**Ejercicio 11** *(SubBúsqueda)*

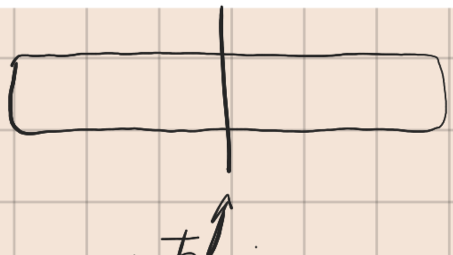
Se tiene un arreglo $A$ de $n$ números naturales. Además se cuenta con estructuras adicionales sobre el arreglo que proveen la función *aparece?* que dado $A$, dos índices $i$, $j$ y un valor natural $e$, devuelve *true* si y solo si $e = A[k]$ para algún $k$ tal que $i \leq k \leq j$. Además se sabe que *aparece?* toma tiempo $O(\sqrt{j-i+1})$, es decir, la raiz cuadrada del tamaño del intervalo de búsqueda.

Se desea encontrar un algoritmo sublineal que encuentra el índice de un elemento $e$ en el arreglo $A$, asumiendo que tal elemento existe en el arreglo. El resultado de la función es justamente el índice $i$ tal que $A[i] = e$.

a) Implementar la función *ubicar?* que tome un arreglo de naturales $A$ de tamaño $n$ y un valor natural $e$, resuelva el problema planteado.

b) Calcular y justificar la complejidad del algoritmo propuesto. La solución debe ser de tiempo estrictamente menor a $O(n)$.

(a)



parto mi
arreglo a la
mitad y pregunto
en qué mitad
aparece
y descarto la que
no lo tiene

```
ubicar? ( A, L, R, e
  if R-L ≤ 0 then                          O(1)
    if R-L = 0 then                        O(1)
      return L
    else
      if A[L] = e then                     O(1)
        return L
      else
        return R
      endif
    endif
  else
    mid = ⌊(L+R)/2⌋                        O(1)

    if aparece?(A, L, mid, e) then         O(√(mid-L+1)) → peor caso O(√(n/2))
      res ← ubicar?(A, L, mid, e)
    else
      res ← ubicar?(A, mid, R, e)
    endif

    return res
```

no hace falta unir el de los otros casos porque la propia recursión lo hace aparecer → sólo 1º llamado

$$T(n) = \begin{cases} T(n/2) + O(\sqrt{n}) & \text{c.c.} \\ \Theta(1) & n=1 \vee n=2 \end{cases}$$

$\rightarrow O(\sqrt{n}) \in \Omega(1)$ y $\sqrt{n/2} < c\sqrt{n}$ con $c<1$?

$\frac{1}{\sqrt{2}}\sqrt{n} < c\sqrt{n}$ si $c = \frac{1}{\sqrt{2}+1}$ vale

$\times$ 3er caso Tm, $T(n) = \Theta(\sqrt{n})$