IzquierdaDominante

versión 1

```python
def izqdom(A, low=0, high=len(inp)):
    if low == high-1: #O(1)
        return True

    mid = low + (high-low)//2 #O(1)

    sumaizq = sum(A[low:mid]) #O(n/2)
    sumader = sum(A[mid:high]) #O(n/2)

    if sumaizq > sumader:
        return izqdom(A, low, mid) and izqdom(A, mid, high) #a=2, c=2
    else:
        return False
```

versión 2

```python
def izqdom(A):
    high = len(A) #O(1)
    low = 0 #O(1)
    mid = high//2 #O(1)

    if high-1 == low: #O(1)
        return True

    sumaizq = sum(A[low:mid]) #O(n/2)
    sumader = sum(A[mid:high]) #O(n/2)

    if sumaizq > sumader:
        return izqdom(A[low:mid]) and izqdom(A[mid:high]) #a=2, c=2
    else:
        return False
```

Complejidad

$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$ y por teorema maestro, notamos que $O(n) = O\left(n^{\log_2 2}\right) \Rightarrow T(n) = \Theta\left(n^{\log_2 2} \log n\right) = \Theta(n \log n)$

IndiceEspejo

```python
def indiceEspejo(A,low,high):

    mid = low + (high-low) //2 #O(1)

    if A[mid] == mid: #O(1)
        return mid
    elif len(A) == 1: #O(1)
        return -1

    elif A[mid] < mid:
        return indiceEspejo(A,mid+1,high) #a=1,c=2
    elif A[mid] > mid:
        return indiceEspejo(A,low,mid-1) #a=1,c=2
```

Complejidad

$T(n) = 1T(\frac{n}{2}) + O(1)$, por teorema maestro $O(n^{\log_2 1}) = O(n^0) = O(1) \Rightarrow T(n) = O(n^{\log_2 1} \log n) = O(1 \log n) = O(\log n)$

ComplexityQuest

$T(n) = T(\frac{n}{2}) + n$
$f(n) = n = O(n), a = 1, c = 2$ luego,

$$O(n) = \Omega(n^{\log_2 1+\varepsilon}) \Rightarrow O(n^1) = \Omega(n^{0+\varepsilon}) \text{ tomo } \varepsilon = 1 \Rightarrow O(n^1) = \Omega(n^1)$$

$$\text{y tenemos que verificar que } af\left(\frac{n}{c}\right) = \frac{n}{2} < kf(n) \text{ para } k < 1$$

$$\text{lo cual es cierto, ya que } \frac{n}{2} < kn \text{ por ejemplo para } k = \frac{2}{3}:$$

$$\frac{1}{2}n < \frac{2}{3}n \wedge O(n) = \Omega(n^\varepsilon) \Rightarrow$$

$$T(n) = \Theta(f(n)) = \Theta(n)$$

$T(n) = 2T(\frac{n}{2}) + \log n$
$f(n) = \log n \in O(n^{\frac{1}{2}}), a = 2, c = 2$

$$f(n) = O(n^{\frac{1}{2}}) = O(n^{\log_2 2-\varepsilon}) = O(n^{1-\varepsilon})$$

$$\text{tomo } \varepsilon = \frac{1}{2} \Rightarrow f(n) = O(n^{\frac{1}{2}}) \Rightarrow$$

$$T(n) = \Theta(n^{\log_c a}) = \Theta(n)$$

SubBúsqueda

Algoritmo

```
def ubicar(A,e,low,high):

    mid = low + (high-low) // 2 #O(1)
    if len(A) == 1 and e not in A: #O(1)
        return False
    if A[mid] == e: #O(1)
        return True

    if aparece(A,low,mid,e): #O(raiz(mid-low+1)) = O(raiz(n/2))
        return ubicar(A,e,low,mid) #a=1, c=2
    else:
        return ubicar(A,e,mid+1,high)
```

Complejidad

$T(n) = aT(\frac{n}{c}) + f(n) = T(\frac{n}{2}) + O(\sqrt{\frac{n}{2}})$ o sea, $f(n) = O(n^{0,5}) \wedge O(n^{\log_c a}) = O(n^0)$ y tenemos por teorema maestro: $O(n^{0,5}) = O(n^{\log_2 1+\varepsilon}) = O(n^\varepsilon)$ y si tomo $\varepsilon = 0,5$:

$f(n) = O(n^{(\log_c a)+\varepsilon})$ y tenemos que ver que:

$af(\frac{n}{c})) > kf(n) \equiv \sqrt{n} > k\sqrt{\frac{n}{2}}, k < 1$ y esto es cierto $\forall k$

Concluimos que $T(n) = \Theta(\sqrt{n})$