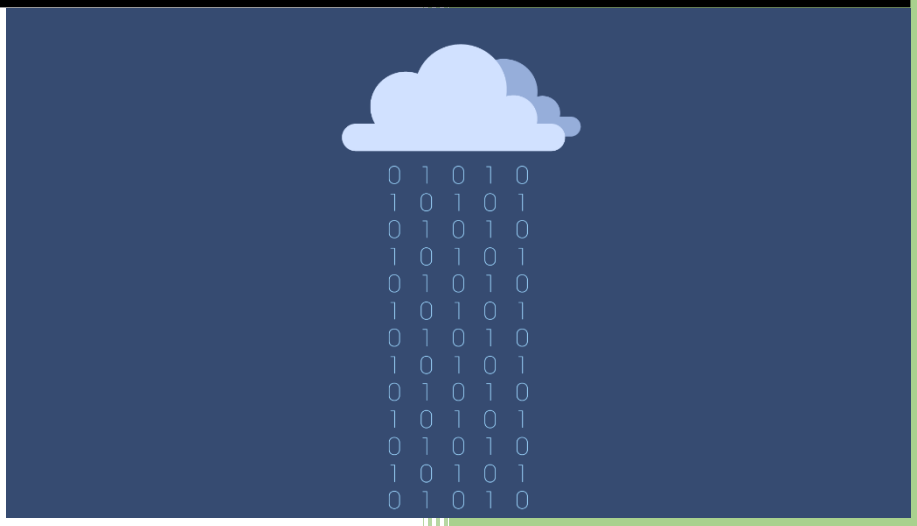
Análisis de algoritmos



Vargas Romero Erick Efraín

Prof. Franco Martínez Edgardo Adrián

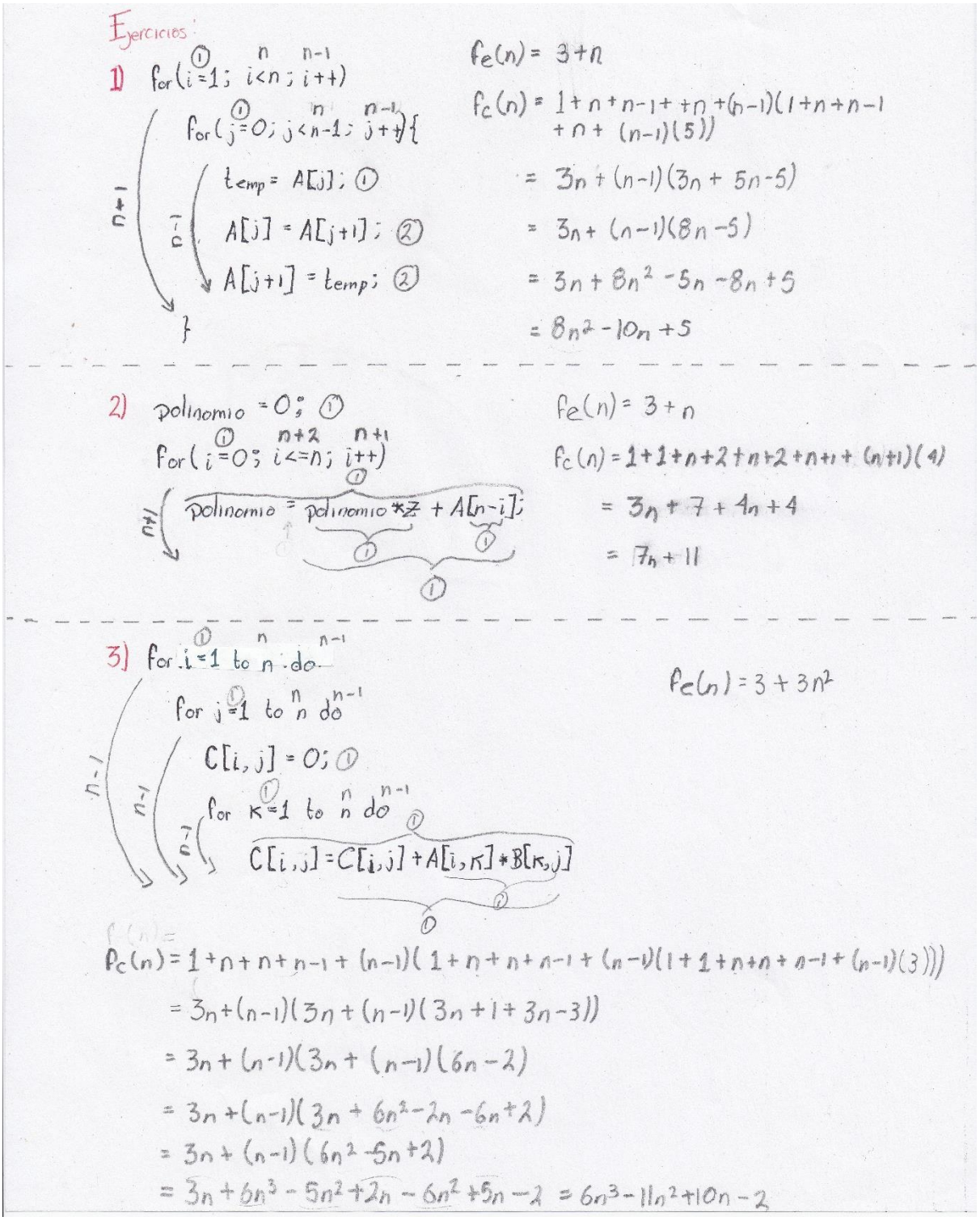
Grupo 3CM2

Tarea 02: Complejidad temporal y análisis de casos

Fecha de entrega 06/09/2018

Ejercicios: Complejidad de los algoritmos

A. Para los siguientes 5 algoritmos determine la función de complejidad temporal y espacial en términos de n. *Considere las operaciones de: asignación, aritméticas, condicionales y saltos implícitos.



4)

```
anterior = 1; (1)
actual = 1; (1)
while (n-1 > 2) {                                   (1)
    aux = anterior + actual; (2)                    (n-2)
    anterior = actual; (1) (1)                      (n-2)
    actual = aux; (1) (1)                           (n-2)
    n = n - 1; (2)                                  (n-2)
}
```

$fe(n) = 3$

$fc(n) = 1 + 1 + \overbrace{n-1}^{\text{comparaciones}} + \overbrace{n-1}^{\text{saltos}} + (n-2)(6)$

$= 2n + 6n - 12 - 18$

$= 8n - 12$

5)

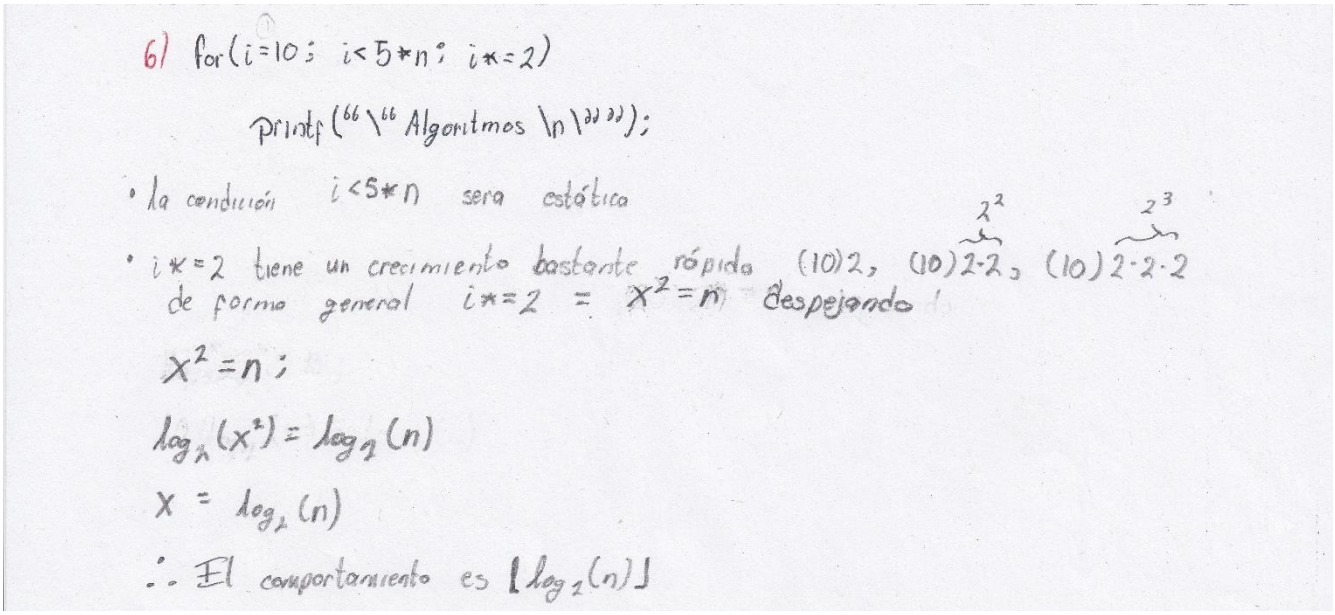
```
for(i = n-1 (2), j = 0 (1); i >= 0 (n+1); i--, j++) (n)   fe(n) = 2 + 2n
    s2[j] = s[i]; (1)                                     (n)

for(i = 0 (1); i < n (n+1); i++) (n)
    s[i] = s2[i]; (1)                                     (n)
```

$fc(n) = 2 + 1 + n+1 + n+1 + 2n + n(1) + 1 + n+1 + n+1 + n + n(1)$

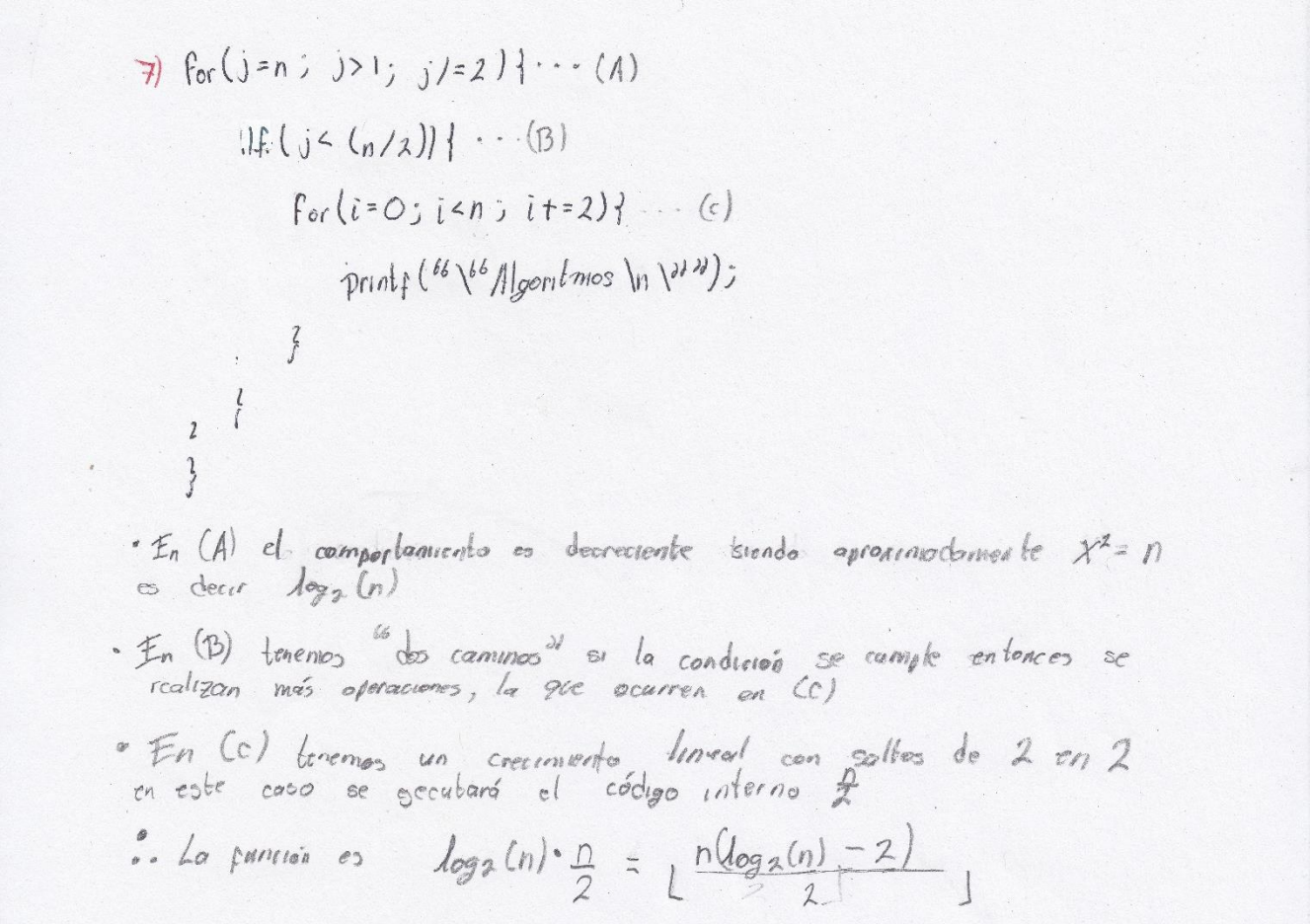
$= 9n + 8$

B. 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 compruébela 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.



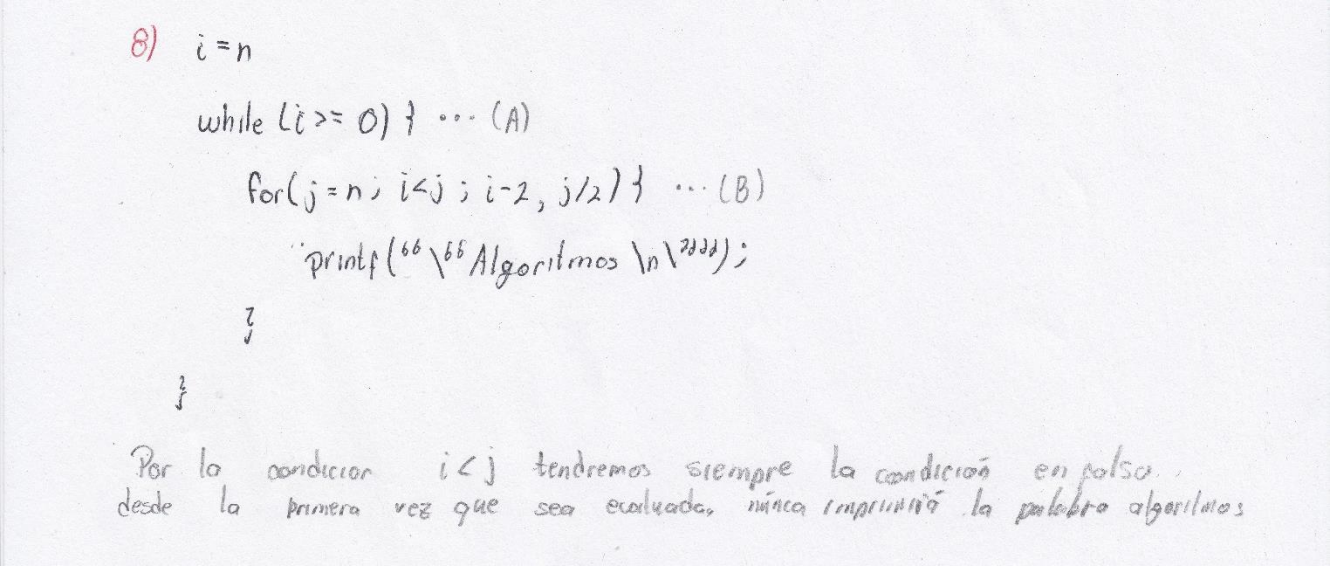

```c
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[]{
    int n, i;
    printf  ("Inroduce una N\n");
    scanf("%d", &n);
    for (i=10; i<n*5; i*=2){
        printf("\"Algoritmos\n\"");
    }
}
```

N	**Impresiones reales**	**Según la función**
10	3	3
100	6	6
1,000	9	9
5,000	12	12
10,000	13	13




```c
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[]{
    int n, i, j;
    printf  ("Inroduce una N\n");
    scanf("%d", &n);
    for (j=n; j>1; j/=2){
        if (j<(n/2)){
            for(i=0; i<n; i+=2){
                printf("\"Algoritmos\n\"");
            }
        }
    }
}
```

N	Impresiones reales	Según la función
10	5	5
100	10	100
1,000	3,500	3,500
5,000	25,000	25,000
10,000	55,000	55,000




```c
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[]{
    int n, i, j;
    printf  ("Inroduce una N\n");
    scanf("%d", &n);
    i=n;
    while(i>=0){
        for (j=n; i<j; i-=2, j/=2){
            printf("\"Algoritmos\n\"");
            k++;
        }
    }
    printf("%d\n", k);
    return 0;
}
```

N	Impresiones reales	Según la función
10	0	0
100	0	0
1,000	0	0
5,000	0	0
10,000	0	0

C. 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.

9) func Producto2Mayores (A, n)

```
if(A[1] > A[2])                    ┐ 3 op
    mayor1 = A[1]; ①               │
    mayor2 = A[2]; ①               ┘
else                               ┐ 3op
    mayor1 = A[2];                 │
    mayor2 = A[1];                 ┘
i = 3;
while (i <= n) {
    if(A[i] > mayor1)              ┐ 3op
        mayor2 = mayor1;           │
        mayor1 = A[i];             ┘
    else if (A[i] > mayor2)        ┐ 3op
        mayor2 = A[i];             ┘
    i = i + 1;
}
return mayor1 * mayor2;
fin
```

(2 comparaciones iguales)

Análisis: Primeramente debemos percatarnos que este código inicialmente busca los dos números más grandes del arreglo, por tanto

- Mejor caso: Si en las primeras posiciones están los números más grandes no se realizan las operaciones de asignación dentro del ciclo while

$\therefore 3 + (n-2)(2) = 3 + 2n - 4 = \underline{2n-1}$

- Peor caso: Si los mayores son los últimos del arreglo

$\therefore 3 + 3(n-2) = 3 + 3n - 6 = \underline{3n-3}$

- Caso medio: Consideremos las probabilidades de que ocurran los eventos al comparar (Suponemos que tienen la misma probabilidad)

$\therefore \frac{1}{2}(2n-1) + \frac{1}{2}(3n-3) = \frac{1}{2}(5n-4)$

---

10) func OrdenamientoIntercambio (a, n)

```
for (i=0; i<n-1; i++){
    for( int j = i+1; j<n ; j++)
        if(a[j] < a[i]){ ①
            temp = a[i]; ①
            a[i] = a[j]; ①
            a[j] = temp; ①
        }
}
fin
```

- Mejor caso: Que el arreglo se encuentre ordenado

$f_{mc}(n) = \sum_{i=0}^{n-1} i$

- Peor caso: Se encuentren en órden descendente

$f_{pc}(n) = \sum_{i=0}^{n-1} i$

- Caso medio:

$f_{cm}(n) = \frac{1}{2} \sum_{i=0}^{n-1} i$

11) Func MaximoComunDivisor(m, n){

```
a = max(n, m);
b = min(n, m);
residuo = 1;
mientras (residuo > 0){
    residuo = a mod b; (1)
    a = b;
    b = residuo
}
MaximoComunDivisor = a;
return MaximoComunDivisor;
}
```

(# elementos de Fibo hasta a)

- Mejor caso: Cuando a sea factor de b es decir a mod b = 0

∴ $f_{tmc}(n) = 1$

- Peor caso: Que a y b sean elementos de la sucesión de Fibonacci, si esto ocurre

∴ $f_{tpc}(n)$ = #elementos de fibonacci hasta que sea igual con a

- Caso medio:

∴ $f_{tem}(n)$ = # elementos de fibo hasta que sea igual con a +1

---

12) Procedimiento BurbujaOptimizado(A, n)

```
cambios = "No";
i = 0
Mientras( i < n-1 && cambios != "No" ) hacer
    cambios = "No";
    para j = 0 hasta (n-2)-i hacer
        Si (A[i] < A[j]) hacer (1)
            aux = A[j];
            A[j] = A[i]; (1)
            A[i] = aux; (1)
            cambios = "Si"
        Fin si
    Fin para
    i = i+1;
Fin mientras
Fin procedimiento
```

(n-1; n-2-i)

*Operaciones básicas: Comparación de los elementos de A[k] y asignaciones hacia A[k]

- Mejor caso: Que todos los A[i] sean mayores que A[j] es decir que A esté ordenado

∴ $f_{tmc}(n) = \sum_{i=0}^{n-1} (3n-6-3i)$

- Peor caso; Cuando A[i] ≤ A[j] se realizan todas las operaciones

∴ $f_{tpc}(n) = (n-1)(n-2-i)(3)$
$= (n-1)(3n-6-3i)$
$= 3n^2-6n-3ni-3n+6-3i$
$= 3n^2-9n-3(ni-i+2)$

pero Ver la "i" obtenemos

$\sum_{i=0}^{n-1}(3n^2-9n-3(ni-i+2))$

- Medio: Suponganos que todos los casos tienen la misma probabilidad

$\frac{1}{2}\left(\sum_{i=0}^{n-1}(3n^2-9n-3ni-6i+12)\right)$

13) Procedimiento BurbujaSimple(A, n)

```
para i=0 hasta n-2 hacer
    para j=0 hasta (n-2)-i hacer
        si (A[j] > A[j+1]) entonces ①
            aux = A[j]
            A[j] = A[j+1] ②
            A[j+1] = aux ③
        fin si
    fin para
fin para
fin procedimiento
```

n-2 - i

n-2

* Operaciones básicas: comparación entre elementos de A[k], asignaciones a A[k]

Mejor caso: A[k] este ordenado

$$f_{bmc}(n) = \sum_{i=0}^{n-2} (n-2)(n-2-i)(1)$$ //

Peor caso: que A[j] > A[j+1]

$$\therefore f_{bpc}(n) = \sum_{i=0}^{n-2} (n-2)(n-2-i)(3)$$

Caso medio: Consideremos probabilidades iguales

$$\frac{1}{2}\sum_{i=0}^{n-2}\{+(n-2)(n-2-i)\}$$ //

14) Procedimiento Ordena (a, b, c){

```
If(a>b){ ①
    If(a>c){ ⑪
        If(b>c){ ①
            salida(a, b, c);
        } else {
            salida(a, c, b);
        }
    } else {
        salida(c, a, b);
    }
⋮
```

- Mejor caso: Si los números cumplen $c>a>b$ ó $c>b>a$ $\therefore f_{bmc}(n) = 2$
- Peor caso: Si los números cumplen $a>b>c$ ó $a>c>b$ ó $b>a>c$ ó $b>c>a$ es decir que se realicen todas las condiciones $\therefore f_{bpc}(n) = 3$
- Caso medio:

$$f_{bcm}(n) = \frac{5}{2}$$ //

```
} else if(b > c) {①
        if(a > c) {①
            salida(b, a, c);
        } else {
            salida(b, c, a);
        }
    } else {
        salida(c, b, a);
    }
}
```

* Operaciones básicas, comparaciones entre a, b, c

15) Procedimiento Selección (A, n)

```
para k = 0 hasta n-2 hacer
    p = k
    para i = k+1 hasta n-1 hacer
        si A[i] < A[p] entonces ①
            p = i
        fin si
    fin para
    temp = A[p] ①
    A[p] = A[k] ①
    A[k] = temp ①
fin para
fin procedimiento
```

n-2 (para k)
n-k (para i)
n-2-k (si)

- Mejor caso: Si A[i] > A[p] es decir A está ordenado

$\therefore f_{mc}(n) = 2n - 4 - k(n-2)$

- Peor caso: Si A[i] < A[p] es decir A esta de forma descendente

$\therefore f_{pc}(n) = 2n^2 - 4n - k(n-2)$

- Caso medio: Considerando mismas posibilidades

$f_{cm}(n) = \frac{1}{2}\left(2n - 4 - k(n-2) + 2n^2 - 4n - k(n-2)\right)$

$= \frac{1}{2}\left(2n^2 - 2n - 4 - 2k(n-2)\right)$

$= n^2 - n - 2 - k(n-2)$