Ejercicio 2 *(ÍndiceEspejo)* ★

Tenemos un arreglo $a = [a_1, a_2, \ldots, a_n]$ de n enteros distintos (positivos y negativos) *en orden estrictamente creciente.* Queremos determinar si existe una posición i tal que $a_i = i$. Por ejemplo, dado el arreglo $a = [-4, -1, 2, 4, 7]$, $i = 4$ es esa posición.

Diseñar un algoritmo dividir y conquistar eficiente (de complejidad de orden estrictamente menor que lineal) que resuelva el problema. Calcule y justifique la complejidad del algoritmo dado.

Buscar un elemento en un arreglo ordenado de elementos distintos → búsqueda lineal

Me pregunto si $A[\lfloor i/2 \rfloor] = i/2$ → si $A[\lfloor i/2 \rfloor] < i/2$ → el subarray $A[0, \lfloor i/2 \rfloor]$ podemos descartarlo porque $\forall i / 0 \le i \le \lfloor i/2 \rfloor \to A[i] \neq i$

(quizás el largo es impar)

como está ordenado si en $A[\lfloor i/2 \rfloor] < \lfloor i/2 \rfloor \to A[\lfloor i/2 \rfloor - 1] < \lfloor i/2 \rfloor - 1$

Esto es porque si $A[\lfloor i/2 \rfloor] < \lfloor i/2 \rfloor \to A[\lfloor i/2 \rfloor] \le \lfloor i/2 \rfloor - 1$

→ si $A[\lfloor i/2 \rfloor] = \lfloor i/2 \rfloor$ ✓

→ si $A[\lfloor i/2 \rfloor] > \lfloor i/2 \rfloor$, descartamos. La justificación es análoga

Conquer → $A[\lfloor i/2 \rfloor] = \lfloor i/2 \rfloor$ $\Theta(1)$ → no puede ser el caso base XQ ya estaríamos asumiendo que ∃ un elemento $/ a_i = i$

Divide → quedarme con un subarray

$[-4, -1, 2, 4, 7]$ → $|A| = 5$ → $\lfloor |A|/2 \rfloor = 2$ → $A[2] = 2$? NO $-1 \neq 2$ como $-1 < 2$

nos quedamos con $[2, 4, 7]$ (índices 3, 5) → $|A| = 5 - 3 = 2$ → $\lfloor |A|/2 \rfloor = 1$ + nueva primera pos $= 1 + 3 = 4$

$A[4] = 4$? sí → me quedo con $[2, 4]$ (índices 3, 4) → $|A| = 4 - 3 = 1$ $\lfloor |A|/2 \rfloor = 0$ ∴

Caso base $|array| = 2$

↳ parte de la dificultad es que necesitamos quedarnos con el arreglo original debido a la indexación

↳ puedo usar variables L y R que se vayan modificando dependiendo del subarray con el que me quede ($R - L/2$)

↳ si $L > 1$ → cuando calculamos $|A|/2$ vamos a tener que sumarle L para

Que sea efectivamente la mitad de nuestro arreglo

```
bool IE( array A, int L, int R ) → invoco con IE(A,1,|A|
  if R-L=1 then                  → si mi subarray tiene size=2
    return A[L]<L ∨ A[R]>R       → me fijo si es una de las puntas
  else
    medio=⌊(R-L)/2⌋
    if L>1 then
      medio += L                 // corrijo indexación
    endif
    if A[medio] < medio
      IE[A, medio+1, R]
    else
      IE[A, L, medio]
```

→ no es tan así porque puede ser A[medio]=medio, no solo A[medio]>medio

$T(n) = 1T(n/2) + \Theta(1)$ → son gratis el conquer y el divide

TM: $\Theta(1) = d \in \Theta(n^{\log_2 1})$

c

$d \in \Theta(n^0)$

$d \in \Theta(1)$ Sí

→ por caso 2 del TM $T(n) = \Theta(n^0 \log n) = \Theta(\log n)$