Ej. 1. Diseño

En una elección intervienen *agrupaciones* políticas, numeradas desde 1 hasta G. Cada agrupación nuclea a varios *candidatos* en una lista electoral. Cada candidato se identifica por un número natural. Durante la jornada de elecciones los votantes registran sus votos en *mesas* electorales. Los votantes se identifican por su DNI, y las mesas se identifican por su nombre, que es un string de largo M. Al finalizar los comicios, cada mesa registra el resultado del recuento en el Centro de Procesamiento de Datos (CPD), que sumariza la información. Los datos que envía la mesa incluyen el conjunto de votantes que votaron en esa mesa y el número de votos para cada agrupación. Se desea diseñar el CPD para cumplir con los requerimientos detallados abajo. **Importante:** una mesa puede registrar varias veces los correspondientes datos al CPD, con el fin de corregir posibles errores u omisiones. Cada vez que una mesa registra sus datos en el CPD, toda la información que esa mesa hubiera enviado anteriormente se descarta por completo, y se sobreescribe con la nueva información.

TAD CANDIDATO es NAT **TAD** AGRUPACIÓN es NAT **TAD** DNI es NAT **TAD** MESA es STRING

TAD CPD

observadores básicos

agrupaciones	: cpd	$\longrightarrow$	conj(agrupación)	
candidatos	: cpd c × agrupación a	$\longrightarrow$	secu(candidato)	$\{a \in \text{agrupaciones}(c)\}$
votantes	: cpd × mesa	$\longrightarrow$	conj(dni)	
votos	: cpd × mesa	$\longrightarrow$	multiconj(agrupación)	

generadores

iniciar	: dicc(agrupación × secu(candidato))	$\longrightarrow$	cpd
registrar	: cpd c × mesa × conj(dni) ds × multiconj(agrupación) as	$\longrightarrow$	cpd
	$\{\#(ds) = \#(as) \wedge (\forall\, a : \text{agrupación})(a \in as \Rightarrow a \in \text{agrupaciones}(c))\}$		

axiomas

...

Fin TAD

Se debe realizar un diseño que provea las siguientes operaciones con las complejidades en peor caso indicadas:

- REGISTRAR(**inout** c : CPD, **in** m : MESA, **in** ds : CONJ(DNI), **in** as : MULTICONJ(AGRUPACIÓN)) — Registrar la información de una mesa — $O(M + G + V_A \cdot \log V)$, donde V es la cantidad total de votantes que hay en el sistema y V_A es la cantidad total de votantes que son *afectados* por esta operación, incluyendo a todos los votantes del conjunto ds y a todos aquellos que tuvieran votos registrados anteriormente en la mesa m. Recordar que esta operación debe *descartar* toda la información que hubiera previamente registrada en la mesa m.
- VOTOSPARA(**in** c : CPD, **in** x : CANDIDATO) $\rightarrow$ res : NAT — Determinar el número de votos que recibió el candidato x, sumando los votos que recibieron todas las agrupaciones en las que aparece. (Notar que un candidato puede figurar en las listas de varias agrupaciones distintas). — $O(G + \log K)$.
- REPETIDOS(**in** c : CPD) $\rightarrow$ res : CONJ(DNI) — Obtener el conjunto de DNIs de los votantes que tengan votos registrados en dos o más mesas diferentes. Observar que este conjunto puede "achicarse" si una mesa registra una versión corregida de sus datos. — O(1).

donde M es la longitud de los nombres de las mesas, G es el número de agrupaciones (recordar que **las agrupaciones se numeran desde 1 hasta** G) y K es la cantidad total de candidatos.

1. Dar una estructura de representación del módulo CPD explicando detalladamente qué información se guarda en cada parte, las relaciones entre las partes, y las estructuras de datos subyacentes.
2. Justificar de qué manera es posible implementar los algoritmos para cumplir con las complejidades pedidas. Escribir el algoritmo para la operación REGISTRAR.

Ej. 2. Ordenamiento

Sea $A[0..n-1]$ un arreglo de n números naturales distintos entre sí. Dado un número $k \in \{1,\ldots,n\}$ decimos que el arreglo A está **k-ordenado** si para todo índice $i < n$ se tiene que el i-ésimo elemento más chico aparece en el arreglo antes de la posición $i+k$. Más precisamente, sea $B[0..n-1]$ el arreglo A ordenado de menor a mayor. Entonces:

$$A \text{ está } k\text{-ordenado} \iff (\forall i : \text{nat})(i < n \Rightarrow_{\mathsf{L}} (\exists j : \text{nat})(j < \text{mín}(i+k, n) \wedge_{\mathsf{L}} B[i] = A[j]))$$

Dicho de otro modo, los primeros m elementos de B se pueden encontrar en el prefijo de A que tiene tamaño $m+k$.

Por ejemplo, el arreglo $A = [5,2,4,7,6]$ está 3-ordenado porque el arreglo ordenado es $B = [2,4,5,6,7]$, de tal forma que el elemento $B[i]$ aparece en la posición j de A con $j < i+3$:

i	j	$< i+3$
0	1	< 3
1	2	< 4
2	0	< 5
3	4	< 6
4	3	< 7

Notar que si el arreglo A está 1-ordenado entonces está ordenado de menor a mayor.

Suponiendo que A está k-ordenado, se pide proponer un algoritmo para ordenarlo con costo $O(n \log k)$ en peor caso. Justificar adecuadamente su complejidad.

Nota: quizás puede ayudar pensar primero en el caso en el que A está 2-ordenado. En tal caso el algoritmo debería ser $O(n)$.

Ej. 3. Dividir y Conquistar

En este ejercicio llamamos **cadena** a una secuencia que empieza en un número x y tiene $k \geq 1$ elementos, cada uno de los cuales es el doble del anterior, es decir, una cadena es una secuencia de la forma $[x, 2x, 4x, \ldots, 2^{k-1}x]$. Dado un arreglo $A[0..n-1]$ de n números naturales ordenados *estrictamente* de menor a mayor (sin repetidos), se quiere implementar una operación:

LONGITUDCADENAMÁSLARGA(in A : ARREGLO(NAT)) → NAT

para determinar la longitud de la subsecuencia[1] del arreglo A que determina la cadena más larga.

Por ejemplo, en el arreglo ordenado $A = [2,3,4,6,8,12,16,18,32,36]$ la longitud de la cadena más larga es 5 (y sus elementos son $[2,4,8,16,32]$). Notar que $[4,8,16]$ y $[3,6,12]$ también son cadenas del arreglo A, pero no tienen la longitud más larga posible. Notar también que podría haber varias cadenas que "empaten" de tal modo que todas tengan la longitud más larga posible.

Proponer un algoritmo que implemente LONGITUDCADENAMÁSLARGA. La complejidad temporal del algoritmo debe ser estrictamente menor que $O(n^2)$ en peor caso. Determinar y justificar la complejidad del algoritmo propuesto.

Sugerencia: representar una cadena $[x, 2x, 4x, \ldots, 2^{k-1}x]$ como una tripla (b,p,q) cuya primera componente b es un número impar tal que el extremo izquierdo de la cadena es $x = 2^p \cdot b$ y el extremo derecho de la cadena es $2^{k-1}x = 2^q \cdot b$, es decir, $q - p = k - 1$. Por ejemplo, la cadena $[12,24,48]$ se puede representar como la tripla $(3,2,4)$. Suponer que se cuenta con una función DESCOMPONER : Nat → ⟨Nat, Nat⟩, con costo $O(1)$ en peor caso, tal que DESCOMPONER$(x) = (b,p)$ donde b es un número impar y $x = 2^p \cdot b$. Por ejemplo, DESCOMPONER$(12) = (3,2)$.