Cotas Inferiores : # pasos mínimos que se debe hacer para resolver un problema. (no necesariamente se resuelve en esa cantidad)

Ejemplo: ordenar arreglo de n elementos → $\Omega(n)$

* Un algoritmo entrega una cota superior.

La complejidad de un problema es el costo del mejor algoritmo que resuelve ese problema

¿Cómo encontrar algoritmo óptimo? ⟹ Cotas inferiores

* Una cota inferior ajustada es cuando no puede haber una mejor cota inferior para el problema, la más alta posible.

Si conocemos la cota superior $O(T(n))$ y a la vez cotas inferiores $\Omega(T(n))$ de un problema, entonces:

- Los algoritmos $O(T(n))$ son óptimos.
- La cota inferior $\Omega(T(n))$ es ajustada.
- el problema tiene complejidad exacta $\Theta(T(n))$.

En el caso contrario, se tiene:

- cota superior $O(T_1(n))$
- cota inferior $\Omega(T_2(n))$ y $T_2(n) = o(T_1(n))$
- no se sabe si el algoritmo es óptimo ni si la cota inferior es ajustada.

¿Cómo calcular las cotas inferiores? ⟹ 3 TÉCNICAS

1. Estrategia del adversario (cotas de peor caso)
2. Teoría de la Información (cotas de caso promedio/peor caso)
3. Reducciones

1. Estrategia del adversario

El algoritmo NO conoce el input y debe aprender lo suficiente sobre este.

El algoritmo "pregunta" y el adversario "responde" de manera de generar el mayor costo posible → entregarle la menor información posible

// Ejemplo: • encontrar un elemento en un arreglo desordenado → n accesos

• en un arreglo ordenado $A[1,n]$: debemos conocer $A[k,k]$

└ El algoritmo conoce i, j t.q $x \in A[i,j]$.

└ El algoritmo pregunta por $A[k]$, para algún k.

(i) $k \notin [i,j]$, no aprende nada

(ii) $k \notin [i,j] \wedge A[k] = x \longrightarrow$ RESPUESTA

(iii) $k \notin [i,j] \wedge A[k] > x \longrightarrow [i, k-1]$

(iv) $k \in [i,j] \wedge A[k] < x \longrightarrow [k+1, j]$

} El intervalo se reduce a lo sumo a la mitad

$\longrightarrow (\log_2 n)$ accesos

★ El adversario nos puede "sugerir" un algoritmo. // búsqueda binaria en arreglo ordenado.

★ Se debe bien definir el adversario para encontrar la cota inferior.

// Encontrar el máximo de un arreglo → $n-1$ comparaciones (modelo grafo o tenis)

// Adivinar el personaje → el adversario responde para dejar la mayor cantidad de personajes posibles.

¿Cómo usar esta técnica? ⇒ crear un **MODELO** de lo que el algoritmo va aprendiendo:

- e° inicial
- e°s finales

} El mínimo costo de llegar del e° inicial a algún e° final es una cota inferior.

¿Cómo saber si nuestra cota inferior es ajustada?

↳ Debemos encontrar un algoritmo (cota superior) que cueste lo mismo que la cota inferior.