



Escuela  
Superior  
de Cómputo

INSTITUTO POLITÉCNICO NACIONAL  
ESCUELA SUPERIOR DE CÓMPUTO

## Ejercicio 02: Complejidad temporal y análisis de casos

Unidad de aprendizaje: Análisis de Algoritmos

Grupo: 3CM3

*Alumno:*

Ramos Diaz Enrique

*Profesor(a):*

Franco Martínez Edgardo Adrián



6 de Septiembre 2018

# Índice

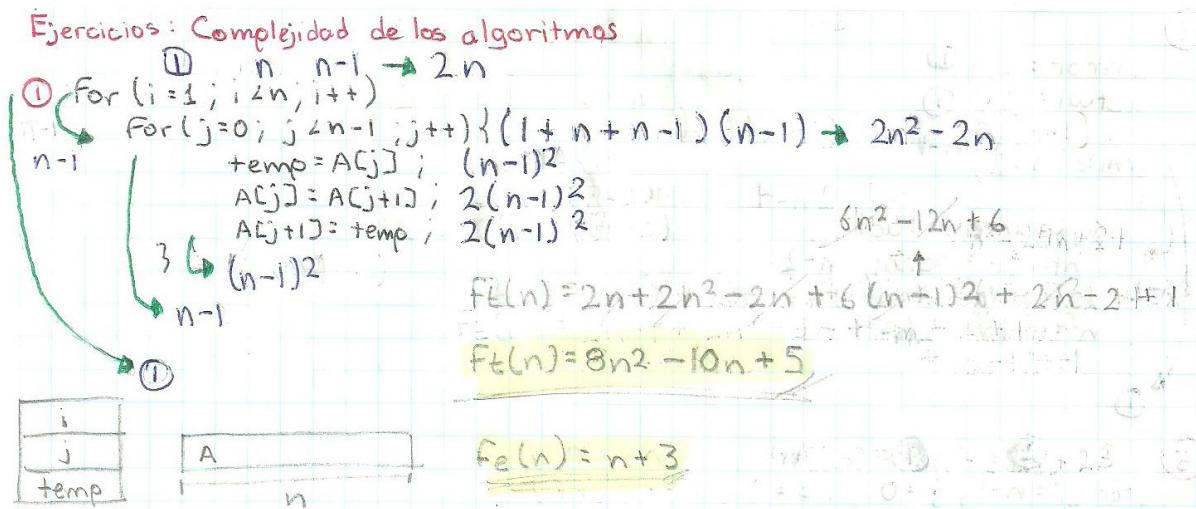
<b>1 Para los siguientes 5 algoritmos determine la función de complejidad temporal y espacial en términos de n . . . . .</b>	<b>2</b>
1.1 Algoritmo 1 . . . . .	2
1.2 Algoritmo 2 . . . . .	3
1.3 Algoritmo 3 . . . . .	4
1.4 Algoritmo 4 . . . . .	5
1.5 Algoritmo 5 . . . . .	7
<b>2 Para los siguientes 3 algoritmos determine el número de veces que se imprime la palabra “Algoritmos”. Determine una función lo más cercana a su comportamiento para cualquier n y compruébelo codificando los tres algoritmos (Adjuntar códigos). De una tabla de comparación de sus pruebas para n’s igual a 10, 100, 1000, 5000 y 100000 y demostrar lo que la función encontrada determina será el número de impresiones. . . . .</b>	<b>8</b>
2.1 Algoritmo 6 . . . . .	8
2.2 Algoritmo 7 . . . . .	10
2.3 Algoritmo 8 . . . . .	12
<b>3 Para los siguientes algoritmos determine las funciones de complejidad temporal, para el mejor caso, peor caso y caso medio. Indique cual(es) son las condiciones (instancia de entrada) del peor caso y cual(es) la del mejor caso. . . . .</b>	<b>13</b>
3.1 Algoritmo 9 . . . . .	13
3.2 Algoritmo 10 . . . . .	14
3.3 Algoritmo 12 . . . . .	15
3.4 Algoritmo 13 . . . . .	16
3.5 Algoritmo 14 . . . . .	17
3.6 Algoritmo 15 . . . . .	18

# 1. Para los siguientes 5 algoritmos determine la función de complejidad temporal y espacial en términos de n

Operaciones básicas: asignación, aritméticas, condicionales y saltos implícitos.

## 1.1. Algoritmo 1

### Análisis



Observamos que el algoritmo esta conformado principalmente por dos ciclos for anidados, de entrada esto nos dice que la complejidad es del orden de  $n^2$ .

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

La función espacial es  $f_e(n) = n + 3$

## Codificación y pruebas con n = 10

```
#define n 10

void Ej1(){
    printf("*****Ejercicio 1*****\n");
    int temp, A[n], cont=0;

    cont++; //asig i
    cont++; //ultima comparacion donde la condicion no se cumple y sale
    for(int i=1; i<n; i++, cont++){
        cont++; //asig j
        cont++; //ultima comparacion donde la condicion no se cumple y sale
        for(int j=0; j<(n-1); j++, cont++){
            temp=A[j]; cont++;
            A[j]=A[j+1]; cont++; cont++;
            A[j+1]=temp; cont++; cont++;
            cont++; //Numero de comparaciones
            cont++; //Numero de saltos
        }
        cont++; //Sale del for de adentro y reinicia

        cont++; //Numero de comparaciones
        cont++; //Numero de saltos

    }
    cont++;

    printf("Operaciones reales: %d\n", cont);
    printf("Operaciones segun la formula: %d\n", 8*n*n - 10*n + 5);
}

enrique@enrike: /media/Google_Drive/5º SEMESTRE/Analisis de al... ⓘ ⓘ ⓘ
Archivo Editar Ver Buscar Terminal Ayuda
oritmos$ gcc x.c -o x
enrique@enrike:/media/Google_Drive/5º SEMESTRE/Analisis de al
oritmos$ ./x
*****Ejercicio 1*****
Operaciones reales: 705
Operaciones segun la formula: 705
enrique@enrike:/media/Google_Drive/5º SEMESTRE/Analisis de al
oritmos$
```

## 1.2. Algoritmo 2

## Análisis

② polinomio = 0; ①  
 For(i=0; i <= n; i++)  
n+1      ①  
polinomio = polinomio \* 2 + A[n-i]; 4(n+1)  
n+1  
①

$F_f(n) = 2 + 2n + 3 + 4n + 4 + n + 1 + 1$   
 $F_f(n) = 7n + 11$   
 $F_f(n) = n + 3$

El algoritmo consiste de un único `for`, y dentro de él existen par de asignaciones y operaciones aritméticas. Esto nos indica que la complejidad será de orden  $n$ .

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

La función espacial es  $f_e(n) = n + 3$

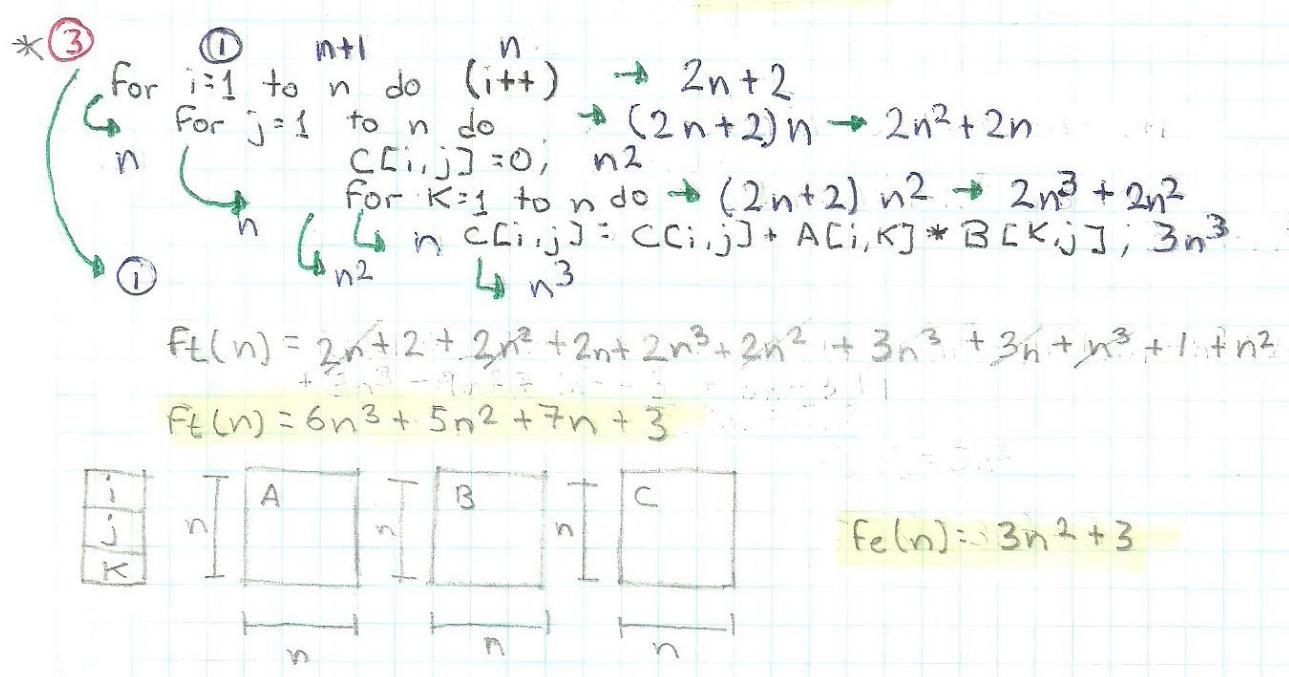
## Codificación y pruebas con $n = 10$

```
// n = 10
void Ej2(){
    printf("****Ejercicio 2*****\n");
    int cont=0;
    cont++; //asig polinomio //polinomio=0;
    cont++; //asig i
    cont++; //ultima comparacion donde la condicion no se cumple
    //y sale
    for (int i = 0; i <= n; ++i, cont++) { //Numero de i++
        cont+=4; //polinomio = polinomio*z + A[n-1];
        cont++; //Numero de comparaciones
        cont++; //Numero de saltos
    }
    cont++;
    printf("Operaciones reales: %d\n", cont);
    printf("Operaciones segun la formula: %d\n", 7*n + 11);
}
```

enrike@enrike: /media/Google\_Drive/5º SEMESTRE/Analisis de al... ● + ×  
Archivo Editar Ver Buscar Terminal Ayuda  
oritmos\$ gcc x.c -o x  
enrike@enrike:/media/Google\_Drive/5º SEMESTRE/Analisis de al  
oritmos\$ ./x  
\*\*\*\*Ejercicio 2\*\*\*\*\*  
Operaciones reales: 81  
Operaciones segun la formula: 81  
enrike@enrike:/media/Google\_Drive/5º SEMESTRE/Analisis de al  
oritmos\$

## 1.3. Algoritmo 3

### Análisis



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 complejidad será de orden  $n^3$ .

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

La función espacial es  $f_e(n) = 3n^2 + 3$

### Codificación y pruebas con n = 3

```
// n = 4
void Ej3(){
    printf("*****Ejercicio 3*****\n");
    int cont=0, i, j, k;

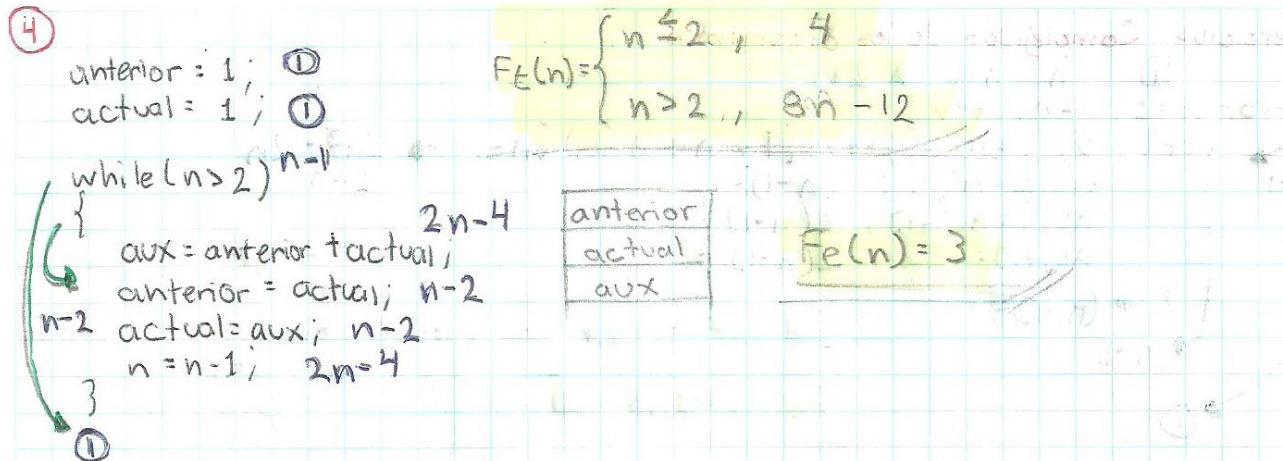
    cont++; //asig i
    cont++; //ultima comparacion donde la condicion no se cumple
    //y sale
    for(i=1; i<=n; i++, cont++){
        for(j=1; j<=n; j++, cont++){
            cont++; //C[i,j]=0;
            cont++; //asig k
            cont++; //ultima comparacion donde la condicion no
            //se cumple y sale
            for(k=1; k<=n; k++, cont++){
                cont+=3; //C[i,j]=C[i,j]+A[i,k]*B[k,j];
                cont++; //Numero de comparaciones
                cont++; //Numero de saltos
            }
            cont++; //Numero de comparaciones
            cont++; //Numero de saltos
        }
        cont++; //Numero de comparaciones
        cont++; //Numero de saltos
    }
    cont++; //Sale del for
    printf("Operaciones reales: %d\n", cont);
    printf("Operaciones segun la formula: %d\n", 6*n*n*n + 5*n*n + 7*n + 3);
}
```

enrike@enrike: /media/Google\_Drive/5° SEMESTRE/Analisis de al... Archivo Editar Ver Buscar Terminal Ayuda  
oritmos\$ gcc x.c -o x  
enrike@enrike:/media/Google\_Drive/5° SEMESTRE/Analisis de al oritmos\$ ./x  
\*\*\*\*\*Ejercicio 3\*\*\*\*\*  
Operaciones reales: 495  
Operaciones segun la formula: 495  
enrike@enrike:/media/Google\_Drive/5° SEMESTRE/Analisis de al oritmos\$

### 1.4. Algoritmo 4

#### Análisis

$$f_t(n) = 1 + 1 + n-1 + 2n-4 + n-2 + n-2 + 2n-11 + n-2 + 1$$



En este algoritmo únicamente existen un único ciclo while, en donde se hace la mayoría de asignaciones y operaciones matemáticas. Nos da a entender que la complejidad será de orden  $n$ .

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 espacial es  $f_e(n) = 3$

## Codificación y pruebas con $n \leq 2$

```
// n = 2
void Ej4(){
    printf("****Ejercicio 4*****\n");
    int cont=0, aux;

    int n1 = n;

    int anterior=1; cont++;
    int actual=1; cont++;

    cont++; // ultima comparacion donde la condicion
    // no se cumple y sale
    while(n1>2){
        cont++; // Numero de comparaciones
        cont++; // Numero de saltos
        aux=anterior+actual; cont+=2;
        anterior=actual; cont++;
        actual=aux; cont++;
        n1 = n1-1; cont+=2;
    }
    cont++; // Sale del while
    printf("Operaciones reales: %d\n", cont);

    if(n<=2) printf("Operaciones segun la formula: %d\n", 4);
    else printf("Operaciones segun la formula: %d\n", 8*n - 12);
}
```

enrique@enrique:/media/Google\_Drive/5º SEMESTRE/Analisis de al... Archivo Editar Ver Buscar Terminal Ayuda  
oritmos\$ gcc x.c -o x  
enrique@enrike:/media/Google\_Drive/5º SEMESTRE/Analisis de al...  
oritmos\$ ./x  
\*\*\*\*Ejercicio 4\*\*\*\*\*  
Operaciones reales: 4  
Operaciones segun la formula: 4  
enrique@enrike:/media/Google\_Drive/5º SEMESTRE/Analisis de al...  
oritmos\$

## Codificación y pruebas con $n > 2$

```
// n = 5
void Ej4(){
    printf("****Ejercicio 4*****\n");
    int cont=0, aux;

    int n1 = n;

    int anterior=1; cont++;
    int actual=1; cont++;

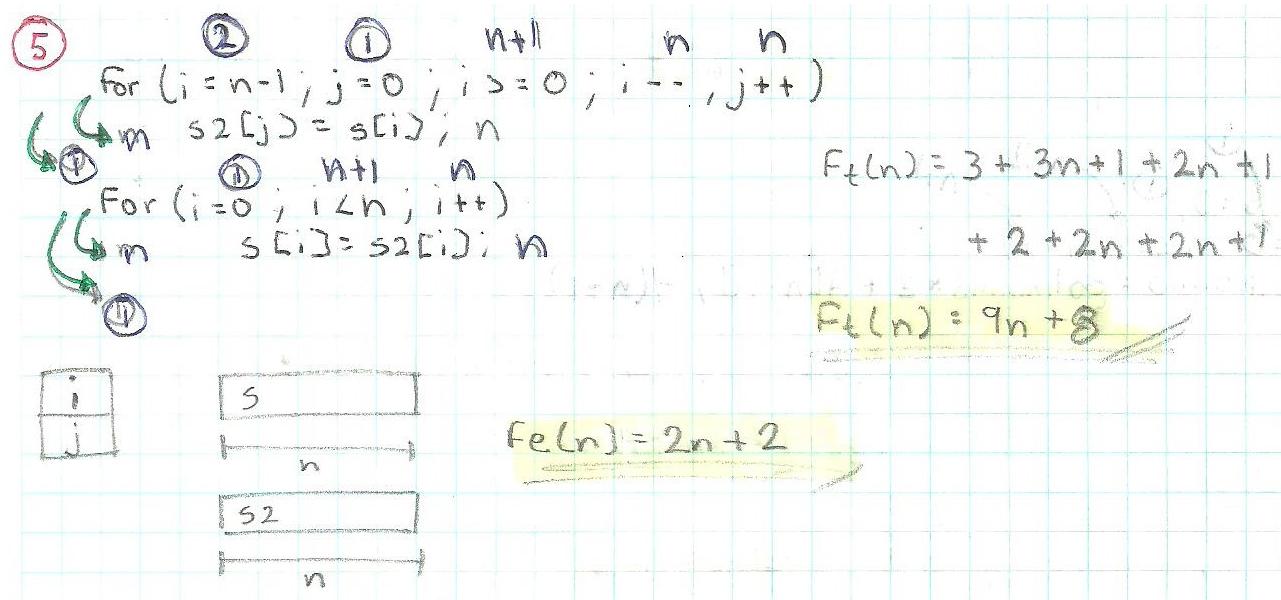
    cont++; // ultima comparacion donde la condicion
    // no se cumple y sale
    while(n1>2){
        cont++; // Numero de comparaciones
        cont++; // Numero de saltos
        aux=anterior+actual; cont+=2;
        anterior=actual; cont++;
        actual=aux; cont++;
        n1 = n1-1; cont+=2;
    }
    cont++; // Sale del while
    printf("Operaciones reales: %d\n", cont);

    if(n<=2) printf("Operaciones segun la formula: %d\n", 4);
    else printf("Operaciones segun la formula: %d\n", 8*n - 12);
}
```

enrique@enrike:/media/Google\_Drive/5º SEMESTRE/Analisis de al... Archivo Editar Ver Buscar Terminal Ayuda  
oritmos\$ gcc x.c -o x  
enrique@enrike:/media/Google\_Drive/5º SEMESTRE/Analisis de al...  
oritmos\$ ./x  
\*\*\*\*Ejercicio 4\*\*\*\*\*  
Operaciones reales: 28  
Operaciones segun la formula: 28  
enrique@enrike:/media/Google\_Drive/5º SEMESTRE/Analisis de al...  
oritmos\$

## 1.5. Algoritmo 5

### Análisis



Similar a algoritmos anteriores, existen dos ciclos for. Sin embargo, estos no estan 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 complejidad simplemente es del orden de  $n$ .

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

La función espacial es  $f_e(n) = 2n + 2$

### Codificación y pruebas con $n = 8$

```
// n = 8
void Ej5(){
    printf("*****Ejercicio 5*****\n");
    int s2[n], s[n], i, j, cont=0;
    cont+=2; //asig i y n-1
    cont++; //asig j
    cont++; //ultima comparacion donde la condicion
    //no se cumple y sale
    for(i=n-1, j=0; i>=0; i--, cont+=2, j++){
        s2[j]=s[i]; cont++;
        cont++; //Numero de comparaciones
        cont++; //Numero de saltos
    }
    cont++;

    cont++; //asig i
    cont++; //ultima comparacion donde la condicion
    //no se cumple y sale
    for(i=0; i<n; i++, cont++){
        s[i]=s2[i]; cont++;
        cont++; //Numero de comparaciones
        cont++; //Numero de saltos
    }
    cont++;

    printf("Operaciones reales: %d\n", cont);
    printf("Operaciones segun la formula: %d\n", 9*n + 8);
}
```

The terminal window shows the execution of the code. It starts with the command `gcc x.c -o x`, then runs the executable `x`. The output displays the results of the algorithm's execution and compares them with the formula  $9n + 8$ .

```
enrike@enrike:/media/Google_Drive/5º SEMESTRE/Analisis de al... ● + ×
Archivo Editar Ver Buscar Terminal Ayuda
oritmos$ gcc x.c -o x
enrike@enrike:/media/Google_Drive/5º SEMESTRE/Analisis de al... ● + ×
oritmos$ ./x
*****Ejercicio 5*****
Operaciones reales: 80
Operaciones segun la formula: 80
enrike@enrike:/media/Google_Drive/5º SEMESTRE/Analisis de al... ● + ×
oritmos$
```

**2. Para los siguientes 3 algoritmos determine el número de veces que se imprime la palabra “Algoritmos”. Determine una función lo más cercana a su comportamiento para cualquier  $n$  y compruébelo codificando los tres algoritmos (Adjuntar códigos). De una tabla de comparación de sus pruebas para  $n$ 's igual a 10, 100, 1000, 5000 y 100000 y demostrar lo que la función encontrada determina será el número de impresiones.**

### 2.1. Algoritmo 6

#### Análisis

$n$	# Impresiones Reales	# Impresiones Función
10	3	3
100	6	6
1000	9	9

A continuación se observa que la función es  $f(n) = \# \text{Algoritmos}$

⑥ For ( $i=10$ ;  $i < n*5$ ;  $i+=2$ )  
 printf("Algoritmos\n");

$F(n) = \log_2(n)$

$f(-1)=0$        $f(4)=1$   
 $f(0)=0$        $f(5)=2$   
 $f(1)=0$        $f(6)=2$   
 $f(2)=0$        $f(7)=3$   
 $f(3)=1$        $f(8)=3$

#### Resultados

$n$	# Impresiones Reales	# Impresiones Función
10	3	3
100	6	6
1000	9	9
5000	12	12
100000	16	16

## Codificación y prueba

```
void Ej6(int n){  
    printf("\n*****Ejercicio 6 con n = %d*****\n", n);  
    int cont=0;  
    for(int i=10; i<n*5; i*=2)  
    {  
        cont++;  
    }  
    printf("# Impre Algoritmos %d\n", cont);  
    printf("Funcion %d\n", (int)(log(n)/log(2)));  
}
```

```
lgoritmos/Ejercicio02$ gcc ej02.c -o x -lm  
enrike@enrike:/media/Google_Drive/5° SEMESTRE/Analisis de  
lgoritmos/Ejercicio02$ ./x  
  
*****Ejercicio 6 con n = 10*****  
# Impre Algoritmos 3  
Funcion 3  
  
*****Ejercicio 6 con n = 100*****  
# Impre Algoritmos 6  
Funcion 6  
  
*****Ejercicio 6 con n = 1000*****  
# Impre Algoritmos 9  
Funcion 9  
  
*****Ejercicio 6 con n = 5000*****  
# Impre Algoritmos 12  
Funcion 12  
  
*****Ejercicio 6 con n = 100000*****  
# Impre Algoritmos 16  
Funcion 16
```

## 2.2. Algoritmo 7

### Análisis

```

⑦ For (j = n; j > 1, j / 2)
    {
        if (j < (n / 2))
            For (i = 0; i < n, i + 2) {
                printf("\nAlgoritmos\n");
            }
    }
}

```

$$f(n) = \left\lceil \log_2 \left( \frac{n}{2} \right) - 1 \right\rceil \frac{n}{2}$$

$$f(n) = \# \text{ "Algoritmos"}$$

$f(-1) = 0$	$f(9) = 5$
$f(0) = 0$	$f(10) = 5$
$f(1) = 0$	$f(11) = 6$
$f(2) = 1$	$f(12) = 6$
$f(3) = 2$	$f(13) = 7$
$f(4) = 3$	$f(14) = 7$
$f(5) = 4$	$f(15) = 8$
$f(6) = 5$	$f(16) = 16$

n	# Impresiones Reales
10	5
100	200
1000	3500

n	# Impresiones Función
10	5
100	200
1000	2000
10000	4000

### Resultados

n	# Impresiones Reales	# Impresiones Función
10	5	6
100	200	232
1000	3500	3982
5000	25000	25719
100000	700000	730482

## Codificación y prueba

```
void Ej7(int n){
    printf("\n****Ejercicio 7 con n = %d*****\n", n);
    int cont=0;
    for(int j=n; j>1; j/=2){
        if(j<n/2){
            for(int i=0; i<n; i+=2)
            {
                cont++;
            }
        }
    }
    printf("# Impre Algoritmos %d\n", cont);
    printf("Funcion %d\n", (int)((n/2)*((log(n/2)/log(2))-1)));
}
```

```
enrike@enrike:/media/Google_Drive/5° SEMESTRE/Analisis de
lgoritmos/Ejercicio02$ ./x

****Ejercicio 7 con n = 10*****
# Impre Algoritmos 5
Funcion 6

****Ejercicio 7 con n = 100*****
# Impre Algoritmos 200
Funcion 232

****Ejercicio 7 con n = 1000*****
# Impre Algoritmos 3500
Funcion 3982

****Ejercicio 7 con n = 5000*****
# Impre Algoritmos 25000
Funcion 25719

****Ejercicio 7 con n = 100000*****
# Impre Algoritmos 700000
Funcion 730482
```

## 2.3. Algoritmo 8

### Análisis

(8)

```
i=n;
while(i>=0){
    for(j=n; i<j; i-=2, j/=2)
        printf("# Algoritmos\n");
}
```

Esta condición  
NUNCA se cumple

$F(n) = \# \text{Algoritmos}$

$F(6)=0$

$F(0)=0$

$F(1)=0$

$$F(n)=0$$

n	# Impresiones Reales	# Impresiones Función
10	0	0
100	0	0
1000	0	0

### Resultados

n	# Impresiones Reales	# Impresiones Función
10	0	0
100	0	0
1000	0	0
5000	0	0
1000000	0	0

### Codificación y prueba

```
void Ej8(int n){
    printf("\n*****Ejercicio 8 con n = %d*****\n", n);
    int i=n, j, cont=0;

    while(i>=0){
        for(j=n; i<j; i-=2, j/=2)
        {
            cont++;
        }
    }
    printf("# Impresiones Algoritmos %d\n", cont);
}
```

```

enrike@enrike:/media/Google_Drive/5° SEMESTRE/Analisis de
lgoritmos/Ejercicio02$ gcc ej02.c -o x -lm
enrike@enrike:/media/Google_Drive/5° SEMESTRE/Analisis de
lgoritmos/Ejercicio02$ ./x

*****Ejercicio 8 con n = 10*****

```

3. Para los siguientes algoritmos determine las funciones de complejidad temporal, para el mejor caso, peor caso y caso medio. Indique cual(es) son las condiciones (instancia de entrada) del peor caso y cual(es) la del mejor caso.

### 3.1. Algoritmo 9

#### Análisis

Operaciones básicas : Comparación entre elementos del arreglo A  
Asignaciones a mayor1, mayor2

⑨ func Producto 2 Mayores (A,n)

```

3 { if(A[1] > A[2]) ①
    mayor1 = A[1]; ①
    mayor2 = A[2]; ①
  } else
    mayor1 = A[2]; ①
    mayor2 = A[1]; ①
  i=3;
  while(i <= n) ②
    if(A[i] > mayor1) ③
      mayor2 = mayor1; ④
      mayor1 = A[i]; ⑤
    else if(A[i] > mayor2) ⑥
      mayor2 = A[i]; ⑦
    i=i+1;
  return = mayor1 * mayor2;
Fin

```

Los 2 mayores son los primeros

Ftmc(n) =  $3 + 2(n-2)$

Orden ASC y los 2 mayores son los últimos.

Ftmc(n) =  $\frac{1}{3} [3 + 3(n-2)]$

$3 + 2(n-2) + \frac{1}{3} [3 + 3(n-2)] + \frac{1}{3} [3 + 2(n-2)]$

$= 2[1 + n-2] + 1 + \frac{2}{3}(n-2)$

$= 2 + 2n - 4 + 1 + \frac{2}{3}n - \frac{4}{3}$

$= \frac{8}{3}n - \frac{7}{3}$

### Operaciones básicas:

- Comparacion entre elementos del arreglo A
- Asignaciones a mayor1, mayor2

**Instancias** =  $I_1, I_2, I_3$

1. Los 2 mayores son los primeros
2. Orden ascendente y los 2 mayores son los ultimos
3. Los 2 numeros mayores estan en distintas posiciones

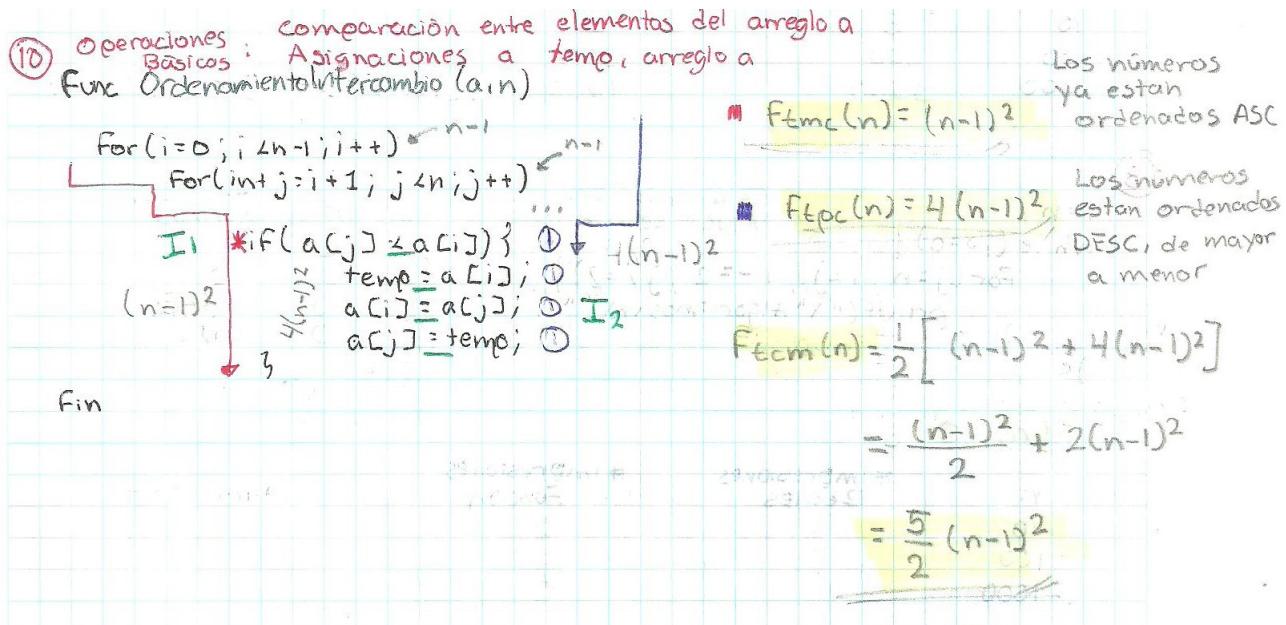
**Mejor Caso (Instancia 1)**  $f_{tmc}(n) = 3 + 2(n - 2)$

**Peor Caso (Instancia 2)**  $f_{tpc}(n) = 3 + 3(n - 2)$

Caso Medio (La probabilidad es  $\frac{1}{3}$ )  $f_{tcm}(n) = \frac{8}{3}n - \frac{7}{3}$

## 3.2. Algoritmo 10

### Análisis



### Operaciones básicas:

- Comparación entre elementos del arreglo a
- Asignaciones a temp y al arreglo a

**Instancias =  $I_1, I_2$**

1. Los números ya están ordenados ascendenteamente
2. Los números están desordenados aleatoriamente o de forma descendente (mayor a menor)

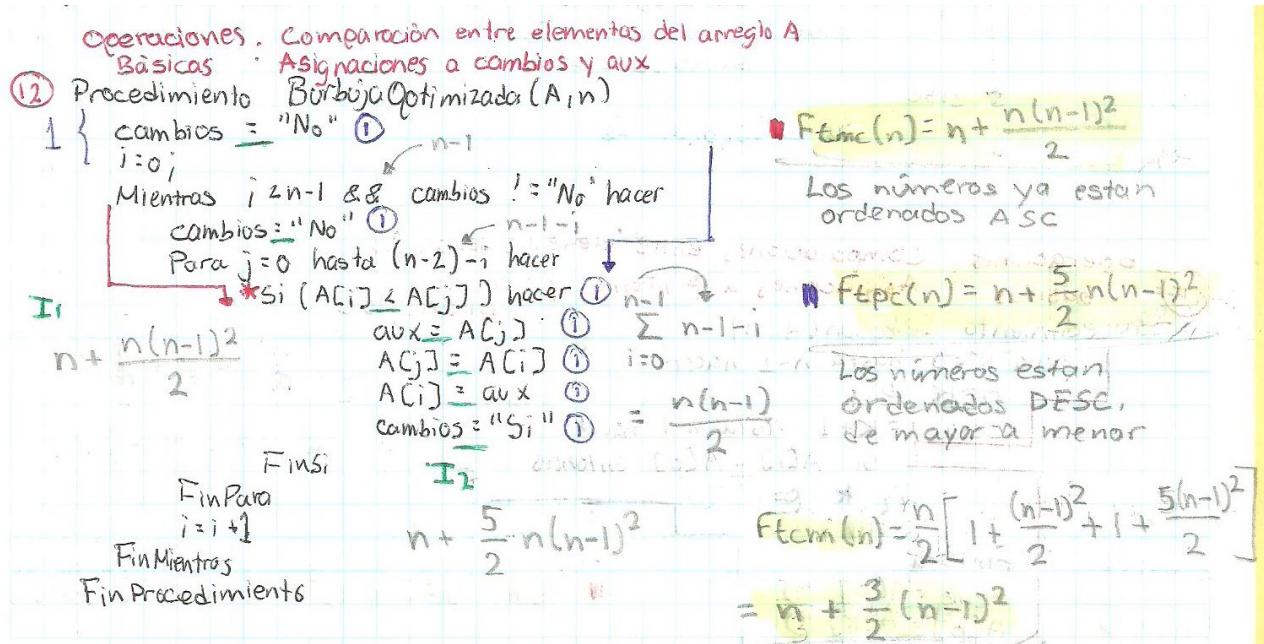
**Mejor Caso (Instancia 1)  $f_{tmc}(n) = (n - 1)^2$**

**Peor Caso (Instancia 2)  $f_{tpc}(n) = 4(n - 1)^2$**

Caso Medio (La probabilidad es  $\frac{1}{2}$ )  $f_{tcm}(n) = \frac{5}{2}(n - 1)^2$

### 3.3. Algoritmo 12

#### Análisis



#### Operaciones básicas:

- Comparación entre elementos del arreglo a
- Asignaciones a cambios y a aux

**Instancias =  $I_1, I_2$**

1. Los números ya están ordenados ascendenteamente
2. Los números están desordenados aleatoriamente o de forma descendente (mayor a menor)

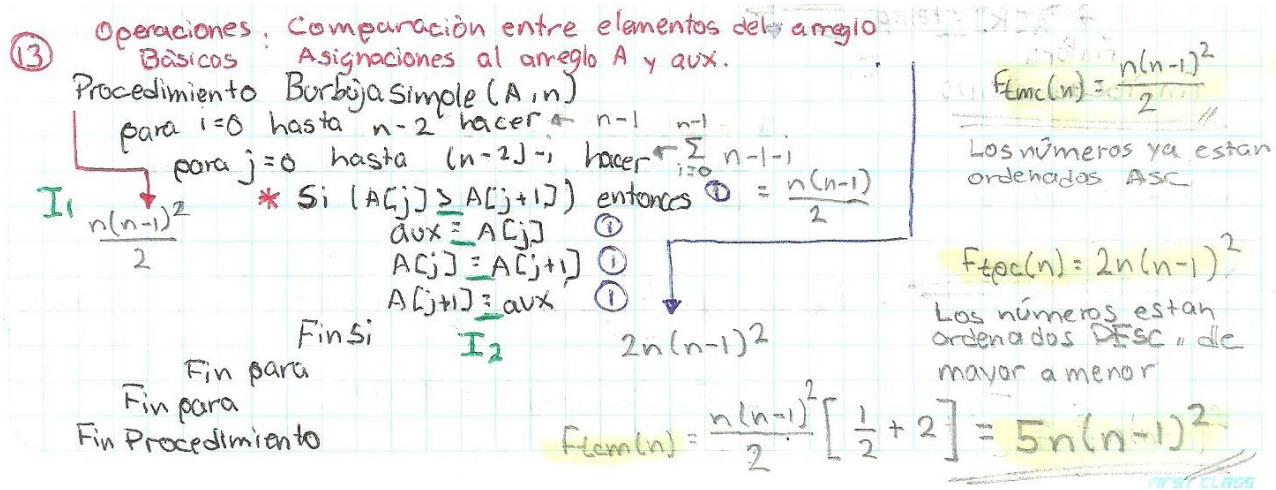
**Mejor Caso (Instancia 1)  $f_{tmc}(n) = n + \frac{n(n-1)^2}{2}$**

**Peor Caso (Instancia 2)  $f_{tpc}(n) = n + \frac{5}{2}n(n-1)^2$**

Caso Medio (La probabilidad es  $\frac{1}{2}$ )  $f_{tcm}(n) = n + \frac{3}{2}n(n-1)^2$

### 3.4. Algoritmo 13

#### Análisis



#### Operaciones básicas:

- Comparación entre elementos del arreglo A
- Asignaciones a aux y al arreglo A

Instancias = I<sub>1</sub>, I<sub>2</sub>

1. Los números ya están ordenados ascendenteamente
2. Los números están desordenados aleatoriamente o de forma descendente (mayor a menor)

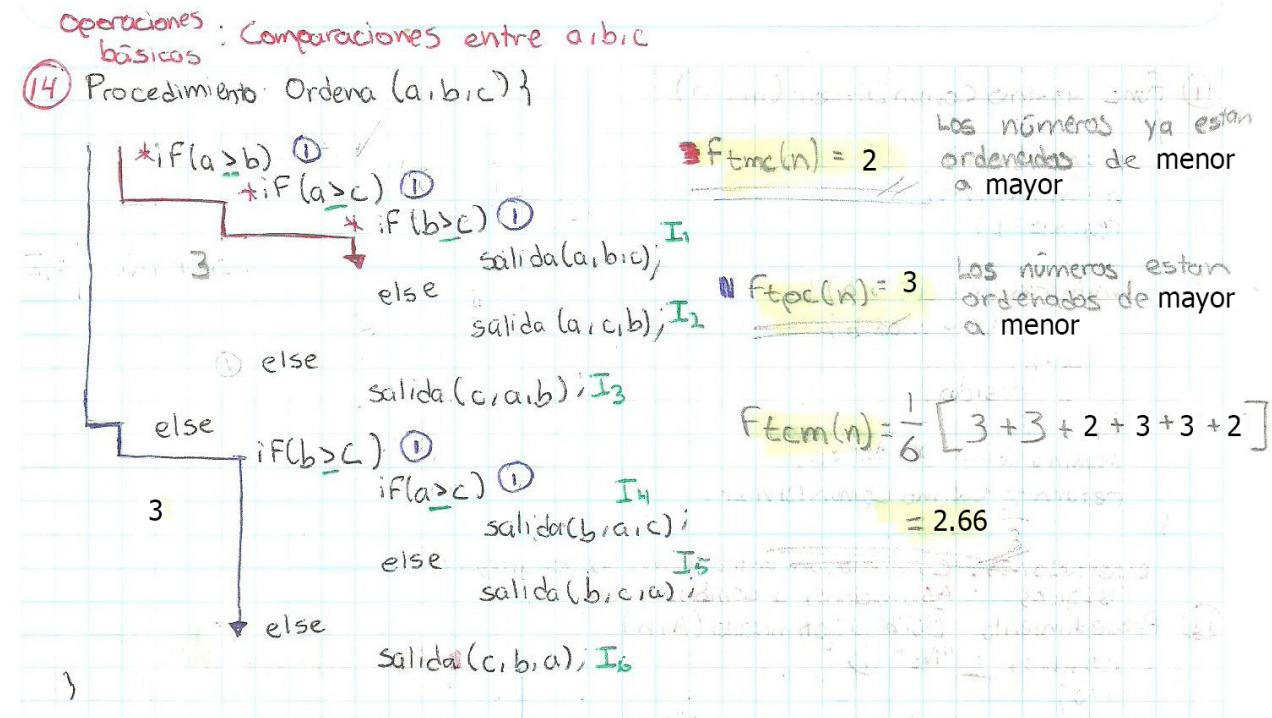
Mejor Caso (Instancia 1)  $f_{tmc}(n) = \frac{n(n-1)^2}{2}$

Peor Caso (Instancia 2)  $f_{tpc}(n) = 2n(n-1)^2$

Caso Medio (La probabilidad es  $\frac{1}{2}$ )  $f_{tcm}(n) = 5n(n-1)^2$

### 3.5. Algoritmo 14

#### Análisis



#### Operaciones básicas:

- Comparación entre a, b y c

Instancias =  $I_1, I_2, I_3, I_4, I_5, I_6$

1. a > b > c
2. a > c > b
3. c > a > b
4. b > a > c
5. a > c > a
6. c > b > a

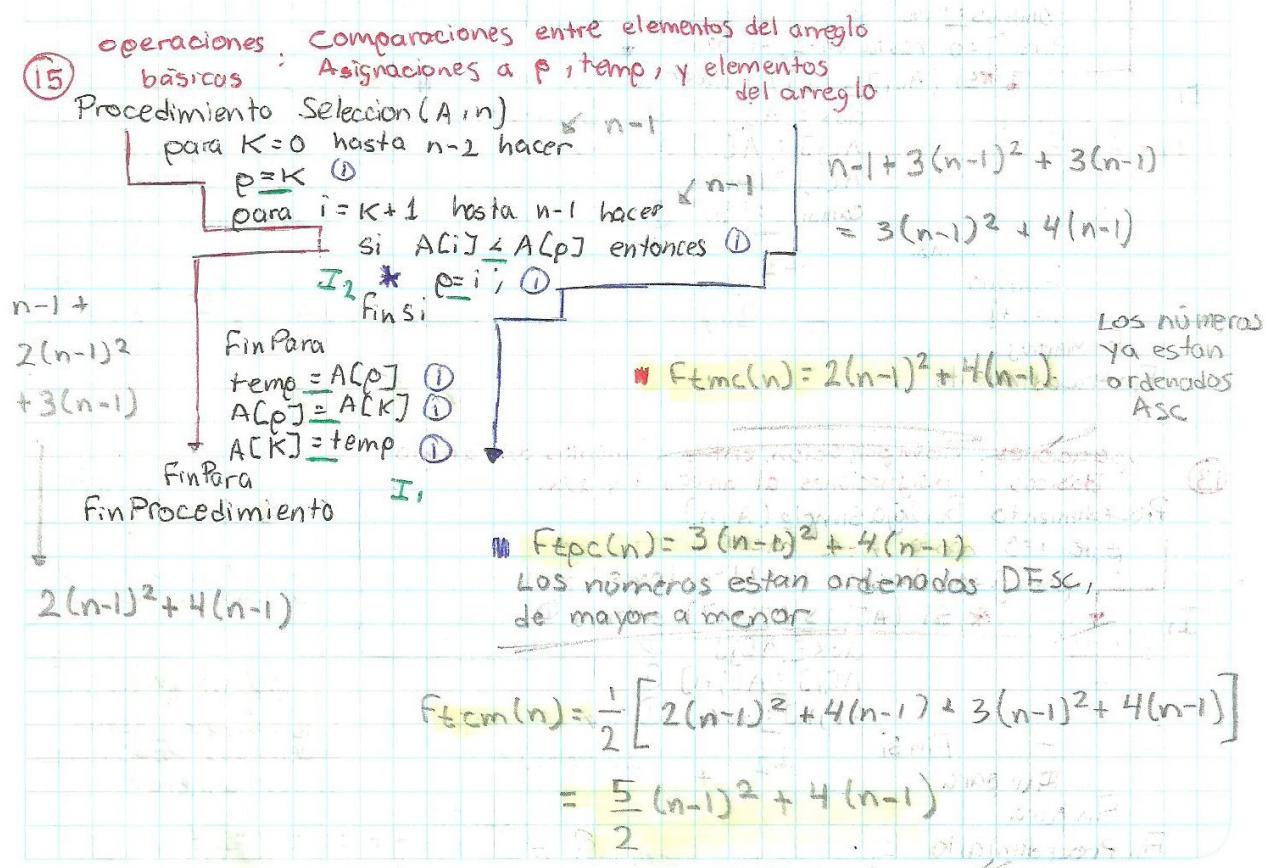
Mejor Caso (Instancia 6)  $f_{tmc}(n) = 2$

Peor Caso (Instancia 1)  $f_{tpc}(n) = 3$

Caso Medio (La probabilidad es  $\frac{1}{6}$ )  $f_{tcm}(n) = 2.66$

### 3.6. Algoritmo 15

#### Análisis



#### Operaciones básicas:

- Comparación entre elementos del arreglo A
- Asignaciones a p, temp y al arreglo A

Instancias =  $I_1, I_2$

- Los números ya están ordenados ascendenteamente
- Los números están desordenados aleatoriamente o de forma descendente (mayor a menor)

Mejor Caso (Instancia 1)  $f_{tmc}(n) = 2n(n-1)^2 + 4(n-1)$

Peor Caso (Instancia 2)  $f_{tpc}(n) = 3n(n-1)^2 + 4(n-1)$

Caso Medio (La probabilidad es  $\frac{1}{2}$ )  $f_{tcm}(n) = \frac{5}{2}n(n-1)^2 + 4(n-1)$