



## INSTITUTO POLITÉCNICO NACIONAL ESCUELA SUPERIOR DE CÓMPUTO

# Ejercicio 04: Análisis de algoritmos no recursivos

Unidad de aprendizaje: Análisis de Algoritmos

Grupo: 3CM3

Alumno:
Ramos Diaz Enrique

Profesor(a): Franco Martínez Edgardo Adrián



9 de Octubre 2018

# Índice

| 1 | Para | los siguientes | 12 | a | lg | ori | itn | nc | S | de | ete | er | m | in | e | la | c | ota | a ( | 0 | 0 |  |  |  |  |  |  |  |  |  |
|---|------|----------------|----|---|----|-----|-----|----|---|----|-----|----|---|----|---|----|---|-----|-----|---|---|--|--|--|--|--|--|--|--|--|
|   |      | Algoritmo 1 .  |    |   |    |     |     |    |   |    |     |    |   |    |   |    |   |     |     |   |   |  |  |  |  |  |  |  |  |  |
|   |      | Algoritmo 2 .  |    |   |    |     |     |    |   |    |     |    |   |    |   |    |   |     |     |   |   |  |  |  |  |  |  |  |  |  |
|   |      | Algoritmo 3 .  |    |   |    |     |     |    |   |    |     |    |   |    |   |    |   |     |     |   |   |  |  |  |  |  |  |  |  |  |
|   | 1.4  | Algoritmo 4 .  |    |   |    |     |     |    |   |    |     |    |   |    |   |    |   |     |     |   |   |  |  |  |  |  |  |  |  |  |
|   |      | Algoritmo 5 .  |    |   |    |     |     |    |   |    |     |    |   |    |   |    |   |     |     |   |   |  |  |  |  |  |  |  |  |  |
|   | 1.6  | Algoritmo 9 .  |    |   |    |     |     |    |   |    |     |    |   |    |   |    |   |     |     |   |   |  |  |  |  |  |  |  |  |  |
|   | 1.7  | Algoritmo 10   |    |   |    |     |     |    |   |    |     |    |   |    |   |    |   |     |     |   |   |  |  |  |  |  |  |  |  |  |
|   | 1.8  | Algoritmo 11   |    |   |    |     |     |    |   |    |     |    |   |    |   |    |   |     |     |   |   |  |  |  |  |  |  |  |  |  |
|   | 1.9  | Algoritmo 12   |    |   |    |     |     |    |   |    |     |    |   |    |   |    |   |     |     |   |   |  |  |  |  |  |  |  |  |  |
|   | 1.10 | Algoritmo 13   |    |   |    |     |     |    |   |    |     |    |   |    |   |    |   |     |     |   |   |  |  |  |  |  |  |  |  |  |
|   | 1.11 | Algoritmo 14   |    |   |    |     |     |    |   |    |     |    |   |    |   |    |   |     |     |   |   |  |  |  |  |  |  |  |  |  |
|   | 1.12 | Algoritmo 15   |    |   |    |     |     |    |   |    |     |    |   |    |   |    |   |     |     |   |   |  |  |  |  |  |  |  |  |  |

## 1. Para los siguientes 12 algoritmos determine la cota O()

Operaciones básicas para el análisis de la complejidad temporal: asignación, aritméticas, condicionales y saltos implícitos.

## 1.1. Algoritmo 1

#### Análisis de Cotas

```
Se repite n-1 veces

for (i=1; i<n; i++)

O(n)

Se repite n-1 veces

for (j=0; j<n-1; j++)

O(n)

O(n)
```

#### Análisis de complejidad temporal

Observamos que el algoritmo esta conformado principalmente por dos ciclos for anidados.

```
La función temporal es f_t(n) = 8n^2 - 10n + 5
```

La función temporal del algoritmo obtenida en análisis previo del Ejercicio 2 corresponde al orden de complejidad obtenido en el análisis de cotas  $O(n^2)$ .

## 1.2. Algoritmo 2

#### Análisis de Cotas

#### Análisis de complejidad temporal

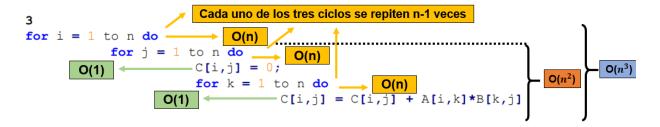
El algoritmo consiste de un unico for, y dentro de él existen par de asignaciones y operaciones aritméticas.

La función temporal es  $f_t(n) = 7n + 11$ 

La función temporal del algoritmo obtenida en análisis previo del Ejercicio 02 corresponde al orden de complejidad obtenido en el análisis de cotas O(n).

## 1.3. Algoritmo 3

#### Análisis de Cotas



#### Análisis de complejidad temporal

Existen 3 ciclos for anidados en este algoritmo, en donde la mayor parte de asignaciones y operaciones aritméticas entre los arreglos A, B y C ocurren en el tercer ciclo.

```
La función temporal es f_t(n) = 6n^3 + 5n^2 + 7n + 3
```

La función temporal del algoritmo obtenida en análisis previo del Ejercicio 2 corresponde al orden de complejidad obtenido en el análisis de cotas  $O(n^3)$ .

## 1.4. Algoritmo 4

#### Análisis de Cotas

```
anterior = 1; O(1)
actual = 1; O(1)
while (n>2) \rightarrow O(n)
{
    Se repite n-2 veces

    aux = anterior + actual; \rightarrow O(1)
    anterior = actual; \rightarrow O(1)
    actual = aux; \rightarrow O(1)
    n = n - 1; \rightarrow O(1)
}
```

#### Análisis de complejidad temporal

En este algoritmo únicamente existen un único ciclo while, en donde se hace la mayoría de asignaciones y operaciones matemáticas.

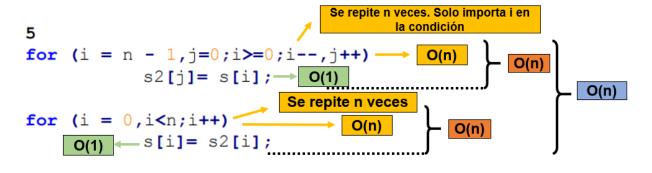
3

La función temporal es 
$$f_t(n) = \begin{cases} 4 & \text{si } n \leq 2\\ 8n - 12 & \text{si } n > 2 \end{cases}$$

La función temporal del algoritmo (siendo el caso n>2) obtenida en análisis previo del Ejercicio02 corresponde al orden de complejidad obtenido en el análisis de cotas O(n).

### 1.5. Algoritmo 5

#### Análisis de Cotas



#### Análisis de complejidad temporal

Similar a algoritmos anteriores, existen dos ciclos for. Sin embargo, estos no están anidados, sino son independientes uno de otro. Pareciera que la complejidad de este algoritmo esta en el orden de  $n^2$ , pero esto es incorrecto.

La función temporal es 
$$f_t(n) = 9n + 8$$

La función temporal del algoritmo obtenida en análisis previo del Ejercicio 02 corresponde al orden de complejidad obtenido en el análisis de cotas O(n).

## 1.6. Algoritmo 9

#### Análisis de Cotas

```
func Producto2Mayores (A,n)
if(A[1] > A[2])
           mayor1 = A[1];
           mayor2 = A[2];
                                        O(1)
           mayor2 = A[1];
                                                                        O(n)
while(i<=n)</pre>
           if(A[i] > mayor1)
                 O(1) mayor2 = mayor1;
                       mayor1 = A[i]; O(1)
                                                       0(1)
           else if (A[i] > mayor2)
                       mayor2 = A[i]; O(1)
return = mayor1 * mayor2; _
fin
```

#### Análisis de complejidad temporal

```
Mejor Caso f_{tmc}(n) = 3 + 2(n-2)
Peor Caso f_{tpc}(n) = 3 + 3(n-2)
Caso Medio f_{tcm}(n) = \frac{8}{3}n - \frac{7}{3}
```

La función temporal del algoritmo obtenida en análisis previo del Ejercicio 02 corresponde al orden de complejidad obtenido en el análisis de cotas O(n) (para los tres casos).

## 1.7. Algoritmo 10

#### Análisis de Cotas

#### Análisis de complejidad temporal

```
Mejor Caso f_{tmc}(n) = (n-1)^2
Peor Caso f_{tpc}(n) = 4(n-1)^2
Caso Medio f_{tcm}(n) = \frac{5}{2}(n-1)^2
```

La función temporal del algoritmo obtenida en análisis previo del Ejercicio 2 corresponde al orden de complejidad obtenido en el análisis de cotas  $O(n^2)$  (para los tres casos).

## 1.8. Algoritmo 11

#### Análisis de Cotas

```
11
func MaximoComunDivisor (m, n)
            a=max(n,m);
                                      Se repite \log_a b veces
                             O(1)
            b=min(n,m);
                             O(1)
                                         O(\log_a b)
            residuo=1; -
            mientras (residuo > 0)
                                                           O(\log_a b)
                   O(1) residuo=a mod b;
                                           _ m * O(1)
                   O(1) b=residuo;
            MaximoComunDivisor=a;
            return MaximoComunDivisor; — O(1)
}
```

## 1.9. Algoritmo 12

#### Análisis de Cotas

```
Procedimiento BurbujaOptimizada(A,n)
           cambios = "No" O(1)
                                                      Se repite n-1 veces
           i=0 O(1)
           Mientras i < n-1 && cambios != "No" hacer -
                                                         → O(n)
                      cambios = "No" O(1)
                                                     Se repite n-1-i veces
                      Para j=0 hasta (n-2)-i hacer
                            O(1) Si(A[i] < A[j]) hacer
                                                                                O(n^2)
                                        O(1) aux = A[j]
                                        O(1) A[j] = A[i]
                                                               O(1)
                                                                       O(n)
                                        O(1) A[i] = aux
                                        O(1) cambios = "Si"
                                  FinSi
                               ......
                       i= i+1 O(1)
fin Procedimiento
```

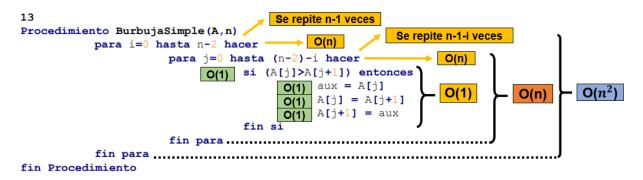
#### Análisis de complejidad temporal

Mejor Caso 
$$f_{tmc}(n) = n + \frac{n(n-1)^2}{2}$$
  
Peor Caso  $f_{tpc}(n) = n + \frac{5}{2}n(n-1)^2$   
Caso Medio  $f_{tcm}(n) = n + \frac{3}{2}n(n-1)^2$ 

La función temporal del algoritmo obtenida en análisis previo del Ejercicio 20 NO corresponde al orden de complejidad obtenido en el análisis de cotas  $O(n^2)$ , siendo de orden  $n^3$ .

## **1.10.** Algoritmo 13

#### Análisis de Cotas



#### Análisis de complejidad temporal

Mejor Caso 
$$f_{tmc}(n) = \frac{n(n-1)^2}{2}$$
  
Peor Caso  $f_{tpc}(n) = 2n(n-1)^2$   
Caso Medio  $f_{tcm}(n) = 5n(n-1)^2$ 

La función temporal del algoritmo obtenida en análisis previo del Ejercicio 20 NO corresponde al orden de complejidad obtenido en el análisis de cotas  $O(n^2)$ , siendo de orden  $n^3$ .

## **1.11.** Algoritmo 14

#### Análisis de Cotas

#### Análisis de complejidad temporal

```
Mejor Caso f_{tmc}(n) = 2
Peor Caso f_{tpc}(n) = 3
Caso Medio f_{tcm}(n) = 2.66
```

La función temporal del algoritmo obtenida en análisis previo del Ejercicio 02 corresponde al orden de complejidad obtenido en el análisis de cotas O(1) (para los tres casos).

## **1.12.** Algoritmo 15

#### Análisis de Cotas

```
Procedimiento Seleccion (A,n)

para k=0 hasta n-2 hacer O(n)

p=k O(1)

para i=k+1 hasta n-1 hacer O(n)

para i=k+1 hasta n-1 hacer O(n)

si A[i]<A[p] entonces O(1)

p=i O(1)

fin para

temp = A[p]O(1)

A[p] = A[k]O(1)

A[k] = temp O(1)

fin para

fin Procedimiento
```

## Análisis de complejidad temporal

```
Mejor Caso f_{tmc}(n) = 2n(n-1)^2 + 4(n-1)
Peor Caso f_{tpc}(n) = 3n(n-1)^2 + 4(n-1)
Caso Medio f_{tcm}(n) = \frac{5}{2}n(n-1)^2 + 4(n-1)
```

La función temporal del algoritmo obtenida en análisis previo del Ejercicio02 NO corresponde al orden de complejidad obtenido en el análisis de cotas  $O(n^2)$ , siendo de orden  $n^3$ .