que es en TRIE
 Si la complejidad de agregar en $O(|k|)$ donde k es la palabra
 \rightarrow que \exists un camino o cola palabra.

Se considera menor directamente de la palabra menor * y tienen menor "sentido" considerar libreta & hoy max 256 char.



E4. Sorting [30 pts]

Dado un arreglo de pares <estudiante, nota>, queremos devolver un arreglo de pares <estudiante, promedio>, ordenado en forma descendente por promedio y, en caso de empate, por número de libreta. → Asc
Los estudiantes se representan con un número de libreta (puede considerar que es un número de a lo sumo 10 dígitos), y la nota es un número entre 1 y 100. La cantidad de estudiantes se considera no acotada.

Se pide dar un algoritmo que resuelva el problema con complejidad $O(n + m \log m)$, donde n es la cantidad total de notas (es decir, el tamaño del arreglo de entrada), y m es la cantidad total de estudiantes.

Ejemplo:

Entrada: <12395, 72>, <45615, 81>, <12395, 94>, <45615, 100>, <78920, 81>, <12395, 90>, <45615, 71>, <78920, 50>

Salida: <12395, 85.3333>, <45615, 84>, <78920, 65.5>

- Describa cada etapa del algoritmo con sus palabras, y justifique por qué cumple con las complejidades pedidas.
- Escriba el algoritmo en pseudocódigo. Puede utilizar (sin reescribir) todos los algoritmos y estructuras de datos vistos en clase.

TRIE

No ACIERTO

Dicho algoritmo ESTUDIANTE, ACUM, CANT → (COSTO O(n))

Luego, m · lg(m) Recorren TODOS los ESTUDIANTES AGUARDANDO

Por LU Nro Dígitos y MAX 10. Pueden ser en TRIE.

Ent.

VAN ALUMNOS NOTAS := NEW Diccionario<Digital<LU, (ACUM, CANT)>>;

// Recorre el ARREGLO DE NOTAS, y GUARDA para ese libro de Diccionarios
lu.Menor q ho.CANT. → Costo O(n)

Buscan si el ALUMNO Z en el TRIE. Cuesta O(1) q Busco por LU q es su
Diccionario. Guardar q Alumno también.

VAR i := 0;

WHILE (i < NOTAS) DO O(m)

VAR ESTA := ALUMNOSNOTAS.ESJA(NOTAS[i]_0); O(1) \rightarrow ACORRADO

IF (ESTA == TRUE) THEN

VAR ANT := ALUMNOSNOTAS.OBTENER(NOTAS[i]_0); // (ALUM, CANTNOTAS) O(1) \rightarrow ACORRADO

ALUMNOSNOTAS.DEFINIR(NOTAS[i]_0, (ANT_0 + NOTAS[i]_1, ANT_1 + 1)); O(1) \rightarrow ACORRADO

ELSE

ALUMNOSNOTAS.DEFINIR(NOTAS[i]_0, (NOTAS[i]_1, 1)); O(1) \rightarrow ACORRADO

ENDIF

i++

END WHILE.

Otro recorrido usando el PROMEDIO. Pasa de n^2 A nm Y TUPLES
(en iteracion = $O(m)$)

VAR ALUMNOS: ARRAY<(LU, PROMEDIO)> := NEW ARRAY||{ALUMNOSNOTAS.TAMAÑO()};
CANT ALUMNOS ↑

VAR iT := ALUMNOSNOTAS.ITERADOR();

i := 0;

WHILE (iT.HAYSIGUIENTE() == TRUE) DO // $O(m)$

VAR DATA := iT.SIGUIENTE(); (LU, (ALUM, CANTNOTAS)) O(1)

ALUMNOS[i] := (DATA_0, DATA[0]/DATA[1]);

i++

END WHILE.

ALUMNOS := MERGESORT(ALUMNOS) // LU ASC $\rightarrow O(m \cdot \log(m))$

ALUMNOS := MERGESORT(ALUMNOS) // PROMEDIO DESC $\rightarrow O(m \cdot \log(m))$

RETURN ALUMNOS;

NOTAS

ORDENAR

Llego, $O(m + m + m \cdot \log(m) + m \cdot \log(m)) = O(m + m \cdot \log(m))$

$m \leq m \cdot \log(m)$

Ejercicio 1. Probar utilizando las definiciones que:

a) $n^2 - 4n - 2 = O(n^2)$.

b) Para todo $k \in \mathbb{N}$ y toda función $f : \mathbb{N} \rightarrow \mathbb{N}$, si $f \in O(n^k)$, entonces $f \in O(n^{k+1})$.

c) Si $f : \mathbb{N} \rightarrow \mathbb{N}$ es tal que $f \in O(\log n)$, entonces $f \in O(n)$.

a)

OP 1:

En TCRN se $\text{Cn} \cdot O(n^2 - 4n - 2) = O(n^2 - 4n) = O(n^2 - n) = O(n^2)$ ✓

OP 2:

$$n^2 - 4n - 2 \leq C \cdot n^2$$

$$\Leftrightarrow \frac{n^2 - 4n - 2}{n^2} \leq C$$

$$\Leftrightarrow \frac{1}{C} - \frac{4}{n} - \frac{2}{n^2} \leq C$$

$\Rightarrow 0 \leq C \Rightarrow C$ es un \mathbb{R} muy grande en tanto que $C \geq 0$.

luego, $n^2 - 4n - 2 \in O(n^2)$

b) $f \in O(n^k) \Rightarrow f \in O(n^{k+1})$

Existe \forall función $f \in O(n^k) \subseteq f \in O(n^{k+1})$

$$f \in O(n^k) \equiv f \leq C \cdot n^k \Rightarrow f = C \cdot n^k$$

↓ Reemplazar.

$$f \in O(n^{k+1}) \equiv f \leq C \cdot n^{k+1} \Rightarrow f \leq C \cdot n^k \cdot n$$

$$\Rightarrow C \cdot n^k \leq C \cdot n^k \cdot n$$

$\Rightarrow 0 \leq n$ dadas las \forall para $n \in \mathbb{N}$.

c) $f \in O(\log n) \Rightarrow f \in O(n)$

$$|f \leq C \cdot \log(n)| \quad |f \leq C \cdot n|$$

$f = C \cdot \log(n)$

$$\Rightarrow C \cdot \log(n) \leq C \cdot n$$

$\Rightarrow \log(n) \leq n$ y esto es V, $\forall n \in \mathbb{N}$.

Ejercicio 2. Determinar la verdad o falsedad de cada una de las siguientes afirmaciones. **Justificar.**

- | | |
|---|---|
| a) $2^n = O(1)$. | g) $\log n = O(n)$. |
| b) $n = O(n!)$. | h) $n! = O(2^n)$. |
| c) $n! = O(n^n)$. | i) $n^5 + bn^3 \in \Theta(n^5) \iff b = 0$. |
| d) $2^n = O(n!)$. | j) Para todo $k \in \mathbb{R}$ $n^k \log(n) \in O(n^{k+1})$. |
| e) Para todo $i, j \in \mathbb{N}$, $i \cdot n = O(j \cdot n)$. | k) Para toda función $f : \mathbb{N} \rightarrow \mathbb{N}$, $f = O(f)$. |
| f) Para todo $k \in \mathbb{N}$, $2^k = O(1)$. | |

a) $2^m \leq C \cdot 1 \iff 2^m \leq C$ fuer m la menorre de C fijo.
 Ent 2^m $\notin O(1)$ ✓

b) $m \leq C \cdot m!$ $\iff m \leq C \cdot m \cdot (m-1)!$ $\iff 1 \leq C \cdot (m-1)!$ y esto es V, $\forall m \in \mathbb{N}_{>1}$. ✓

c) $m! \leq C \cdot m^m \iff m \cdot (m-1)! \leq C \cdot m \cdot m \cdot m \iff (m-1)! \leq C \cdot m \cdot m$, d's qd V
 Tiene da 0 > 0.

d) $2^m \leq C \cdot m!$, V a partir de m=4.

CB: m=4 $\Rightarrow 2^4 \leq 4! \iff 16 \leq 24$ ✓

$$H_i: 2^k \leq C \cdot k!$$

$$QPQ: 2^{k+1} \leq C \cdot (k+1)!$$

$$\Rightarrow 2^k \cdot 2 \leq C \cdot k+1 \cdot k!$$

$$\Rightarrow C \cdot k! \cdot 2 \leq C \cdot k+1 \cdot k!$$

$$\Rightarrow 2 \leq k+1$$

$$\Rightarrow 1 \leq k \quad \forall k \geq 4.$$

e) $i \cdot m \leq C \cdot (i \cdot m) \iff i \leq C \cdot 1 \quad \forall m \text{ s.t. } i > 1$

$$f) 2^k \leq c \cdot 1 \Leftrightarrow 2^k \leq c \quad \text{FALSO.}$$

Comprobado: $2^6 \leq 2^k$ pero $2^6 \notin O(1)$.

$$g) \log(m) \leq c \cdot m \Leftrightarrow \frac{\log(m)}{m} \leq c \Leftrightarrow \log(m) \leq c \cdot m \quad \text{y esto es V, } \forall m \in \mathbb{N} \quad \checkmark$$

$$h) m! \leq c \cdot 2^m \quad \text{falso.} \quad \checkmark$$

Comprobado: $m=4$

$$2^4 \neq 16$$

$$k) f \leq c \cdot g \Leftrightarrow 1 \leq c, \text{ VERDADERO} \quad \checkmark$$

Ejercicio 3. ¿Qué significa, intuitivamente, $O(f) \subseteq O(g)$? ¿Qué se puede concluir acerca del crecimiento de f y g cuando, simultáneamente, tenemos $O(f) \subseteq O(g)$ y $O(g) \subseteq O(f)$?

QUE f y g crecen igualmente

$$\begin{aligned} & \cdot f \leq c \cdot g \quad \} 1 \\ & \quad \Rightarrow 1 \leq c \cdot g \Rightarrow g \leq c \cdot c \cdot f \Rightarrow 1 \leq c^2 \quad \} = \\ & \cdot g \leq c \cdot f \quad \} 2 \quad \Rightarrow 1 \leq c \cdot f \Rightarrow f \leq c \cdot c \cdot g \Rightarrow 1 \leq c^2 \quad \} = \end{aligned}$$

Ejercicio 5. Para cada una de las siguientes afirmaciones, decida si son verdaderas o falsas y justifique su decisión.

- a) $O(n^2) \cap \Omega(n) = \Theta(n^2)$
- b) $\Theta(n) \cup \Theta(n \log n) = \Omega(n \log n) \cap O(n)$
- c) $f \in O(g) \iff O(f) \subseteq O(g)$
- d) Si $f \in \Omega(g)$, entonces $O(f) \cap \Omega(g) = O(g) \cap \Omega(f)$
- e) Si $f(n) < g(n)$ para todo n , entonces $\Theta(f) = \Theta(g)$
- f) Si $f \in O(g)$, entonces $f * g \in \Theta(g)$

a) F.

$$O(n^2) \cap \Omega(n) = \{n^2, n, \dots\} \cap \{n, n^2, \dots\} = \{n, n^2\} \quad \text{y no } \Theta(n^2).$$

pero $n < n^2$.

O quiere decir q es EXACTAMENTE. Luego, es Falsa. \checkmark

$$b) \Theta(m) \cup \Theta(m \log(m)) = \Theta(m \log m)$$

$$\mathcal{R}(m \log(m)) \cap O(m) = \{m \cdot \log(m)\} \cap \{m, 1\} = \emptyset$$

CONTAR EJEMPLO:

FAUSA. Pues $m \cdot \log(m) \in \Theta(n) \cup \Theta(m \cdot \log(m))$ pero $\notin \mathcal{R}(m \cdot \log(n)) \cap O(m)$ ✓

$$c) f \in O(g) \Leftrightarrow O(f) \subseteq O(g)$$

$$\Rightarrow \text{Caso } f \leq c \cdot g \text{ con } c \cdot f \leq c \cdot g$$

$$\Leftrightarrow O(f) \subseteq O(g) \Rightarrow f \in O(g)$$

$$\text{Caso } f \in O(p) \wedge O(p) \subseteq O(g) \text{ con } f \in O(g) \quad \checkmark$$

$$d) f \in \mathcal{R}(g) \Rightarrow O(f) \cap \mathcal{R}(g) = O(g) \cap \mathcal{R}(f)$$

$$f = m$$

$$\mathcal{R}(f) = \{m, m^2\} \quad f = m^2$$

$$\begin{array}{ll} O(p) = \{m^2, m\} & O(g) = \{m, 1\} \\ \mathcal{R}(g) = \{n, n^2\} & \mathcal{R}(p) = \{m^2, n^3\} \end{array}$$

$$\text{Contar ejemplos: } g = m \quad f = m^2$$

$$\begin{aligned} f \in \mathcal{R}(g) &\not\Rightarrow \{m^2, m, \dots\} \cap \{m, m^2, \dots\} = \{m, \dots, 1\} \cap \{m^2, \dots\} \\ &\not\Rightarrow \{m, m^2\} = \emptyset \quad F. \quad \checkmark \end{aligned}$$

$$e) f \in O(p) \Rightarrow f \cdot g \in \Theta(g) \quad \text{FAUSA} \quad \checkmark$$

Contar ejemplos:

$$f = m \quad g = m^2$$

$$f \in O(g) \not\Rightarrow m^3 \in \Theta(m^2) \quad \checkmark$$

Ejercicio 6. Para cada una de las siguientes afirmaciones, decida si son verdaderas o falsas y justifique su decisión.

a) $n + m = O(nm)$.

b) $n + m^5 = O(m^5)$.

c) $nm = O(n + m)$.

d) $m^5 = O(n + m^5)$.

a) $n + m \leq c \cdot nm$

$$\Leftrightarrow \frac{1}{n} + \frac{1}{m} \leq c \quad \forall n, m. \quad \checkmark$$

b) $n + m^5 \leq c \cdot m^5$

$$\Leftrightarrow \frac{n}{m^5} + 1 \leq c$$

$$\Leftrightarrow n + 1 \leq c \quad \text{Pues } m \text{ NO es ACOTADO, Falso} \quad / \quad \text{pues si } n > m^5, n \notin O(m^5)$$

c) $n \cdot m \leq c \cdot (n + m)$

Falso pues si $n = m$ tenemos $n < m^2$ y $m^2 \notin n$.

Ej: $n = m \quad m = m$

$$m^2 \leq m + m \Leftrightarrow m^2 \leq m \quad \text{Falso} \quad /$$

d) $m^5 = O(n + m^5)$

• Si $n > m^5$ entonces $m^5 \in O(n)$

• Si $m^5 > n$ entonces $m^5 \in O(m^5)$

VERDADERA.

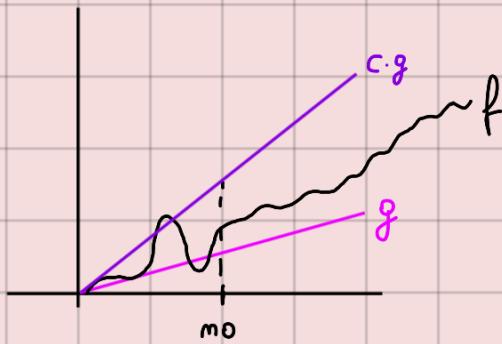
$$m^5 \leq c \cdot (n + m^5)$$

$$\Leftrightarrow \frac{m^5}{n} + 1 \leq c$$

$$\Leftrightarrow m^s + 1 \leq c$$

$f \in O(g) \Rightarrow f$ crece superiormente que g , $\forall m > m_0$

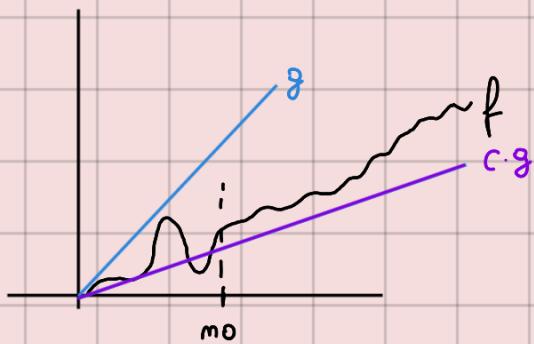
↳ más fuerte de los g dados



$$(\exists c \in \mathbb{R} > 0, m_0 \in \mathbb{N}) (f(m) \leq c \cdot g(m), \forall m > m_0)$$

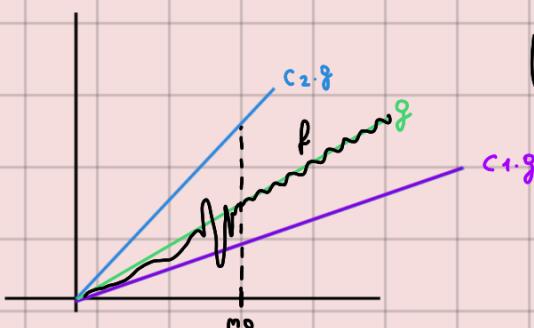
$f \in \Omega(g) \Rightarrow f$ crece inferiormente que g , $\forall m > m_0$

↳ más débil de los g dados



$$(\exists c \in \mathbb{R} > 0, m_0 \in \mathbb{N}) (f(m) \geq c \cdot g(m), \forall m > m_0)$$

$f \in \Theta(g) \Rightarrow f$ crece a la misma tasa que g .



$$(\exists c_1, c_2 \in \mathbb{R} > 0, m_0 \in \mathbb{N}) (c_1 \cdot g(m) \leq f(m) \leq c_2 \cdot g(m), \forall m > m_0)$$

acotada de materias. En cada materia, las notas están entre 0 y 10, y se aprueban si la nota es mayor o igual a 7.

```
TAD Sistema {
    proc RegistrarMateria(inout s: Sistema, in m: materia)
    proc RegistrarNota(inout s: Sistema, in m: materia, in a: alumno, in n: nota)
    proc NotaDeAlumno(in s: Sistema, in a: alumno, m: materia): nota
    proc CantAlumnosConNota(in s: Sistema, in m: materia, n: nota): Z
    proc CantAlumnosAprobados(in s: Sistema, in m: materia): Z
}
```

Objetivo. O(1) en CONSULTA/INSERTIÓN/ACTUALIZACIÓN

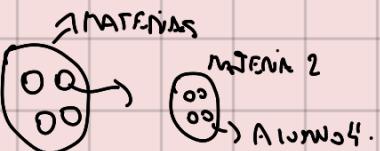
Dados $m = \text{cantmaterias}$ y $n = \text{cantalumnos}$ se desea diseñar un módulo con los siguientes requerimientos de complejidad temporal:

- RegistrarMateria en $O(\log m)$
- RegistrarNota en $O(\log n + \log m)$
- NotaDeAlumno en $O(\log n + \log m)$
- CantAlumnosConNota y CantAlumnosAprobados en $O(\log m)$

1) Registrar materia depende solo de CANT MATERIA. En la inserción en logarítmica tenemos opción por un árbol AVL.

MATERIAS: DICCIONARIO<LOG<MATERIA, ??>

2) Registrar nota recibe MATERIA, ALUMNO y NOTA y es $O(\log m + \log n)$, lo que nos dice que en cada una nos tiene que encontrar la materia, los alumnos tienen su propio ÁRBOL AVL.



Entonces, modificando 1 no quita

MATERIAS: DICCIONARIO<LOG<MATERIA, DICCIONARIO<LOG<ESTUDIANTE, NOTA>>

3) Tener alumno, reutilizar 2 & 1.

4) CANT ALUMNOS CON NOTA

CANT ALUMNOS APROBADOS

* 1) Recorre en cada mat, recorre cada alumno y si la nota menor 10 es en $O(m \cdot (\log m + \log n))$ y NO cumple.

Pues en $O(\log m)$ donde m es CANT MATERIAS dentro de un $O(\log m)$ dentro MATERIA pero la suma d繁e ser $O(1)$

Pues me hablan de CANT ALUMNOS CON NOTA para una materia el que cumple 0-10 nota, pues en ese caso se haría 11 veces cosa que no me

Dice la CANT de alumnos con esa materia.

O(1)

AlumnosNota<MATERIA, ARRAY<NOTA>{11}>

Ofrece, ¿Qué materia tiene los mejores y peores notas de un alumno?

- Buena MATERIA en ALUMNOSNOTA: O(log m)
- Ocells o. ARRAY[NOTA] & max 1: O(1)

Luego,

VAR MATERIASALUMNOS: Diccionario<MATERIA, Diccionarios<lu, NOTA>>

VAR NOTASALUMNOS: Diccionario<MATERIA, ARRAY<NOTA>{11}>

El INVERP tiene:

- TODA MATERIA EN MATERIASALUMNOS ESTÁ EN NOTASALUMNOS.
- TODA NOTA EN CADA MATERIA EN MATERIASALUMNOS PARA CADA LU ESTÁ ENTRE 0 y 10.
- TODA MATERIA EN NOTASALUMNOS ESTÁ EN MATERIASALUMNOS.
- LA CANTIDAD (NÚMERO) Q.F. ESTÁ EN CADA POSICIÓN, PARA CADA MATERIA EN NOTASALUMNOS ES LA SUMA DE TODOS LOS ESTUDIANTES DE CADA MATERIA DE MATERIASALUMNOS CON ESA NOTA.

Ej: {"ALGEBRA": {"222/23": 1, "254/21": 1}}

{"ALGEBRA": [0, 1, 0, 0, ... 0]}

POD=11

↳ VINCUL NOTAS DE MATERIASALUMNOS Y NOTASALUMNOS Xq si las KEYS
están sincronizadas y el numero (cant) de Alumnos difiere de MATERIASALUMNOS se
obtendrá que SOLO NOTASALUMNOS difiere de MATERIASALUMNOS y no el resto.

Ejercicio 5. Considere el TAD Diccionario con historia, cuya especificación es la siguiente:

```
TAD DiccionarioConHistoria<K, V> {
    obs data: dict<K, V>
    obs cant: dict<K, int>

    proc nuevoDiccionario(): DiccionarioConHistoria<K, V>
        asegura { res.data == {} }
        asegura { res.cant == {} }
```

```

proc esta(in d: DiccionarioConHistoria<K, V>, in k: K): Bool
asegura { res ↔ k in d.data}

proc definir(inout d: DiccionarioConHistoria<K, V>, in k: K, in v: V)
asegura { d.data == setKey(old(d).data, k, v)}
asegura { d.cant == setKey(old(d).cant, k, sumai(k, old(d).cant))}

proc obtener(in d: DiccionarioConHistoria<K, V>, in k: K): V
requiere { k in d.data}
asegura { v == d.data[k]}

proc borrar(inout d: DiccionarioConHistoria<K, V>, in k: K): V
requiere { k in d.data}
asegura { d.data == delKey(old(d).data, k)}
asegura { d.cant == delKey(old(d).cant, k)}

proc cantSignificados(in d: DiccionarioConHistoria<K, V>, in k: K): Z
requiere { k in d.data}
asegura { res == d.cant[k]}

aux sumai(k:K, cant: dict<K, int>)
{ if (k in cant) then (cant[k] + 1) else 1 fi }

```

- Si n es la cantidad de claves que están definidas en el diccionario, se debe diseñar este TAD respetando los siguientes órdenes de ejecución en el peor caso:

- esta $O(n)$
 - nuevoDiccionario $O(1)$
 - obtener $O(n)$
 - definir $O(n)$
 - borrar $O(n)$
 - cantSignificados $O(n)$

2. Si quisieramos conservar la historia de cada clave cuando se borran y luego vuelve a definir, ¿es posible mantener las complejidades de todas las operaciones? Si es posible, explique las modificaciones necesarias a la estructura. Si no, explique las modificaciones necesarias para que cambien la menor cantidad posible de complejidades, cuáles cambian y por qué.

1)

1) ESTA, O(m) significa que TASA O(m) de vez en cuando es KEY. Podría ser

DATA: DICCIONARIO LINEAL (KEY, VALOR)

2) OBTENER, O(M) MISMO 1) (que el registro me dice que la KEY está, no hace falta que haga programación reflexiva).

3) BORRAR, $O(n)$. BORRAR EN DATA, δ EN UNA ESTRUCTURA DE
CANT: DICCIONARIO<LINES, (KEY, CANT)>

Como sé que para BORRAR KEY E DATA, si ESTÁ EN DATA tambien en D.CANT.

Er deir, de manieren den Keys einzuordnen.

4) DEFINIER, O(M) UND O(m) FÜR DEN ALGO WEIFEN DASS NO ERST DEFINIERT IM D.CANT, XQ
AUF ESGN DEFINIE BESZ D.CANT[KEF]:= D.CANT[KEF] + 1 UND D.CANT[KFY]:= 1;

Con D. DATA me hace falta n.º 619 o no es más el valor directo.

La m viene de q solo Tardos la vino xq tienen las minor keys. De los latidos, una serie m q dice m (ej).

9) CANTSIGNIFICADOS, 0(m) en BUSCAR PONKEY en D.CANT. (nos el regresa me dice que la KEY existe, me hace falta que haga programación de función).

2) Cuanto somos lo chau, mejor q no le haga se

en CANT. En decir, formó algo así:

DAT: {"A": 1} CANT: {"A": 1}

BORRAR A

DATA: {} CANT: {"A": 1}

AGREGAR B: 1

DATA: {"B": 1} Costo O(m) CANT: {"A": 1, "B": 1} Costo O(m)

Como como keys difieren el costo de consulta

θ inserción cambia.

INVREP:

- TODA CLAVE EN DATA ESTÁ EN CANT.
- TODA CLAVE EN CANT ESTÁ EN DATA.

ABSTRACTS:

- PARA TODA CLAVE EN D'.DATA ESTÁ EN D.DATA y el $D'.Data[k] = D.Data[k]$
 - PARA TODA CLAVE EN D'.CANT ESTÁ EN D.CANT y el $D'.Cant[k] = D.Cant[k]$
- No relaciona $D'.Data$ (KEY) con $D.Cant$ o $D'.Cant$ (KEY) y $D.Data$ para que sea implícito por INVREP.

VALOR.

Módulo $DCh<K,V>$ implementa $\text{DiccionarioConHistoria}<K,V>$ {

VAR DATA: DiccionarioLineal<K,V>

VAR CANT: DiccionarioLineal<K,int>

PROC NUEVODiccionario(): DCh<K,V> {

RES.DATA := NEW DiccionarioLineal<K,V>();

RES.CANT := NEW DiccionarioLineal<K,V>();

}

PROC ESTA(in D:DCh<K,V>, in K:K): Boolean {

RETURN D.DATA.ESTA(K); //O(m)

```

PROC DEFINIR(inout D:DCH<K,V>, in K:K, in V:V) {
    D.DATA.DEFINIR(K,V); //O(m)
    IF(D.CANT.ESTA(K) == TRUE) THEN
        VAR CANT:int := D.CANT.OBTENER(K); //O(m)
        D.CANT.DEFINIR(K, CANT+1); //O(m)
    ELSE
        D.CANT.DEFINIR(K,1); //O(m)
    ENDIF
}

```

```

PROC OBTENER(in D:DCH<K,V>, in K:K):V {
    RETURN D.DATA.OBTENER(K); //O(m)
}

```

```

PROC BORRAR(inout D:DCH<K,V>, in K:K):V {
    REQUERIRE:  $\exists K \in D.CANT$  //Por TAD, al definir se guarda K en DATA y CANT  $\Rightarrow$  SI NEQUIENE
    TAD dice K se guarda en K de CANT indirectamente
    VAR VAL:V := D.DATA.OBTENER(K); //Por Copia
    D.DATA.BORRAR(K); //O(m)
    D.CANT.BORRAR(K); //O(m)
    RETURN VAL
}

```

```

PROC CANTSIGNIFICADOS(in D:DCH<K,V>, in K:K):int {
    REQUERIRE: {K ∈ D.CANT} //Por TAD, al definir se guarda K en DATA y CANT  $\Rightarrow$  SI NEQUIENE
    TAD dice K se guarda en K de CANT indirectamente
    RETURN D.CANT.OBTENER(K); //O(m)
}

```

