2.a.

$T_5 \rightarrow T_4 \rightarrow T_3 \rightarrow T_2 \rightarrow T_1$

El término dominante es log(n). Los términos constantes no afectan la complejidad.
$T_5 = \log_2(n) + 5$
Orden: O(log n)

$n^{(1.5)}$ domina por completo a los términos logarítmicos y constantes.
$T_4 = 45 + 2.7\cdot\log_4(n) + (\log_2(5))\cdot n^{1.5}$
Orden: $O(n^{1.5})$

El término dominante es n^5, el de mayor exponente dentro del polinomio.
$T_3 = 2\cdot\log_5(n) + \text{sqrt}(n) + 3\cdot n^2 + 2\cdot n^5$
Orden: $O(n^5)$

El término exponencial 2^n crece más rápido que cualquier polinomio.
$T_2 = 3\cdot 2^n + 5\cdot n^4 + 2\cdot n$
Orden: $O(2^n)$

4^n tiene el crecimiento más rápido y domina la función.
$T_1 = n^2 + 2\cdot 4^n + 53$
Orden: $O(4^n)$

2.b.

Buscamos mayor entero n tal que $n! \leq 10^9$

Cálculo rápido:
10!=3 628 800
11!=39 916 80011
12!=479 001 600$\leq 10^9$
13!=6 227 020 > 10^9

Resultado: **nmax =12n_{\max} = 12nmax=12**.

T(n)	Ecuación a resolver	$n_{\max}$ (aprox.)
$\log n$	$\log_2 n \leq 10^9$	$n \leq 2^{10^9}$ (≈ $10^{3.01\cdot 10^8}$, inmenso)
n	$n \leq 10^9$	1,000,000,000
$n \log n$	$n \log_2 n \leq 10^9$	39,620,077
n^2	$n^2 \leq 10^9$	31,622
2^n	$2^n \leq 10^9$	29
$n!$	$n! \leq 10^9$	12

2.c.

Incremento al duplicar n — factor T(2n)/T(n):

1. **T(n)=1**

$$\frac{T(2n)}{T(n)} = \frac{1}{1} = 1$$

Interpretación: no cambia; coste constante.

2. **T(n)=log n** (log base 2)

$$\frac{T(2n)}{T(n)} = \frac{\log_2(2n)}{\log_2 n} = \frac{\log_2 n + 1}{\log_2 n} = 1 + \frac{1}{\log_2 n}$$

Interpretación: para n grande el factor tiende a 1 (cambio **prácticamente nulo**).

3. **T(n)=n**

$$\frac{T(2n)}{T(n)} = \frac{2n}{n} = 2$$

Interpretación: duplicar el tamaño duplica el coste.

4. **T(n)=n log n** (log base 2)

$$\frac{T(2n)}{T(n)} = \frac{2n\log_2(2n)}{n\log_2 n} = 2 \cdot \frac{\log_2 n + 1}{\log_2 n} = 2\left(1 + \frac{1}{\log_2 n}\right)$$

Interpretación: algo **más** que 2, pero se acerca a 2 cuando n crece.

5. **T(n)=n^2**

$$\frac{T(2n)}{T(n)} = \frac{(2n)^2}{n^2} = 4$$

Interpretación: duplicar n cuadruplica el coste.

6. **T(n)=n^3**

$$\frac{T(2n)}{T(n)} = \frac{(2n)^3}{n^3} = 8$$

Interpretación: duplicar n lo multiplica por 23=82^3=823=8.

7. **T(n)=2^n**

$$\frac{T(2n)}{T(n)} = \frac{2^{2n}}{2^n} = 2^n$$

Interpretación: factor exponencial para n moderado ya es astronómico

Factores al duplicar n:

- 1 → factor 1
- log n → factor 1 + 1/$\log_2$ n (≈1 para n grande)
- n → factor 2
- n log n → factor 2(1 + 1/$\log_2$ n) (≈2)
- n^2 → factor 4

- n^3 → factor 8
- 2^n → factor 2^n (crecimiento explosivo)

2.d.

Considere el siguiente algoritmo:

```
def algoritmo(L, p, x):
if p == len(L):
return False
if L[p] == x:
return True
return algoritmo(L, p + 1, x)
```

¿Qué implementa?

Es una búsqueda lineal recursiva en la lista L, empezando en índice p, que comprueba si existe el elemento x.

- Si p alcanza len(L) devuelve False (no encontrado).
- Si L[p] == x devuelve True (encontrado).
- En otro caso llama recursivamente con p+1.

Análisis temporal (suponiendo n = len(L) - p, es decir, número de elementos por revisar)

Podemos modelar el tiempo por la recurrencia:

$T(n) = T(n-1) + c$, con $T(0)=c_0$,

donde c y c_0 son constantes (coste de comparar, comprobar índice y llamada).

- **Peor caso T_peor(n): cuando x no está en la lista (o está en la última posición). El algoritmo revisa los n elementos.**
  T_peor(n) = Θ(n) (linear). Si se cuenta comparaciones, aproximadamente n comparaciones.
- **Mejor caso T_mejor(n): cuando x se encuentra inmediatamente en L[p] (primer elemento examinado). Solo una comparación.**
  T_mejor(n) = Θ(1) (constante).
- **Caso promedio T_mean(n): si x está en la lista y su posición es equiprobable entre las n posiciones, la esperada es revisar la mitad en promedio. Si también se considera la probabilidad de no estar, la constante cambia pero sigue siendo proporcional a n. T_mean(n) = Θ(n) (específicamente ≈ n/2 comparaciones si está con prob. uniforme).**
- **Notación asintótica: el orden de complejidad es**
  O(n) en el peor caso y en promedio.