Act 1

• Alg. iterativo que calcule la potencia de un ℤ.
Luego calc. su complejidad

```
⇒ PotIte(num base, num exp): num
    resultado ← 1                          c
    i ← 1                                  c
    si exp = 0 Entonces                    c
        devolver (1)                       c   no es el peor caso
    Mientras i <= exp hacer                c
        resultado ← resultado * base       2c (asignac, mult.)
        i ← i + 1                          2c (suma, asignac)
    devolver (resultado)
```

caso base → 4c

otro caso → $3c + \sum_{exp}^{n} (c+4c) + c = T(n)$

$\Rightarrow O(n)$

} un algoritmo de complejidad logarítmica

• Lo mismo pero recursivo

```
⇒ PotRecur(num base, num exp): num
    si exp = 0 Entonces
        devolver (1)
    valor ← base * PotRecur(base, exp-1)
    devolver (valor)
```

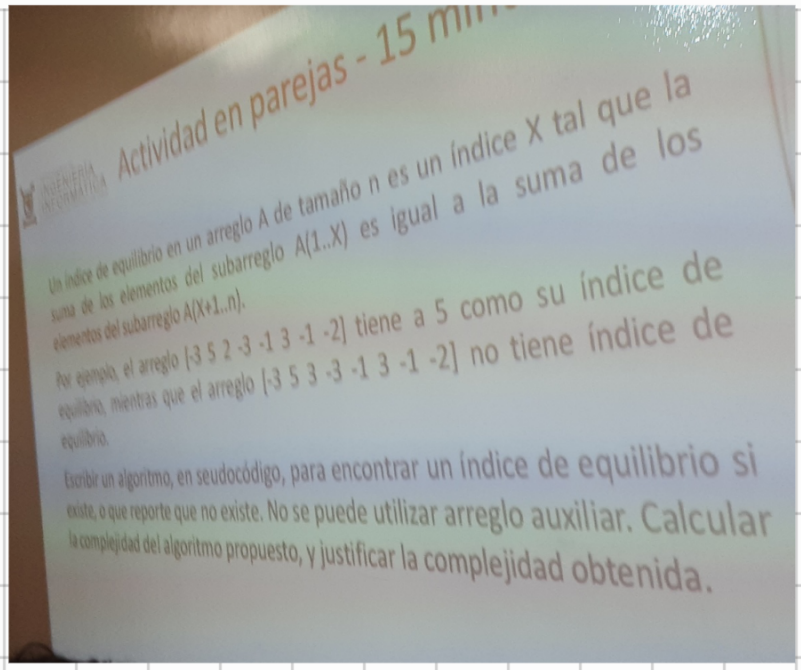
ecu de recurrencia → $t(n) = \begin{cases} 2c & n = 0 \\ 4c + t(n-1) & \text{, otro caso} \end{cases}$

Notamos $a = 1$, $b = 1$, $k = 0$

$\Rightarrow t(n) <= 1 \cdot t(n-1) + O(1) \Rightarrow O(n^{0+1}) = O(n)$

orden lineal

Act. 2



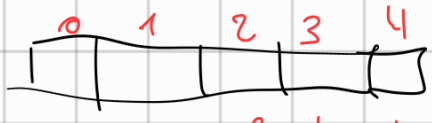
Actividad en parejas - 15 min

Un índice de equilibrio en un arreglo A de tamaño n es un índice X tal que la suma de los elementos del subarreglo A(1..X) es igual a la suma de los elementos del subarreglo A(X+1..n).

Por ejemplo, el arreglo [-3 5 2 -3 -1 3 -1 -2] tiene a 5 como su índice de equilibrio, mientras que el arreglo [-3 5 3 -3 -1 3 -1 -2] no tiene índice de equilibrio.

Escribir un algoritmo, en seudocódigo, para encontrar un índice de equilibrio si existe, o que reporte que no existe. No se puede utilizar arreglo auxiliar. Calcular la complejidad del algoritmo propuesto, y justificar la complejidad obtenida.

ver ppt
Más claro



5 elementos

complejidad

n^2

```
⇒ Alg(array A): num
    n ← obtener tamaño array
    Para i ← 1 hasta n
        suma1 ← 0
        suma2 ← 0
        para m ← 1 hasta n
            si m <= i
                suma1 ← suma1 + A[m]
            sino          .. caso m > i
                suma2 ← suma2 + A[m]
        si suma1 = suma2 entonces
            para t ← 1 hasta n
                si i = A[t] entonces
                    devolver(i)
            sino
                devolver(-1)
```

c idea
c ver sumas, si
3c es que existe,
c ver la posic,
2c y luego ver si
ese Nº de posic
en el array.

c
... el caso de
3c que ∃ sumas
c iguales, ahora
c buscamos si
c está ese num
en el array

$$T(n) = c + n(3c + 2c + n(6c) + c + n(5c))$$
$$= c + n(6c + 6cn + 5cn) = c + 6cn + 11cn^2$$
$$\Rightarrow O(n^2)$$ orden cuadrático.