1º
```
For (i=1 ; i<=n ; i++)
    For (j=0 ; j<=n-1 ; j++)
    {
      temp = A[j];
      A[j] = A[j+1];
      A[j+1] = temp;
    }
```
n · n · 2

Burro es muy obvio que por las iteraciones la complejidad es $(O)_{n^2}$

2º
```
polinomio = 0
For (i=0 ; i<=n ; i++)
{
  polinomio = polinomio * z + A[n-i];
}
```
n · 4

Como se repite n por lo tanto es: $O(n)$

3º
```
For i = 0 to n do
    For j = 1 to n do
        C[i,j] = 0
        for k = 1 to n do
            C[i,j] = C[i,j] + A[i,k] * B[k,j]
```
n · n · n · 3

Por los Facanidados no importa que se repitan diferentes veces son dominados asintoticamente:

$= O(n^3)$

40

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

$O(n)$

50

```
for (i=n-1 ; j=0 ; i>=0; i--, j++ )
    S[j] = S[i]
for (i=0, i<n; i++
    S[i] = S2[i];
```

$O(n)$

A pesar de que son diferentes repeticiones de los for y al no estar anidados son dominados por $O(n)$

```
9° Func  Producto2Mayores (A, n)
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 ;          →1
while (i<=n) ——(n-3)——
        if (A[i] > mayor1) →1
                mayor2 = mayor1; →1
                mayor1 = A[i]; →1
        else if (A[i] > mayor2) →1
                mayor2 = A[i]; →1
        i = i + 1 →2
return = mayor1 * mayor2; →2
Fin
```

1

2

n

O(n)

Tomando el ciclo dentro de while la complejidad mayor es de 2 por la operación de asignación y suma se repite n-3 pero es dominado por n y como el if de arriba y el su valor es de 1 por lo tanto O(n)

10.

$O(n^2)$

Func Ordenamiento Intercambio (a, n)

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

$O(1)$ $O(n)$ $O(n^2)$

"n se domina asintoticamente por $O(n^2)$"

11. Fun Maximo Comun Divisor. (m, n)

```
{   a = max (n,m) ;   → 1
    b = min(n,m) ;   → 1
    residuo = 1;     → 1
    mientras (residuo > 0)
    {   residuo = a mod b ;  → log(a)
        a = b ;              → 1
        b = residuo :        → 1
    }
    Maximo Comun Divisor = a ;  → 1
    return Maximo Comun Divisor ; → 1
}
```

$O(\log n)$

12°

```
Procedimiento BurbujaOptimizada (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[j] < A[j]) hacer
                aux = A[j]
                A[j] = A[j]
                A[j] = aux
                cambios = "Si"
            Fin Si
        Fin Para
        i = i+1
    Fin Mientras
Fin Procedimiento
```

O(n²)

O(1) O(n) O(n²)

On es Dominada asintoticamente por $O(n^2)$

14.º Procedimiento Ordena (a, b, c)

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

O(1)

Al no tener en el codigo la funcion salida, no podre hacer un análisis más afondo entonces: O(1)

15 Procedimiento Seleccion(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
            p = i
        finsi
    fin para
    temp = A[p]
    A[p] = A[k]
    A[k] = temp
fin para
```

n2

Fin procedimiento

$O(n^2)$

Aunque el ciclo interno es de $O(n)$ es dominado asintoticamente $\underline{O(n^2)}$