Parcial 1

El coste temporal asintótico del fragmento

```
s=0; for(i=0;i<n;i++) for(j=i;j<n;j++) s+=i*j;
```

y el del fragmento

```
s=0; for(i=0;i<n;i++) for(j=0;j<n;j++) s+=i*i*j;
```

son ...

Seleccione una:

- a. ... el del segundo, menor que el del primero.
- b. ... iguales.
- c. ... el del primero, menor que el del segundo. ✗

Un problema de tamaño n puede transformarse en tiempo $O(n^2)$ en nueve de tamaño $n/3$; por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante. ¿cual de estas clases de coste temporal asintótico es la más ajustada?

Seleccione una:

- a. $O(n^2)$
- b. $O(n \log n)$
- c. $O(n^2 \log n)$

Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a. $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$ ✓
- b. $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$
- c. $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n=1 \\ n+3f(n/3) & n>1 \end{cases}$$

Seleccione una:

- a. $f(n) \in \Theta(n)$
- b. $f(n) \in \Theta(n \log n)$
- c. $f(n) \in \Theta(n^3)$ ✗

Recorrer sólo uno de los triángulos (el superior o el inferior) de una matriz $n \times n$ tiene una complejidad asintótica, con respecto a n que está en ...

Seleccione una:

- a. $O(n\sqrt{n})$
- b. $O(n \log n)$
- c. $\Omega(n^2)$

Indica cuál es la complejidad de la función siguiente:

```
unsigned sum( const mat &A ) {   // A es una matriz cuadrada
unsigned d = A.n_rows();
unsigned a = 0;
    for( unsigned i = 0; i < d; i++ )
        for( unsigned j = 0; j < d; j++ )
            a += A(i,j);
return a;
}
```

Seleccione una:

- a. $O(n)$
- b. $O(n \log n)$
- c. $O(n^2)$ ✗

Parcial 1

Sea $f(n)$ la solución de la relación de recurrencia $f(n)=2f(n-1)+1$; $f(1)=1$. Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a. $f(n) \in \Theta(n)$
- b. $f(n) \in \Theta(n^2)$
- c. $f(n) \in \Theta(2^n)$ ✓

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n){
if (n<=1)
   return 0;
unsigned sum = desperdicio (n/2) + desperdicio (n/2);
for (unsigned i=1; i<n-1; i++)
    for (unsigned j=1; j<=i; j++)
        sum+=i*j;
return sum;
}
```

Seleccione una:

- a. $\Theta(2^n)$
- b. $\Theta(n^2 \log n)$ ✗
- c. $\Theta(n^2)$

Los algoritmos de ordenación *Quicksort* y *Mergesort* tienen en común ...

Seleccione una:

- a. ... que se ejecutan en tiempo $O(n)$.
- b. ... que aplican la estrategia de *divide y vencerás*.
- c. ... que ordenan el vector sin usar espacio adicional. ✗

Indica cuál es la complejidad, en función de n, del fragmento siguiente:

```
int a = 0;
for( int i = 0; i < n; i++ )
    for(  int j = i; j > 0; j /=2 )
        a += A[i][j];
```

Seleccione una:

- a. $O(n)$ ✗
- b. $O(n \log n)$
- c. $O(n^2)$

Pertenece $3n^2+3$ a $O(n^3)$?

Seleccione una:

- a. No. ✗
- b. Sólo para $c=1$ y $n_0=5$.
- c. Sí.

La versión de *Quicksort* que utiliza como pivote el elemento del vector que ocupa la posición central ...

Seleccione una:

- a. ... no presenta casos mejor y peor distintos para instancias del mismo tamaño.
- b. ... se comporta mejor cuando el vector ya está ordenado. ✓
- c. ... se comporta peor cuando el vector ya está ordenado.