

# Algoritmos de ordenamiento

Anabell Sandoval, Andreu Boada de Atela, Fernando Aguilar,  
David Jaramillo

Análisis de Algoritmos

Agosto 2013

# Algoritmos de ordenamiento

- 1 Ordenación por inserción
- 2 Algoritmo de mezcla

# Ordenación por inserción

- Se va recorriendo el arreglo desde el segundo elemento de izquierda a derecha.
- Se intercambian dos a dos.

# Algoritmo de mezcla

- Es mas fácil ordenar dos arreglos de menor tamaño que uno más grande. A esta técnica se le llama "divide y vencerás."
- Se parte el arreglo tantas veces como sea necesario hasta llegar a tener arreglos de dos elementos.

# Implementación en C++

- ¿Cómo se programó?

# Implementación en C++

- Se hicieron 30 muestras aleatorias para tamaños del problema de  $n = 50h$  con  $h = 1, 2, \dots, 100$ .
- Construimos  $M_l \in \mathbb{R}^{n \times k}$  donde  $n$  es el tamaño del problema y  $k$  es el número de muestras.

Se calculan

$$\bar{m}_I = \frac{1}{n} \begin{pmatrix} \sum_{j=1}^k M_{I_1,j} \\ \sum_{j=1}^k M_{I_2,j} \\ \vdots \\ \sum_{j=1}^k M_{I_n,j} \end{pmatrix} \in \mathbb{R}^n$$

los promedios por tamaño de problema, donde  $M_{I_{ij}}$  es el tiempo de ejecución del problema de tamaño  $i$  correspondiente a la muestra  $j$  por el método de ordenamiento por inserción.

- Se calcula también  $\overline{m}_M$  utilizando el método de ordenamiento por mezcla.
- Se grafican los tiempos promedio tanto para el método de inserción y el de mezcla para todos los valores de  $n$ .



# Comparación por tiempo de ejecución

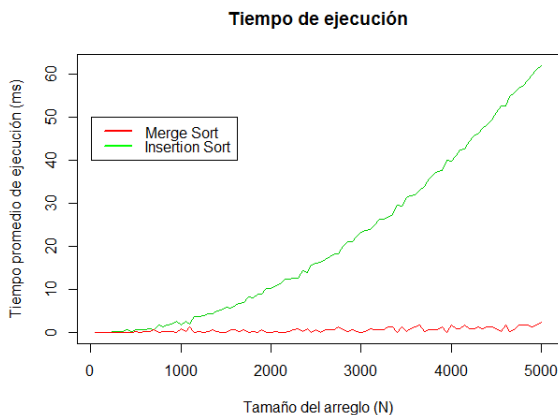
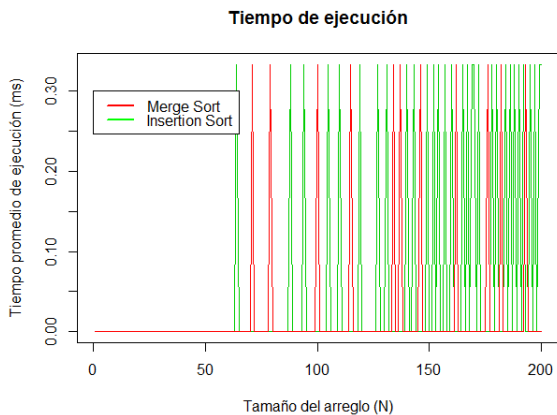


Figura : Tiempo de ejecución en milisegundos para  $n$

# C++ es muy rápido



**Figura :** Tiempo de ejecución en milisegundos para  $n$

Si se cuenta el número de comparaciones...

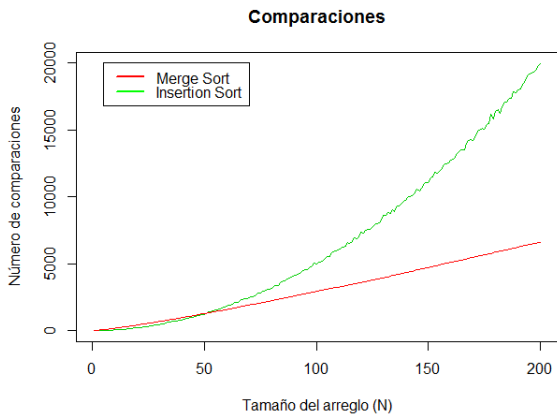


Figura : Tiempo de ejecución en milisegundos para  $n$

## Problema a)

- método de ordenación por inserción:  $O(2n^2)$ .
- método de ordenación por mezcla:  $O(64n \lg(n))$ .
- ¿Para qué valor de  $n$  el algoritmo de inserción corre más rápido que el de mezcla?

Se puede plantear la siguiente desigualdad:

$$2n^2 \leq 64n \lg(n).$$

Se puede fácilmente ver que  $n = 163$ .

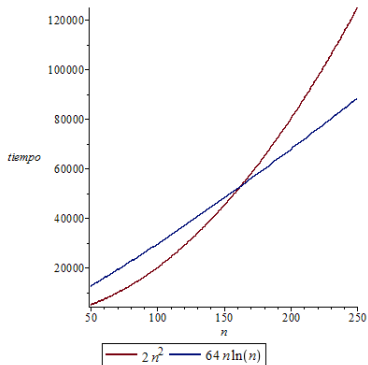


Figura : Comparación de funciones

Si se cuenta el número de comparaciones...

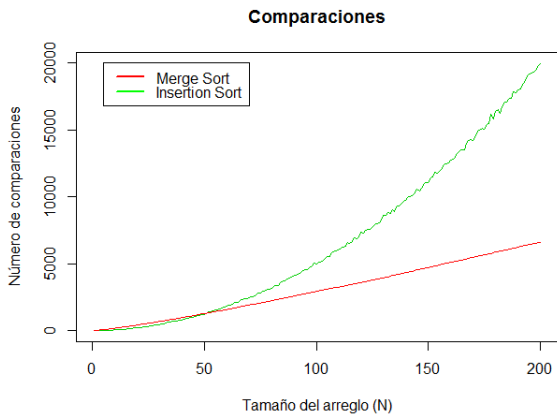


Figura : Tiempo de ejecución en milisegundos para  $n$

## Problema b)

- Un algoritmo cuyo tiempo de ejecución es  $100n^2$  corre más rápido que un algoritmo cuyo tiempo de ejecución es de  $2^n$ .
- Planteando nuevamente la desigualdad

$$100n^2 \leq 2^n$$

para  $n \in \mathbb{N}$  máxima se encuentra que  $n = 14$ .



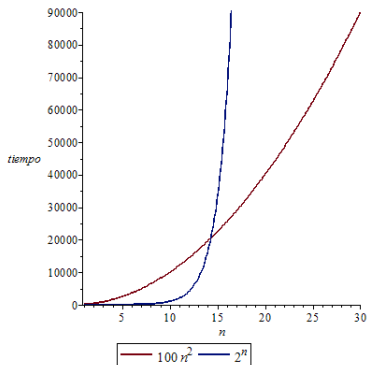


Figura : Comparación de funciones

## Problema c)

- Consideremos la función  $f(n) = \lg(n)$ .
- Valor máximo de  $n$  tal que el problema pueda ser resuelto en tiempo  $t$

Se tiene que 1 seg es igual a  $10^6$  microsegundos. Planteamos la ecuación

$$\lg(n) = f(n) = 10^6$$

y tomando exponencial de ambos lados obtenemos

$$n = \exp(10^6) \approx 3.033215397 \times 10^{434294}.$$

Se tienen las siguientes equivalencias:

	microsegundos
1 seg	$10^6$
1 min	$6 \times 10^7$
1 hora	$3.6 \times 10^9$
1 día	$8.64 \times 10^{11}$
1 mes	$2.592 \times 10^{13}$
1 año	$2.75 \times 10^{14}$
1 siglo	$2.75 \times 10^{16}$

$f(n)$	1 seg	1 min	1 hora	1 día	1 mes
$\lg(n)$	$3 \times 10^{434294}$	$8 \times 10^{26057668}$	$7 \times 10^{1563460134}$	$2 \times 10^{375230432364}$	$3 \times 10^{11291656529484}$
$\sqrt{n}$	$1 \times 10^{12}$	$3 \times 10^{15}$	$1 \times 10^{19}$	$7 \times 10^{23}$	$6 \times 10^{26}$
$n$	$10^6$	$6 \times 10^7$	$3.6 \times 10^9$	$8.64 \times 10^{11}$	$2.592 \times 10^{13}$
$n \lg(n)$	87847	3950157	188909174	$3 \times 10^{10}$	$9 \times 10^{11}$
$n^2$	1000	7745	60000	929516	5099019
$n^3$	100	391	1532	9524	29624
$2^n$	19	25	31	39	44
$n!$	9	11	12	14	15

Figura : Tamaño de problema para cada  $f(n)$  y tiempo  $t$

$f(n)$	1 año	1 siglo
$\lg(n)$	$1 \times 10^{119430982523394}$	$1 \times 10^{11943098252339425}$
$\sqrt{n}$	$7 \times 10^{28}$	$7 \times 10^{32}$
$n$	$2.75 \times 10^{14}$	$2.75 \times 10^{16}$
$n \lg(n)$	$9 \times 10^{12}$	$8 \times 10^{14}$
$n^2$	16583123	165831239
$n^3$	65029	301840
$2^n$	47	54
$n!$	16	18

Figura : Tamaño de problema para cada  $f(n)$  y tiempo  $t$

# Complejidad factorial

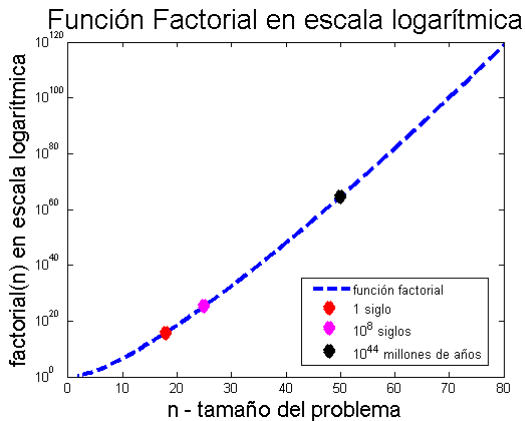


Figura : ¿Qué tan difícil es el factorial?