**Complejidad computacional**

**Eficiencia**

* Capacidad de resolver el problema propuesto empleando un bajo consumo de recursos computacionales. Los principales recursos son:
* **Coste espacial:** Cantidad de memoria requerida
* **Coste temporal:** Tiempo necesitado para resolver el programa

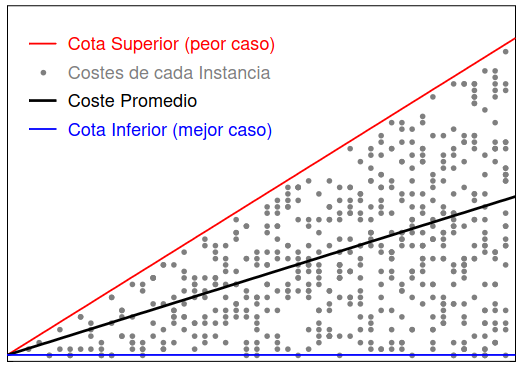
**Talla**

* Dado un programa que opera con un número **n**, el valor de n se denomina **talla**.
* Ejemplo: si un programa opera con un vector, la talla será el número de datos en el vector
* A la hora de diseñar un algoritmo, debemos considerar que el programa sea eficiente para tallas elevadas.

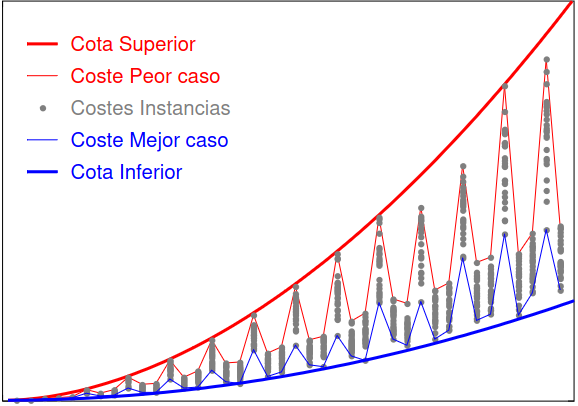
**Paso**

* Un **paso** es la ejecución de un segmento de código cuyo tiempo de proceso no depende de la talla, o está limitado por alguna constante.
* Constituyen 1 paso:
  + Asignación, operaciones aritméticas o lógicas, comparación, acceso a un elemento de un vector
  + Cualquier secuencia finita cuya longitud no dependa de la talla.
* El **coste computacional de un programa T(n)** es el número de pasos en función de la talla (n).
* Los pasos que no están relacionados con la talla se pueden ignorar. Por ejemplo, si un bucle se ejecuta un nº constante de veces, no tendrá repercusión en T(n)
  + El nº total de pasos estará en función de la talla.

**Casos mejor, peor y promedio**

* En ocasión, el coste computacional de un programa no depende sólo de la talla.
* Por ejemplo, en un algoritmo de búsqueda lineal, el tiempo de ejecución no dependerá sólo del nº de elementos en la lista, sino de la posición del elemento que buscamos (si está al principio se encontrará antes).
  + Este es un factor difícil de medir cuando se considera el algoritmo.
* Por esto, para un algoritmo se pueden medir por separado la **cota inferior (Ω)** y la **cota superior (O)**: el nº de pasos requeridos en el mejor y peor caso, respectivamente.
* Un algoritmo de búsqueda lineal será **Ω(1)** (el elemento puede ser el primero) y **O(n)**

**Notación asintótica**

* En ocasiones es imposible expresar la complejidad computacional sólo como una función de la talla, pues para ciertas instancias el coste es distinto (ver imagen, donde el coste es mayor para n múltiplo de 3)
* En estas ocasiones conviene funciones que **acoten** superior e inferiormente los costes de todas las instancias para tallas grandes.
* Siendo t(n) el coste en función de la talla, estas funciones de cota serán O(t(n)) y Ω(t(n))
* Sean f y t funciones, se dice que **f(x) es O(t(x))** si existen dos constantes (testigos) c y k tales que:

|f(x)| ≤ c\*|t(x)| en x > k

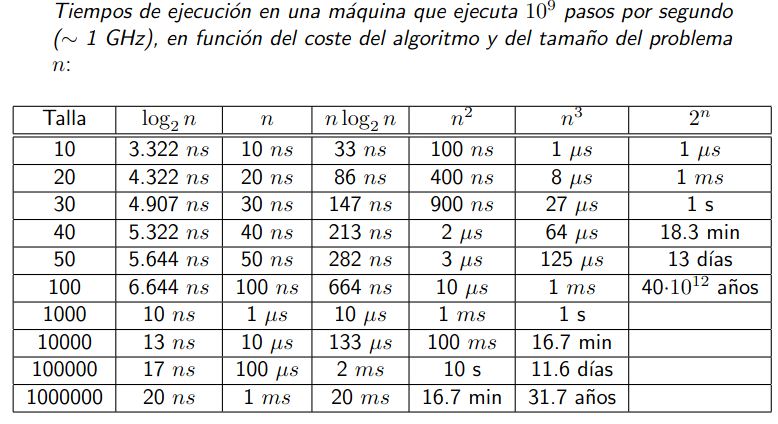
* Sean f y t funciones, se dice que **f(x) es Ω(t(x))** si existen dos constantes (testigos) c y k tales que:

|f(x)| ≥ c\*|g(x)| en x > k

* Sean f y t funciones, se dice que **f(x) es Ө(t(x))** si f(x) es **O(t(x))** **y** **Ω(t(x))**
* O(1) ⊂ O(log n) ⊂ O(√n) ⊂ O(n) ⊂ O(n log n) ⊂ O(n2) ⊂ O(n3) ⊂ O(2n)⊂ O(n!) ⊂ O(nn)

Sublineales Superlineales

**Consecuencias prácticas**



**Otros conceptos**

* **Complejidad espacial:**
  + **Posición de memoria:** Espacio de almacenamiento ocupado por uno o más datos, cuya extensión es independiente de la talla (equivalente a paso)
  + **Coste espacial:** nº de posiciones de memoria requeridas para la ejecución de un programa
* En ocasiones, la talla se puede expresar en función de más de un parámetro (ej: una matriz mxn)