

# Técnicas para la verificación empírica de la complejidad

## Tema 1

### Algoritmos

Fac. de Informática, Universidade da Coruña

Alberto Valderruten  
Santiago Jorge



- **Aplicación del “método empírico” al análisis de algoritmos:**
  - medir tiempos de ejecución (experimentos sistemáticos)
  - $\Rightarrow$  tabla de tiempos para distintos valores de  $n$
  - $\Rightarrow$  ¿ $O$ ?
- Método empírico: Renacimiento, s. XVII (Galileo)
  - “Mide lo que se pueda medir;*
  - lo que no se pueda... hazlo medible!”*
- **Verificación** empírica: normalmente, se parte de una función  $f(n)$  candidata (obtenida mediante el análisis).
- **Aplicación:** Trabajo en prácticas

# Estrategia para la medición de tiempos de ejecución (1)

## Informe de resultados

- **Contexto:**

- indicar **qué** se está midiendo:  
algoritmo, caso, características de la entrada, etc.
- indicar **dónde** se está midiendo:  
id. ordenador del laboratorio (no debe ser un servidor!),  
características del ordenador personal  
 $\leftrightarrow$  ¿control del entorno de experimentación?
- indicar unidades de tiempo:  $\mu s$ ,  $ms$ ,  $s$ ...

- **Tabla de tiempos:**  $m$  mediciones para distintos valores de  $n$

$n$	$t(n)$
$n_1$	$t_1$
$n_2$	$t_2$
$n_3$	$t_3$
$\dots$	$\dots$
$n_m$	$t_m$

# Estrategia para la medición de tiempos de ejecución (2)

- **Normas** en la obtención de tablas de tiempos (prácticas):

$n$	$t(n)$
$n_1$	$t_1$
$n_2$	$t_2$
$n_3$	$t_3$
$\dots$	$\dots$
$n_m$	$t_m$

- Los  $n_i$  deben seguir una **progresión geométrica**:  
→ \*2, \*10 únicamente
- Debe medirse un mínimo de 5 valores de  $n$  ( $m \geq 5$ ):  
→ idealmente 7-8 mediciones
- **No debe haber tiempos nulos** ( $t_i > 0$ )...
- ... ni tiempos muy pequeños medidos directamente  
→  $t_i \geq$  **umbral de confianza**:  $t_i \geq 500\mu s$   
→ *estrategia para la medición de tiempos pequeños*
- No esperar mucho tiempo por un resultado  
→ abortar mediciones e indicarlo en la tabla, si procede

## Medición de tiempos pequeños:

```
leer_tiempo (ta);  
repetir K veces:  
    alg(n);  
leer_tiempo (tb)
```

- K debe ser una potencia de 10
- en la tabla se pondrá  $(tb - ta)/K$
- y se indicará con una nota al pie que esa medición corresponde a un tiempo promedio de  $K$  ejecuciones del algoritmo
- *sólo válido para algoritmos que no modifican la entrada:*
  - Qué hacer por ej. con un algoritmo de ordenación?
    - Ok para medir ordenación de una entrada ordenada
    - No usar en cualquier otro caso!

# Estrategia para la medición de tiempos de ejecución (4)

## Medición de tiempos pequeños (Cont.):

Ejemplo: *“ordenación de un vector aleatorio”*

Se podría pensar en algo así (**Atención: esta solución está MAL!**):

```
inicializar(vector);  
leer_tiempo (ta); alg(vector); leer_tiempo (tb);  
t:=tb-ta;  
si (t < 500) entonces {                                % ``umbral de confianza``  
    t:=0;  
    repetir K veces: {  
        inicializar(vector);  
        leer_tiempo (ta); alg(vector); leer_tiempo (tb);  
        t:=t+(tb-ta)                                     % ESTA MAL  
    };  
    t:=t/K  
}
```

¿Valoración de esta solución? **Suma de errores! → EVITAR**

# Estrategia para la medición de tiempos de ejecución (5)

## Medición de tiempos pequeños (Cont.):

Ejemplo: *“ordenación de un vector aleatorio”*

```
inicializar(vector);  
leer_tiempo (ta); alg(vector); leer_tiempo (tb);  
t:=tb-ta;  
si (t < 500) entonces {           % ``umbral de confianza``  
    leer_tiempo (ta);  
    repetir K veces: {  
        inicializar(vector); alg(vector)  
    };  
    leer_tiempo (tb);  
    t1:=tb-ta;                     % debería estar por encima de 500!  
    leer_tiempo (ta);  
    repetir K veces:               % debe ser la misma constante K  
        inicializar(vector);  
    leer_tiempo (tb);  
    t2:=tb-ta;  
    t:=(t1-t2)/K  
}
```

## ¿Valoración de esta solución?

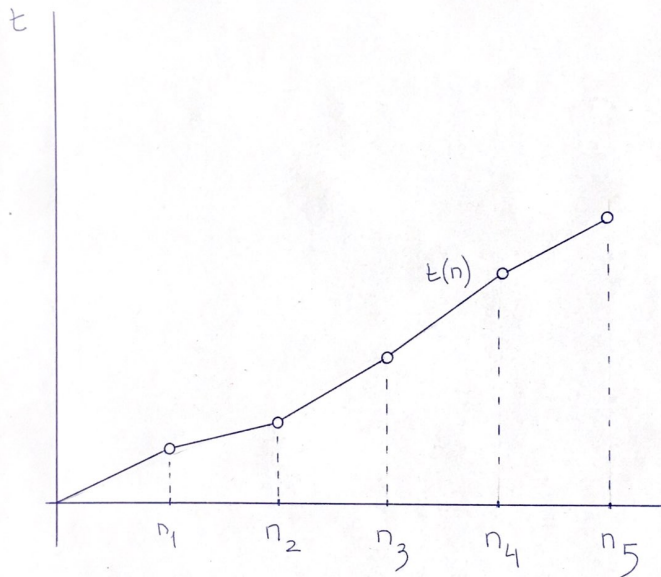
- se mide más “ruido” (gestión del bucle...) → dudas sobre la calidad de la mediciones
- riesgo de tiempos negativos → comprobar las tablas, calcularlas varias veces. . .
- $t_1$  también debería estar, a su vez, por encima del umbral de confianza
- . . .

A pesar de todo, la menos mala

→ *“Norma” para las prácticas*



# Ejemplo de gráfica de tiempos



# Estrategia para determinar la complejidad del algoritmo (1)

$n$	$t(n)$
$n_1$	$t_1$
$n_2$	$t_2$
$n_3$	$t_3$
$\dots$	$\dots$
$n_m$	$t_m$

**Punto de partida:**

**Objetivo:** encontrar  $f(n)$  tal que  $t(n)/f(n) \rightarrow C \text{ cte. } > 0$

**donde  $f(n)$  sea una de las funciones características** (“típicas”):  
 $\log n, n, n \log n, n^k, 2^n \dots$

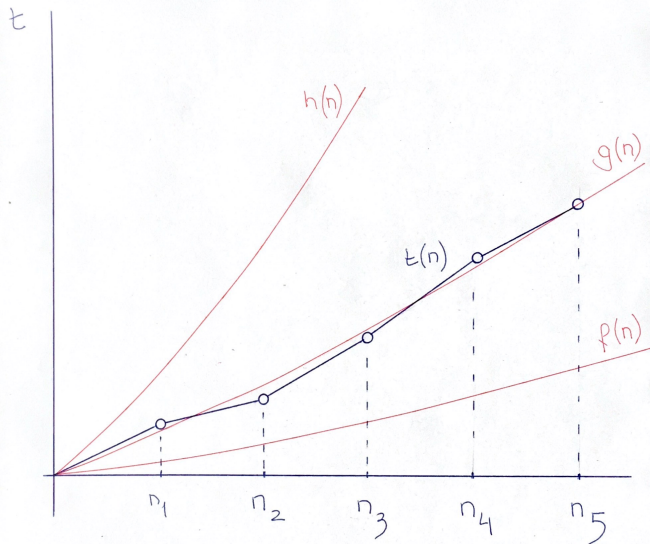
$\leftrightarrow$  una función con *el mismo crecimiento!*

- diremos entonces que  $f(n)$  **es una cota (superior) ajustada** (para  $t(n)$ )

- $t(n) = O(f(n))$

$\rightarrow$  ¿Cómo asegurarse de que la serie  $t(n)/f(n) \rightarrow C > 0$ ?

# Ejemplo de gráfica de tiempos comparando posibles cotas



# Estrategia para determinar la complejidad del algoritmo (2)

## Técnica propuesta:

$n$	$t(n)$	$t(n)/f(n)$	$t(n)/g(n)$	$t(n)/h(n)$
$n_1$	$t_1$	$t_1/f(n_1)$	$t_1/g(n_1)$	$t_1/h(n_1)$
$n_2$	$t_2$	$t_2/f(n_2)$	$t_2/g(n_2)$	$t_2/h(n_2)$
$n_3$	$t_3$	$t_3/f(n_3)$	$t_3/g(n_3)$	$t_3/h(n_3)$
...	...	...	...	...
$n_m$	$t_m$	$t_m/f(n_m)$	$t_m/g(n_m)$	$t_m/h(n_m)$

## Donde:

- $f(n)$  es una cota **ligeramente subestimada** como cota superior ajustada
- $g(n)$  es una cota **ajustada**:  $t(n) = O(g(n))$
- $h(n)$  es una cota **ligeramente sobrestimada**

Atención al orden de las funciones!  $\langle f(n), g(n), h(n) \rangle$  ordenadas

Ejemplo:  $\langle n, n \log n, n^{1,5} \rangle$ ;  $\langle n^{0,8}, n, n^{1,2} \rangle$ ; ...

→ ¿Cómo diferenciar las 3 situaciones?

## Estudio de la convergencia de una serie $t(n)/f(n)$ :

Dadas:

- $t(n)$  la serie de mediciones obtenidas (tiempos)
- $f(n)$  la cota con la que vamos a comparar las mediciones (una función característica)

$\Rightarrow$

- Si  $t(n)/f(n) \rightarrow \infty$  (diverge) cuando  $n \rightarrow \infty$ :  
 $f(n)$  es una cota **subestimada**
- Si  $t(n)/f(n) \rightarrow C > 0$  cuando  $n \rightarrow \infty$ :  
 $f(n)$  es una cota **ajustada**:  $t(n) = O(f(n))$
- Si  $t(n)/f(n) \rightarrow 0$  (decrece) cuando  $n \rightarrow \infty$ :  
 $f(n)$  es una cota **sobrestimada**

# Estrategia para determinar la complejidad del algoritmo (4)

## Presentación de los resultados:

(en prácticas: ficheros de texto bien alineados)

$n$	$t(n)$	$t(n)/f(n)$	$t(n)/g(n)$	$t(n)/h(n)$
$n_1$	$t_1^*$	$t_1/f(n_1)$	$t_1/g(n_1)$	$t_1/h(n_1)$
$n_2$	$t_2^*$	$t_2/f(n_2)$	$t_2/g(n_2)$	$t_2/h(n_2)$
$n_3$	$t_3$	$t_3/f(n_3)$	$t_3/g(n_3)$	$t_3/h(n_3)$
...	...	...	...	...
$n_m$	$t_m$	$t_m/f(n_m)$	$t_m/g(n_m)$	$t_m/h(n_m)$
		$f(n)$ es subestimada	$g(n)$ es ajustada <b>Cte = C</b>	$h(n)$ es sobrestimada

\*: tiempo promedio de  $K$  ejecuciones del algoritmo

**Tabla i:** estudio de la complejidad de  $\langle alg, caso \rangle$

**Discusión:** explicar dificultades, mediciones anómalas, cualquier duda en el análisis...

**Conclusión:**  $t(n) = O(g(n))$

## Presentación de los resultados (Cont.): precauciones

- las cotas utilizadas deben indicarse explícitamente (y en orden)
- las constantes ( $K$ ,  $C$ ) deben indicarse explícitamente  
→  *$C$  puede aproximarse mediante un intervalo*
- comprobar recomendaciones sobre  $n^0$  de mediciones, progresión de los valores de  $n$ , validez de los resultados. . .
- todos los números de las series calculadas deben tener al menos tres cifras significativas
- comprobar recomendaciones sobre contexto (introducción, unidades de tiempo. . .), discusión y conclusiones
- ¿dificultades en el análisis? → *repetir las mediciones varias veces y elegir las “mejores series”*

## Ejemplo:

$n$	$t(n)$		$t(n)/n^{1,8}$	$t(n)/n^2$	$t(n)/n^{2,2}$
500	49,633	*	0.0006881	0.0001985	0.0000573
1000	186,558	*	0.0007427	0.0001866	0.0000469
2000	754,000		0.0008620	0.0001885	0.0000412
4000	2862,000		0.0009396	0.0001789	0.0000341
8000	11390,000		0.0010739	0.0001780	0.0000295
16000	45582,000		0.0012342	0.0001781	0.0000257
32000	183722,000		0.0014285	0.0001794	0.0000225
			$n^{1,8}$ es subestimada	$n^2$ es ajustada Cte = 0.00018	$n^{2,2}$ es sobrestimada

\*: tiempo promedio (en  $\mu s$ ) de 1.000 ejecuciones del algoritmo

**Tabla 1:** estudio de la complejidad de Inserción, caso medio

**Discusión:** primeros valores

**Conclusión:**  $t(n) = O(n^2)$



# Consideraciones finales para las prácticas

- ¿Qué funciones utilizar?
  - *mirar bien el enunciado* - Cf. práctica 1
  - Algunas funciones características*
  - Valorar la progresión de las series, deben resultar útiles
- Evaluación
  - *plantilla para la corrección de las prácticas*
  - Criterios de calidad de los *programas* y del *informe de resultados*
  - Fórmula final
- Examen individual de prácticas