```
Asmad Murga Victor Manuel
LU: 760/19
------------------------Ejercicio de Divide & Conquer-------------------------
Se por el enunciado que tenemos un arbol AVL. Por lo tanto, se que está balanceado el mismo
En los nodos tenemos valores enteros, positivos o negativos

Entonces, puedo plantearme lo siguiente. Un arbol binario tiene una raiz y tiene direccion
Izquierda y Derecha
Entonces, puedo plantearme una estrategia
Busco la maxima suma en la izquierda, busco la suma en la derecha
Pero pasa algo, que pasa si la suma es parte del medio, es decir, incluye a la raiz?
Entonces, tengo un tercer caso que es suma INCLUYENDO a la raiz

Ahora, pienso.
Caso Base:
  El arbol es nil. es decir, no tiene hijos

Caso Recursivo:
  Recorro ambas ramas, izquierda y derecha
  Por cada rama me quedo con la maxima suma

maximaSumaEnElArbol(in A: ab(nat)) -> res: (sumaC : nat, sumaCR : nat)
  if nil?(A) then                                   // O(1)
    return (0, 0)                                   // O(1)
  else
    dIzq <- maximaSumaEnElArbol(izq(A))             // T(n/2)
    dDer <- maximaSumaEnElArbol(der(A))             // T(n/2)
    Suma1 <- dIzq.sumaC                             // O(1)
    Suma2 <- dDer.sumaC         s                   // O(1)
    Suma3 <- raiz(A) + dIzq.sumaCR + dDer.sumaCR    // O(1)
    cam <- max(Suma1, Suma2, Suma3)
    desdeR <- max(0, raiz(A),
      raiz(A) + dIzq.sumaCR, raiz(A) + dDer.sumaCR) // O(1)
      return (cam, desdeR)
  endif

Con esto me es suficiente pues
 -> Tomo el arbol y divido en dos partes, izquierda y derecha. Esto es dividir la cantidad de nodos en 2 (n/2)
 -> Busco la maxima suma que pueda obtener a partir de ambas ramas
    => Esto es recursivo
 -> Tengo los 2 valores e inicio un tercer valor que sea la suma de los maximos por rama mas la raiz
    => De esta forma busco en que lado es el maximo
 -> Retorno el maximo entre las 3 sumas y la raiz inclusive, porque puede haber algun caso donde a izqueirda tenga un negativo y a derecha ningun nodo y estar balanceado
    Entonces en ese caso el maximo sería la raiz. Ej
            1
          /
        -1
    Aca el maximo es 1, no -1. Si hago la suma de ambos el resultado es 0 entonces el maximo es la raiz, a derecha tengo 0
    pues no hay nada

    Finalmente el algoritmo cumple con lo pedido

    Complejidad:
      Tengo c = 2 pues divido el arbol en dos, y a = 2 pues es la cantidad de particiones que voy a utilizar
      Calculo:
        Log_2 (2) = 1
      Ahora, mi f(n) es O(1) pues no tengo casos de conquista

      Finalmente tengo por la recurrencia:
                [ θ(1)              si n = 1 ]
        T(n) =  [ 2*T(n/2) + f(n)   si n > 1 ]

      Mi caso base se cumple en O(1)
      Luego, checkeo mi f(n) con O(1)
      f(n) < O(1)
      f(n) = θ(1)
      f(n) > Ω(1)

      Por el caso 1 del teorema maestro
        O(n^log_2(2) - epsilon) = O(n^1 - epsilon)
        Tomo epsilon = 1. Luego O(n¹ - ¹) = O(n°) = O(1)
        Finalmente f(n) ∈ O(1)
          (Esto porque es el caso donde el f(n) es menor que la parte recursiva,en este caso, es O(1))

      Finalmente, por el primer caso del teorema maestro
      T(n) = θ(n) como pedia el enunciado
```