**Ejercicio 9** *(MergeSelectivo)*

Dados dos arreglos de naturales, ambos ordenados de manera creciente, se desea buscar, dada una posición $i$, el $i$-ésimo elemento de la unión de ambos. Dicho de otra forma, el $i$-ésimo del resultado de hacer merge ordenado entre ambos arreglos. Notar que no es necesario hacer el merge completo. Se puede asumir que cada natural aparece a lo sumo en uno de los arreglos, y a lo sumo una vez.

a) Implementar la función *iésimoMerge* que dados los arreglos $A$ y $B$, y un valor ´$i$ natural, resuelva el problema planteado.

b) Calcular y justificar la complejidad del algoritmo propuesto. La complejidad temporal debe ser $O(\log^2 n)$, dónde $n = \text{tam}(A) = \text{tam}(B)$. **Hint:** Observar que, dado el valor de un elemento de alguno de los dos arreglos, se puede averiguar en tiempo $O(\log n)$ entre qué par de posiciones consecutivas del otro arreglo quedaría, y de allí deducir cuál sería su posición en el merge.

c) **Desafío adicional:** Intente resolver el mismo problema en tiempo $O(\log n)$ (este ítem es bastante más difícil).

A y B → arreglos de naturales ordenados crecientemente donde c/ natural aparece a lo sumo en 1 de los arreglos y a lo sumo 1 vez

↳ ordenados sin repetidos

A = [1, 3, 5, 9] → i = 5

B = [2, 4, 6, 10]

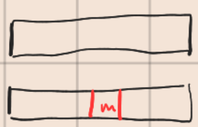
[1, 2, 3, 4, 5] ~~6, 9~~ 10]

→ Como los arreglos están ordenados podemos hacer una búsqueda binaria sobre 1 de ellos.

↳ tienen el mismo tamaño

↳ A =

B = m



Idea algoritmo:

1. Dame el elemento de la mitad de B
2. Buscame la pos en la que iría en A. Sup que es j → A[j] > B[mitad]
3. Si B[mitad] iría en la pos j de A, si yo mergeara me quedaría en la pos j + mitad del arreglo mergeado. Esto es porque al B estar ordenado todos los elementos que van de 0 a mitad -1 < mitad

4. Si $j + mitad = i \rightarrow$ devolver $B[mitad]$

$j + mitad = i-1 \rightarrow$ devolver $\min\{B[mitad+1], A[j]\} \rightarrow$ si $j = i-1 \rightarrow$ como $A[j] > B[mitad]$ y $B[mitad+1] > B[mitad]$ y ambos están ordenados crecientemente $\rightarrow$ el próximo en el merge está entre estos

$j + mitad = i+1 \rightarrow$ devolver $\min\{B[mitad-1], A[j-1]\} \rightarrow A[j] > B[mitad] > A[j-1]$ y $B[mitad-1] < B[mitad] \rightarrow$ entre el mínimo de $B[mitad-1]$ y $A[j-1]$ va a estar el próx elem a colocar.

no hay repetidos

Las 2 justif valen porque como A y B están ordenados y no tienen reps $\rightarrow$ los valores de A y B propuestos son los ⊕ cercanos a $B[mitad]$

$j + mitad > i+1 \rightarrow$ descarto el subarray $B[mitad+1, \dots, n]$

$j + mitad < i-1 \rightarrow$ descarto el subarray $B[0, \dots, mitad]$

5. Si no terminé busco en el array que corresponda de B hasta finalizar

$\rightarrow$ Como tengo 2 binary searchs anidadas y las complejidades de las comparaciones son $O(1)$ ya que involucran acceder a elementos indexados y comparar ints

$\rightarrow T(n) = O(\log^2 n)$