Algoritmos de ordenamiento

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

Agosto 2013

Problemas a resolver

a)

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

Se puede plantear la siguiente desigualdad:

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

Resolviendo la desigualdad para $n \in \mathbb{N}$ máxima se puede fácilmente ver que $n = 163$.
En la siguiente gráfica se muestran las funciones $2n^2$ y $64n\lg(n)$ para valores de $n = 50, ..., 230$. La intersección de ambas funciones ocurre cuando $n = 163$ con un tiempo de $t = 2(163^2) = 53138$.

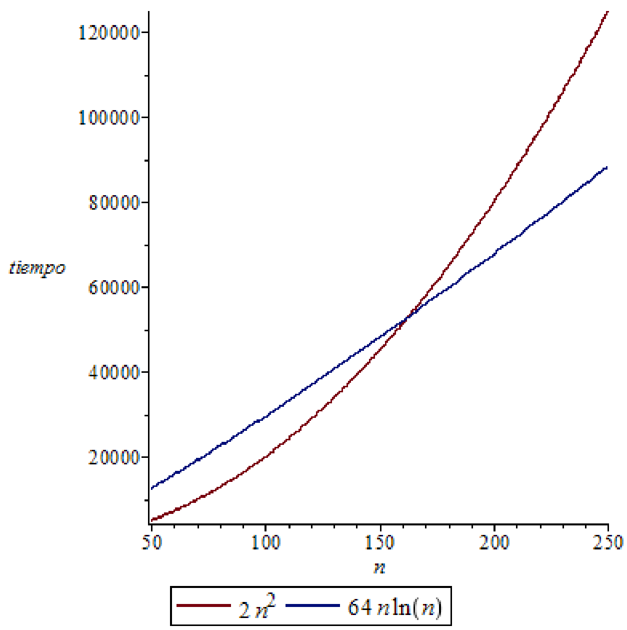

Figura 1: Comparación de funciones

b)

El problema consiste en encontrar el valor más pequeño de n tal que un algoritmo cuyo tiempo de ejecución es $100n^2$ corre más r ápido 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 $n = 14$.

En la siguiente gráfica se observan las dos funciones y la intersecci ón ocurre en $n = 14$ con un tiempo $t = 16384$.

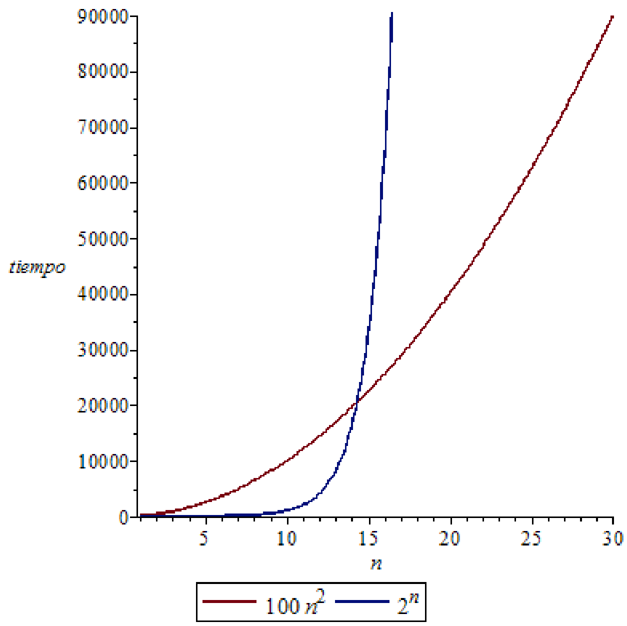

Figura 2: Comparación de funciones

c)

Se desea obtener la n máxima con la que se pueden resolver las distintas funciones $f(n)$ en un tiempo t.

Para obtener estos resultados se realizaron distintos cálculos en Maple, así como en Matlab.

Por ejemplo, consideremos la función $f(n) = \lg(n)$.

Se tiene que 1 seg es igual a 10^6 microsegundos. Planteamos la ecuaci ón

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

y tomando exponencial de ambos lados obtenemos

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

Se tienen las siguientes equivalencias:

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

Para cada función $f(n)$ y tiempo t, la siguiente tabla muestra el valor máximo de n tal que el problema pueda ser resuelto en tiempo t.

$f(n)$	1 seg	1 min	1 hora	1 día	1 mes
$\lg(n)$	3×10^{434294}	8×10^{26057668}	7×10^{1563460134}	$2 \times 10^{375230432364}$	$3 \times 10^{11291656529484}$
$\sqrt{n}$	1×10^{12}	3×10^{15}	1×10^{19}	7×10^{23}	6×10^{26}
n	10^6	6×10^7	3.6×10^9	8.64×10^{11}	2.592×10^{13}
$n \lg(n)$	87847	3950157	188909174	3×10^{10}	9×10^{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

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

Resolver un problema con complejidad factorial es muy costoso. En la gráfica siguiente se da una idea de qué tan complejo es este problema.

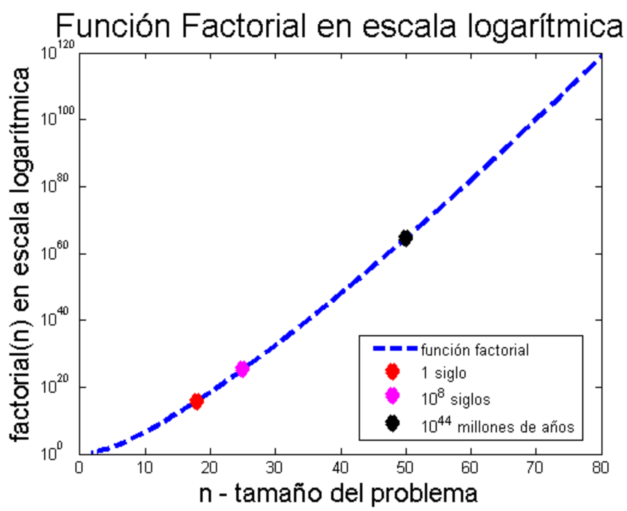

Figura 3: ¿Qué tan difícil es el factorial?

Algoritmos de Ordenamiento

Los algoritmos de ordenamiento que se van a comparar son:

1. Ordenamiento por inserción
2. Ordenamiento por mezcla

Ordenamiento por inserción

Este algoritmo iterativo es muy sencillo. Se va recorriendo el arreglo desde el segundo elemento de izquierda a derecha. En cada uno de estos recorridos, se compara con cada uno de los elementos que tiene a su izquierda. Mientras cada elemento sea mayor o igual al elemento que se tomó, se intercambian. Se podría decir que va creando subarreglos y los va ordenando cada vez.

Ordenamiento por mezcla

Este algoritmo involucra recursión en su proceso, pues se trata de partir tantas veces como sea necesario el arreglo hasta llegar a tener arreglos de dos elementos.

A partir de aquí, por cada recursión, se deben ordenar ambos arreglos y mezclarlos ya ordenados en el arreglo original.

La eficiencia de este algoritmo se basa en que es mas fácil ordenar dos arreglos de menor tamaño que uno más grande. A esta técnica se le llama "divide y vencerás."

Implementación en C++

Ambos algoritmos se implementaron en C++ utilizando Visual C++ Express 2010. Se hicieron 30 muestras aleatorias para tamañ os del problema de $n = 50h$ con $h = 1, 2, \ldots, 100$.

Construimos $M_I \in \mathbb{R}^{n\times k}$ donde n es el tamaño del problema y k es el nú mero de muestras.

Se calculan

$$\overline{m}_I = \frac{1}{n}\begin{pmatrix} \sum_{j=1}^{k} M_{I_{1,j}} \\ \sum_{j=1}^{k} M_{I_{2,j}} \\ \vdots \\ \sum_{j=1}^{k} M_{I_{n,j}} \end{pmatrix} \in \mathbb{R}^n$$

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.

Las siguentes gráficas muestran un análisis comparativo entre ambos m étodos.

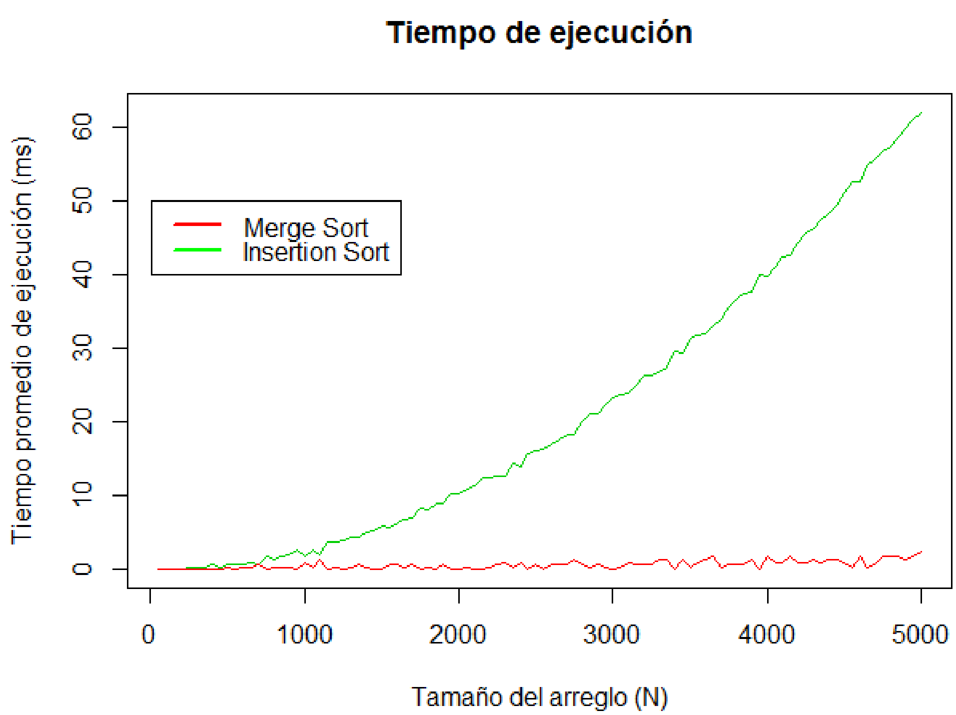

Figura 4: Inserción o Mezcla

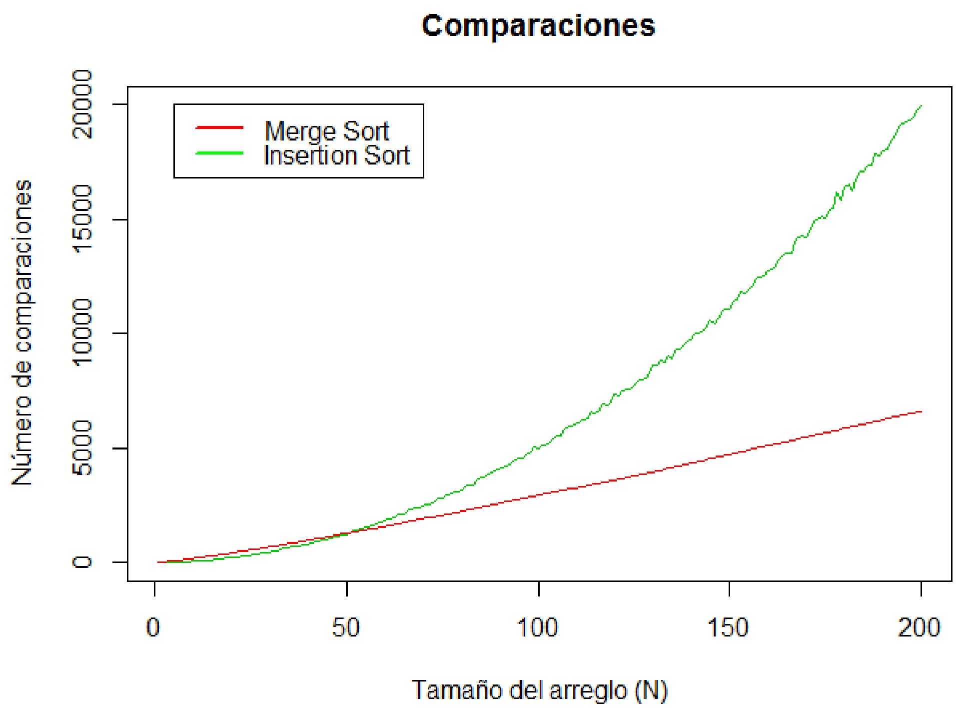

Figura 5: Inserción o Mezcla

Bibliografía

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein C. (2009) *Introduction to Algorithms.* Massachusetts Institute of Technology, 3rd edition.