

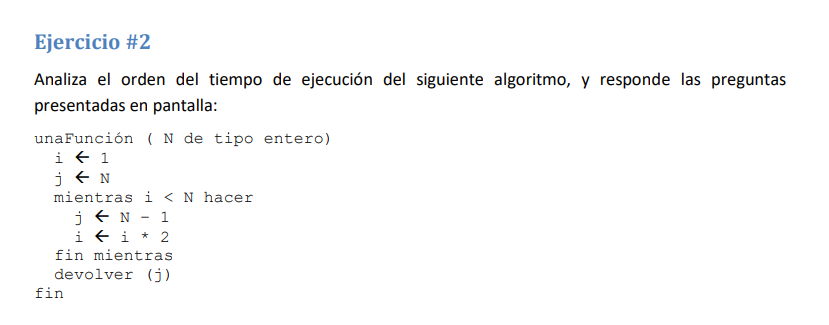


Orden del tiempo



O(1)+ O(n)+ O(1)+ O(1)+ O(1) = O(n) + 5 = O (n)

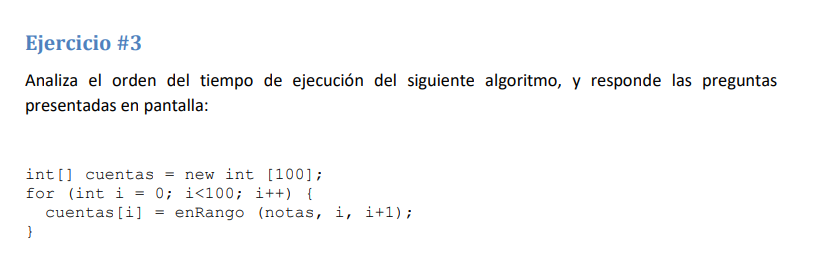
Resultado es el código es de O (n), al tener un orden n y sumarle variables constantes o asignaciones o condicionales que son de orden 1, tomo la orden de mayor valor para ver el tiempo de mi código.





En este caso tenemos que el código es todo de O (1) pero las operaciones dentro del bucle no son de tiempo constante, la cantidad de veces que el bucle se ejecuta es lo que determina la complejidad total.

El bucle se ejecuta log2 (N) veces, lo que significa que la complejidad total del bucle es O (log N)

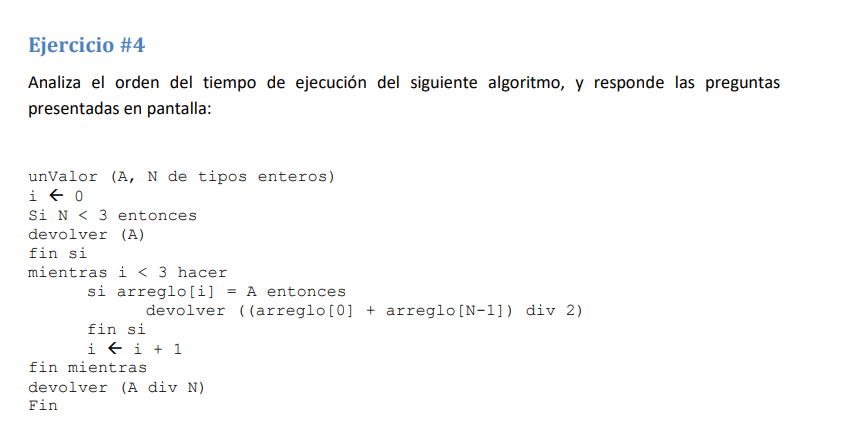




En la línea 3 cabe destacar que tengo si un crecimiento pero que es constante en + 1. La complejidad total del código es O (n), donde n es el número de iteraciones del bucle.

En el caso que I<100 (que el 100) no tuviera ese valor fijo, se lo podría tomar como una llamada de tiempo logartimo.

Entonces el orden de tiempo del código es O(n).





Este código es de tiempo constante O(1) ya que la cantidad de trabajo realizado no depende del tamaño del N, sino que está limitado a un número fijo de operaciones.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente



Como el arreglo se reduce a la mitad en cada iteración, el número de iteraciones es proporcional al logaritmo en base 2 del número de elementos. Por lo tanto es un algoritmo de búsqueda binaria y de orden = O(log N).

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente



En este código es de tiempo O (N log N ) porque tiene dos ciclos de nivel de recursión, aunque no están directamente relacionados, se multiplican para obtener la complejidad total d.e O (N log N)

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente



En este caso tenemos dos bucles anidadados de orden (n), la cual deja que el orden es de tipo n cuadrado.