|  |  |  |  |
| --- | --- | --- | --- |
| Logo, company name  Description automatically generated | Nombres: Juliana Alexandra Moreno Jiménez | | |
| Fecha: 21 de marzo de 2023 | Ciclo: 1 |
| **Estructura de Datos** | | |
| **Taller 10 – Cálculo de Complejidades** | | |

1. Para cada uno de los tiempos que toma un algoritmo en terminar, hallar el orden de complejidad correspondiente:
2. – Cúbica   
    - Lineal  
    - Constante

Complejidad = O ()

1. - Polinómica   
    - Raíz de n

– Constante

– Constante

Complejidad = O ()

– Cúbica logaritmo de n

– Cuadrática logaritmo de n

– Constante

Complejidad = O ()

1. Calcule la complejidad del siguiente algoritmo

**fun** XXXXXX(**n: Int**): Unit {

**var** x = 0**; 1**

**for (**i in 1 .. n step 5){ n/5 (1+2\*+n/2)(n/5)+1

var j = 1 1

while (j <= n) { (1+2\*+n/2)(n/5)

x = x + j; 1 2

j \*= 4 1 1+2\*+n/2

}

**for (k in n downto 1 step 2**){ n/2 n/2

x = x + 1; 1

}

}

}

– Cuadrática

– n logaritmo de n

- Lineal

– Constante

Complejidad = O ()

1. Responda las siguientes preguntas a partir del código de la siguiente función:

**fun** buscar(a: IList<Int>, elem: Int): Int {  
 **val** n: Int = a.**size 1  
 for** (i **in** 0 *until* n) { 1 1+1=2  
 **if** (a[i] == elem) { 1 1\*1  
 **return** i 1 =1  
 } 1  
 **else** {  
 **return** -1 1  
 }  
 }  
}

* 1. ¿Cuántas veces se repite el **for** que está dentro de la función?

El for que está dentro de la función buscar se repite 1 vez, para encontrar el elem que se pasa por i o determinar que no se encuentra en la lista de a. Aun así se ignoraría el segundo return ya que el primero ya esta dando por terminada la función.

* 1. A partir de la respuesta anterior, ¿cuál es la complejidad de la función **buscar** presentada previamente?

1+1

2 – Constante

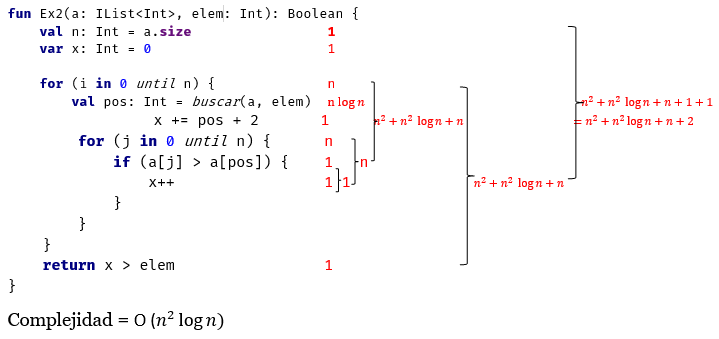
Complejidad = O(1)

1. Suponga que se tienen los siguientes dos algoritmos para resolver el mismo problema, y suponga que la función **buscar** tiene complejidad .

|  |  |
| --- | --- |
| **fun** Ex1(a: IList<Int>, elem: Int): Boolean {  **val** pos: Int = *buscar*(a, elem)  **val** n: Int = a.**size  var** x: Int = pos    **for** (i **in** 0 *until* n) {  x += 2  **for** (j **in** 0 *until* n) {  **if** (a[j] > a[pos]) {  x++  }  }  }  **return** x > elem } | **fun** Ex2(a: IList<Int>, elem: Int): Boolean {  **val** n: Int = a.**size  var** x: Int = 0   **for** (i **in** 0 *until* n) {  **val** pos: Int = *buscar*(a, elem)  x += pos + 2  **for** (j **in** 0 *until* n) {  **if** (a[j] > a[pos]) {  x++  }  }  }  **return** x > elem } |

Gráfico

Descripción generada automáticamente con confianza media



Indique con cuál de los dos algoritmos se queda para resolver el problema. Justifique muy bien su respuesta.

Me quedaría con el segundo algoritmo que tiene una complejidad de , el primer algoritmo tiene una complejidad de pero es el mejor es ya que en orden de complejidad va primero antes que .