

## Estructuras de Datos y Algoritmos

14-04-2025

## Mesa de Estudio 1

Autor: *Diego Banda* (diego.banda@mail.udp.cl)

## 1. Notación Big O

### 1.1. Cotas superiores

Para las siguientes funciones determine las cotas superiores utilizando notación Big O.

1.  $5n + 3$
2.  $4n^2 + 100n + 25$
3.  $2^n + n^3$
4.  $\log n + 100$
5.  $3n^3 + 2n \log n + 7$
6.  $n! + 2^n$
7.  $\sqrt{n} + \log n$
8.  $2^n + 3^n$
9.  $n + n^2 + n^3 + \dots n^k$
10.  $\frac{n^3}{1000} + 999n$

### 1.2. Algoritmos

Para los siguientes algoritmos, analice el tiempo de ejecución en términos de Big O ( $O(f(n))$ ).

1.  $f1$

```
1 public static void f1(int N){
2     for(int i = 0 ; i < N ; i++){
3         // Body
4     }
5 }
```

Listing 1: f1

2.  $f2$

```
1 public static void f2(){
2     for(int i = 0 ; i < 1024 ; i++){
3         // Body
4     }
5 }
```

Listing 2: f2

3.  $f3$

```
1 public static void f3(int N){
2     for(int i = 0 ; i < N*N ; i++){
3         // Body
4     }
5 }
```

```
4     }  
5 }
```

Listing 3: f3

4. Tiempo de ejecución del método *f5*

```
1 public static void f4(int N){  
2     for(int i = 0 ; i < N ; i++){  
3         System.out.println(i);  
4     }  
5 }  
6  
7 public static void f5(){  
8     for(int i = 0 ; i < 1000 ; i++){  
9         f4(i);  
10    }  
11 }
```

Listing 4: f4 y f5

5. *f6*

```
1 public static int f6(int N){  
2     if(N == 0) return 0;  
3     return f6(N-1) + 1;  
4 }
```

Listing 5: f6

6. *f7*

```
1 public static void f7(int N){  
2     if(N == 0) return 0;  
3     f7(N - 1);  
4     f7(N - 1);  
5 }
```

Listing 6: f7

7. *f8*

```
1 public static int f6(int N){  
2     if(N == 0) return 0;  
3     return f6(N / 2) + 1;  
4 }
```

Listing 7: f8

## 2. Compilador a papel

### 2.1. Algoritmo Misterioso: Postfix

Usted recibe una carta con un algoritmo misterioso en ella. Dado que usted es un experto en la materia, responderá las siguientes preguntas acerca de este misterioso algoritmo:

```
1 static int AlgoritmoMisterioso(String expr){
2     Stack <Integer> stack = new Stack<>();
3
4     for (int i = 0 ; i < expr.length() ; i++){
5         char c = expr.charAt(i);
6
7         if (c == ' '){
8             continue;
9         }
10
11        if (Character.isDigit(c)){
12            stack.push(c - '0');
13        }
14        else {
15            int v1 = stack.pop();
16            int v2 = stack.pop();
17
18            switch (c){
19                case '+':
20                    stack.push(v2 + v1);
21                    break;
22                case '-':
23                    stack.push(v2 - v1);
24                    break;
25                case '/':
26                    stack.push(v2 / v1);
27                    break;
28                case '*':
29                    stack.push(v2 * v1);
30                    break;
31            }
32        }
33    }
34    return stack.pop();
35 }
```

Listing 8: Algoritmo Misterioso

**Pista:** La expresión  $c - '0'$  tiene como resultado el valor del número  $c$ . Ejemplos:  $'0' - '0' = 0$ ,  $'4' - '0' = 4$ .

- Para cada uno de los siguientes casos de prueba, indique el valor retornado por el método *AlgoritmoMisterioso* al recibir el caso como argumento:
  - 5 5 +
  - 6 6 / 9 \*
  - 2 4 \* 2 2 \* +
- En base a sus observaciones del punto anterior, explique en no más de **3 líneas**, ¿De qué forma el algoritmo modifica los datos que recibe?
- Analice el algoritmo en cuestión y describa su tiempo de ejecución en términos de  **$O(f(n))$** , donde  **$n$**  es el largo del arreglo recibido como argumento y  **$f$**  es una función matemática propuesta por usted. Fundamente su respuesta.

## 2.2. Algoritmo Misterioso 2

Se define una lista enlazada de la siguiente manera:

```
1  class linkedList {  
2      class Node {  
3          int value;  
4          Node next;  
5      }  
6      Node head;  
7  }
```

Listing 9: Implementación LinkedList

Analice con detenimiento el siguiente algoritmo y responda las siguientes preguntas.

```
1  static Node algoritmoMisterioso(Node head){  
2      if(head == null || head.next == null)  
3          return head;  
4  
5      Node temp = algoritmoMisterioso(head.next);  
6  
7      head.next.next = head;  
8  
9      head.next = null;  
10  
11     return temp;  
12 }
```

Listing 10: Algoritmo Misterioso 2

1. Calcule el output del algoritmo para la siguiente lista enlazada.
  - LL: 1 -> 7 -> 10 -> -8 -> 666 -> null
2. Describa la complejidad de tiempo del algoritmo utilizando  $O(f(n))$ , argumente su respuesta de manera breve.

### 3. A programar

#### 3.1. Suma de dos listas

Se define una lista enlazada de la siguiente manera:

```
1  class linkedList {  
2      class Node {  
3          int value;  
4          Node next;  
5      }  
6      Node head;  
7  }
```

Listing 11: Implementación LinkedList

Considere el uso de esta lista enlazada para representar números enteros: con un dígito por nodo, en orden inverso de los dígitos (el último nodo representa el dígito más significativo). Por ejemplo, la lista 2 -> 9 -> null representa el número 92. La lista 1 -> 3 -> 9 -> null representa el número 931. Considerando lo anterior se le solicita que:

1. Defina un método *suma*, el cual recibe como argumentos los head de dos listas que representan números y retorna el head de una nueva lista, la cual representa la suma de los dos números recibidos. Como ejemplo:

■ Caso 1:

Input:

- L1: 8 -> 3 -> 2 -> null
- L2: 3 -> 4 -> 3 -> null

Output:

- suma: 1 -> 8 -> 5 -> null

■ Caso 2:

Input:

- L1: 7 -> 1 -> 9 -> null
- L2: 5 -> 9 -> null

Output:

- suma: 2 -> 1 -> 0 -> 1 -> null

2. Analice el algoritmo diseñado y describa su tiempo de ejecución en términos de  $O(f(n, m))$ , donde  $n$  y  $m$  son la cantidad de datos en la primera y segunda lista, respectivamente, y  $f$  es una función matemática propuesta por usted. Fundamente su respuesta.

#### 3.2. Eliminar duplicados de una lista enlazada ordenada

Se define una lista enlazada de la siguiente manera:

```
1  class linkedList {  
2      class Node {  
3          int value;  
4          Node next;  
5      }  
6      Node head;  
7  }
```

Listing 12: Implementación LinkedList

Usted es un apasionado coleccionista que le gusta tener todo ordenado y sin repetir nada, su colección es de una lista enlazada, la cual siempre tiene números ordenados pero a veces se tienen duplicados sin darse cuenta. Se le pide a usted todo un experto en listas enlazadas, que realice un algoritmo que recibe el head de una lista enlazada ordenada y elimina los duplicados.

1. Programe una función llamada *eliminarRepetidos* que recibe el head de una lista y retorna la lista sin duplicados. Como ejemplo:
  - Caso 1:  
Input:
    - Head: 1 -> 2 -> 2 -> 3 -> 4 -> nullOutput:
    - sinDuplicados: 1 -> 2 -> 3 -> 4 -> null
  - Caso 2:  
Input:
    - Head: 1 -> 1 -> 1 -> 1 -> nullOutput:
    - sinDuplicados: 1 -> null
2. Determine el tiempo de ejecución utilizando la notación  $O(f(n))$ .