Algorítmica: práctica 1

Análisis de la eficiencia de algoritmos

Sofía Almeida Bruno
Antonio Coín Castro
María Victoria Granados Pozo
Miguel Lentisco Ballesteros
José María Martín Luque

16 de marzo de 2017

Introducción

El objetivo de esta práctica es estudiar la eficiencia de los algoritmos que se nos proponen. Para ello realizaremos un estudio teórico, empírico e híbrido de cada uno. El estudio teórico consiste en expresar el número de operaciones $T(n)$ requeridas para un problema concreto en función del tamaño n, siempre en el *caso peor*. Para el estudio empírico hemos medido los tiempos de ejecución de cada algoritmo para cada uno de los tamaños de las entradas. Para el estudio híbrido, la idea es determinar las constantes ocultas en la expresión de la eficiencia teórica, y comprobar mediante un ajuste por mínimos cuadrados si coincide con la eficiencia empírica.

Algoritmos de ordenación

Burbuja

Revisa cada elemento de la lista con el siguiente, intercambiándose de posición si no están en el orden correcto. Su eficiencia teórica es $O(n^2)$.

Insercción

Consideramos el elemento N-ésimo de la lista y lo ordenamos respecto de los elementos desde el primero hasta el N-1-ésimo. Su eficiencia teórica es $O(n^2)$.

Selección

Consiste en encontrar el menor de todos los elementos de la lista e intercambiarlo con el de la primera posición. Luego con el segundo, y así sucesivamente hasta ordenarlo todo. De nuevo, su eficiencia teórica es $O(n^2)$.

Mergesort

Se basa en la técnica de divide y vencerás. Consiste en dividir la lista en sublistas de la mitad de tamaño, ordenando cada una de ellas de forma recursiva. Si el tamaño de la lista es 0 o 1 la lista ya está ordenada. Para acabar juntamos todas las sublistas en una sola. Su eficiencia teórica es $O(n \log n)$.

Quicksort

También se basa en la técnica de divide y vencerás. En primer lugar elegimos un elemento de la lista, que llamaremos *pivote*. A continución los elementos de la lista se ordenarán de forma que la derecha del pivote queden los mayores y a la izquierda los

menores. De esta forma dividimos la lista en dos sublistas, la de la derecha y la de la izquierda. Repetiremos el proceso mientras las sublistas tengan más de un elemento. Su eficiencia teórica también es $O(n \log n)$.

Heapsort

Este algoritmo consiste en almacenar todos los elementos del vector a ordenar en una estructura de datos llamada montículo (*heap*). Luego, se extrae el nodo que queda como nodo raíz del montículo (cima) en sucesivas iteraciones obteniendo el conjunto ordenado. Basa su funcionamiento en una propiedad de los montículos, por la cual, la cima contiene siempre el menor elemento (o el mayor, según se haya definido el montículo) de todos los almacenados en él. Su eficiencia teórica es $O(n \log n)$.

Otros algoritmos

Floyd

Es un algoritmo de análisis sobre grados para encontrar el camino mínimo en grafos ponderados. El algoritmo compara todos los posibles caminos a través del grafo entre cada par de vértices. Es un ejemplo de **programación dinámica**, y su eficiencia teórica es $O(n^3)$.

Hanoi

Hay tres pilas de discos, llamadas origen, auxiliar y destino. La primera de ellas está ordenada según tamaño creciente de los discos, de arriba hacia abajo. Se moverá un disco de la pila origen a la destino si hay un único disco en la pila origen. En caso contrario, se moverán todos los discos a la auxiliar, excepto el más grande. Por último, moveremos el disco mayor al destino, y movemos los $n-1$ restantes encima del mayor. El número de pasos crece exponencialmente con el número de discos, y su eficiencia teórica es $O(2^n)$.

Cálculo de la eficiencia empírica

Puesto que la eficiencia teórica de cada algoritmo es diferente, no podemos realizar las mediciones para los mismos valores de entrada en todos los algoritmos. Así, los agrupamos según su orden de eficiencia.

Para el cálculo de la eficiencia empírica, hemos utilizado un *script* que realiza tantas ejecuciones de cada algoritmo como le indiquemos, tomando como parámetros el valor inicial, el incremento, y el valor final de los datos de entrada.

Tablas

Las tablas que hemos obtenido tras realizar las mediciones son las siguientes.

Algoritmos que son $O(n^2)$ (tiempos en segundos)

Elementos	Burbuja	Selección	Inserción
2,000	$8{,}02 \cdot 10^{-3}$	$5{,}4 \cdot 10^{-3}$	$4{,}21 \cdot 10^{-3}$
4,000	$3{,}5 \cdot 10^{-2}$	$2{,}17 \cdot 10^{-2}$	$1{,}74 \cdot 10^{-2}$
6,000	$8{,}93 \cdot 10^{-2}$	$4{,}84 \cdot 10^{-2}$	$3{,}87 \cdot 10^{-2}$
8,000	0,16	$8{,}52 \cdot 10^{-2}$	$6{,}94 \cdot 10^{-2}$
10,000	0,26	0,13	0,11
12,000	0,39	0,19	0,15
14,000	0,55	0,26	0,21
16,000	0,73	0,34	0,32
18,000	0,93	0,43	0,38
20,000	1,18	0,52	0,42
22,000	1,44	0,63	0,51
24,000	1,71	0,76	0,61
26,000	2,02	0,89	0,72
28,000	2,35	1,03	0,82
30,000	2,72	1,18	0,94
32,000	3,1	1,34	1,07
34,000	3,53	1,52	1,21
36,000	3,95	1,71	1,42
38,000	4,4	1,9	1,57
40,000	4,89	2,1	1,7
42,000	5,39	2,33	1,88
44,000	5,94	2,54	2,17
46,000	6,52	2,79	2,26
48,000	7,11	3,03	2,61
50,000	7,69	3,3	2,67

Algoritmos que son $O(nlog(n))$ (tiempos en segundos)

Elementos	Mergesort	Quicksort	Heapsort
5,000	$8{,}51 \cdot 10^{-4}$	$5{,}32 \cdot 10^{-4}$	$6{,}97 \cdot 10^{-4}$
10,000	$1{,}72 \cdot 10^{-3}$	$1{,}13 \cdot 10^{-3}$	$1{,}51 \cdot 10^{-3}$
15,000	$2{,}49 \cdot 10^{-3}$	$1{,}76 \cdot 10^{-3}$	$2{,}37 \cdot 10^{-3}$
20,000	$3{,}8 \cdot 10^{-3}$	$2{,}5 \cdot 10^{-3}$	$3{,}83 \cdot 10^{-3}$
25,000	$5{,}23 \cdot 10^{-3}$	$3{,}18 \cdot 10^{-3}$	$4{,}31 \cdot 10^{-3}$
30,000	$5{,}38 \cdot 10^{-3}$	$3{,}78 \cdot 10^{-3}$	$5{,}82 \cdot 10^{-3}$
35,000	$7{,}45 \cdot 10^{-3}$	$4{,}44 \cdot 10^{-3}$	$6{,}1 \cdot 10^{-3}$
40,000	$8{,}24 \cdot 10^{-3}$	$5{,}19 \cdot 10^{-3}$	$7{,}22 \cdot 10^{-3}$
45,000	$9{,}78 \cdot 10^{-3}$	$6{,}32 \cdot 10^{-3}$	$9{,}49 \cdot 10^{-3}$
50,000	$1{,}16 \cdot 10^{-2}$	$6{,}49 \cdot 10^{-3}$	$9{,}79 \cdot 10^{-3}$
55,000	$1{,}01 \cdot 10^{-2}$	$7{,}75 \cdot 10^{-3}$	$1{,}05 \cdot 10^{-2}$
60,000	$1{,}14 \cdot 10^{-2}$	$8{,}36 \cdot 10^{-3}$	$1{,}2 \cdot 10^{-2}$
65,000	$1{,}28 \cdot 10^{-2}$	$8{,}71 \cdot 10^{-3}$	$1{,}26 \cdot 10^{-2}$
70,000	$1{,}43 \cdot 10^{-2}$	$9{,}37 \cdot 10^{-3}$	$1{,}34 \cdot 10^{-2}$
75,000	$1{,}57 \cdot 10^{-2}$	$1{,}02 \cdot 10^{-2}$	$1{,}45 \cdot 10^{-2}$
80,000	$1{,}74 \cdot 10^{-2}$	$1{,}13 \cdot 10^{-2}$	$1{,}63 \cdot 10^{-2}$
85,000	$1{,}87 \cdot 10^{-2}$	$1{,}18 \cdot 10^{-2}$	$1{,}68 \cdot 10^{-2}$
90,000	$2{,}02 \cdot 10^{-2}$	$1{,}28 \cdot 10^{-2}$	$1{,}81 \cdot 10^{-2}$
95,000	$2{,}21 \cdot 10^{-2}$	$1{,}33 \cdot 10^{-2}$	$1{,}92 \cdot 10^{-2}$
$1 \cdot 10^5$	$2{,}4 \cdot 10^{-2}$	$1{,}39 \cdot 10^{-2}$	$2{,}06 \cdot 10^{-2}$
$1{,}05 \cdot 10^5$	$2{,}05 \cdot 10^{-2}$	$1{,}5 \cdot 10^{-2}$	$2{,}1 \cdot 10^{-2}$
$1{,}1 \cdot 10^5$	$2{,}21 \cdot 10^{-2}$	$1{,}55 \cdot 10^{-2}$	$2{,}25 \cdot 10^{-2}$
$1{,}15 \cdot 10^5$	$2{,}29 \cdot 10^{-2}$	$1{,}63 \cdot 10^{-2}$	$2{,}38 \cdot 10^{-2}$
$1{,}2 \cdot 10^5$	$2{,}45 \cdot 10^{-2}$	$1{,}68 \cdot 10^{-2}$	$2{,}48 \cdot 10^{-2}$
$1{,}25 \cdot 10^5$	$2{,}61 \cdot 10^{-2}$	$1{,}81 \cdot 10^{-2}$	$2{,}6 \cdot 10^{-2}$
$1{,}3 \cdot 10^5$	$2{,}79 \cdot 10^{-2}$	$1{,}85 \cdot 10^{-2}$	$2{,}72 \cdot 10^{-2}$
$1{,}35 \cdot 10^5$	$2{,}88 \cdot 10^{-2}$	$1{,}93 \cdot 10^{-2}$	$2{,}84 \cdot 10^{-2}$
$1{,}4 \cdot 10^5$	$3{,}03 \cdot 10^{-2}$	$2{,}03 \cdot 10^{-2}$	$3{,}01 \cdot 10^{-2}$
$1{,}45 \cdot 10^5$	$3{,}12 \cdot 10^{-2}$	$2{,}08 \cdot 10^{-2}$	$3{,}06 \cdot 10^{-2}$

Algoritmos que son $O(n^3)$ (tiempos en segundos)

Elementos	Floyd
50	$7{,}41 \cdot 10^{-4}$
100	$6{,}13 \cdot 10^{-3}$
150	$2{,}05 \cdot 10^{-2}$
200	$4{,}5 \cdot 10^{-2}$
250	$8{,}92 \cdot 10^{-2}$
300	0,15
350	0,24
400	0,35
450	0,5
500	0,68
550	0,91
600	1,17
650	1,49
700	1,9
750	2,34
800	2,88
850	3,44
900	4,07
950	4,79
1,000	5,57
1,050	6,43
1,100	7,44
1,150	8,51
1,200	9,63
1,250	10,87
1,300	12,27
1,350	13,73
1,400	15,51
1,450	16,79

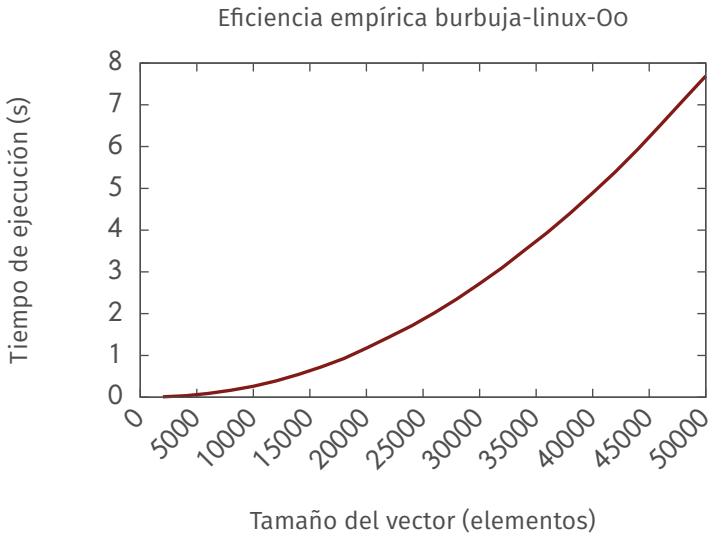
Algoritmos que son $O(2^n)$ (tiempos en segundos)

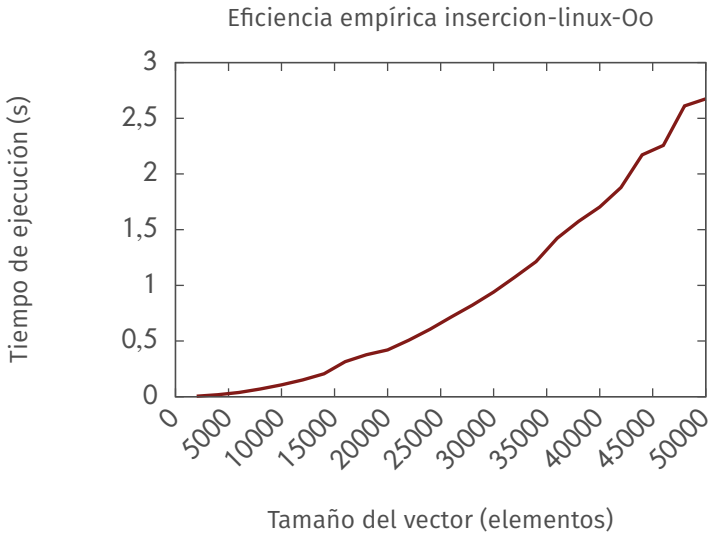
Elementos	Hanoi
5	$1 \cdot 10^{-6}$
6	$1 \cdot 10^{-6}$
7	$1 \cdot 10^{-6}$
8	$2 \cdot 10^{-6}$
9	$4 \cdot 10^{-6}$
10	$7 \cdot 10^{-6}$
11	$1{,}3 \cdot 10^{-5}$
12	$2{,}7 \cdot 10^{-5}$
13	$5{,}1 \cdot 10^{-5}$
14	$1 \cdot 10^{-4}$
15	$2 \cdot 10^{-4}$
16	$4{,}37 \cdot 10^{-4}$
17	$8{,}23 \cdot 10^{-4}$
18	$1{,}58 \cdot 10^{-3}$
19	$3{,}17 \cdot 10^{-3}$
20	$6{,}45 \cdot 10^{-3}$
21	$1{,}41 \cdot 10^{-2}$
22	$2{,}58 \cdot 10^{-2}$
23	$5{,}24 \cdot 10^{-2}$
24	0,1
25	0,21
26	0,41
27	0,81
28	1,62
29	3,26
30	6,51

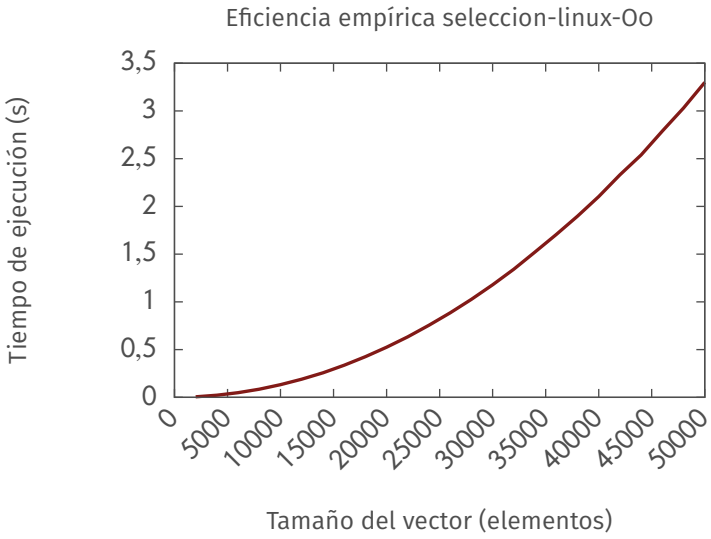
Gráficos

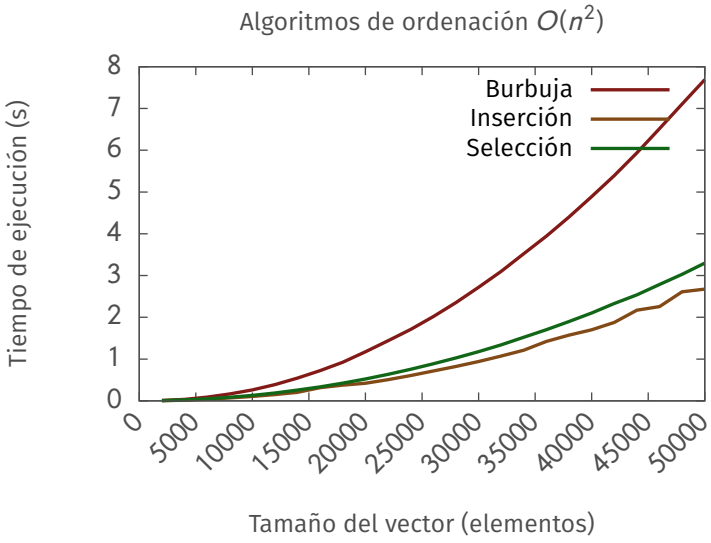
A continuación se muestran los gráficos que corresponden a las tablas anteriores. Se ha realizado un gráfico para cada algoritmo, así como otros donde se pueden visualizar varios algoritmos.

Algoritmos que son $O(n^2)$

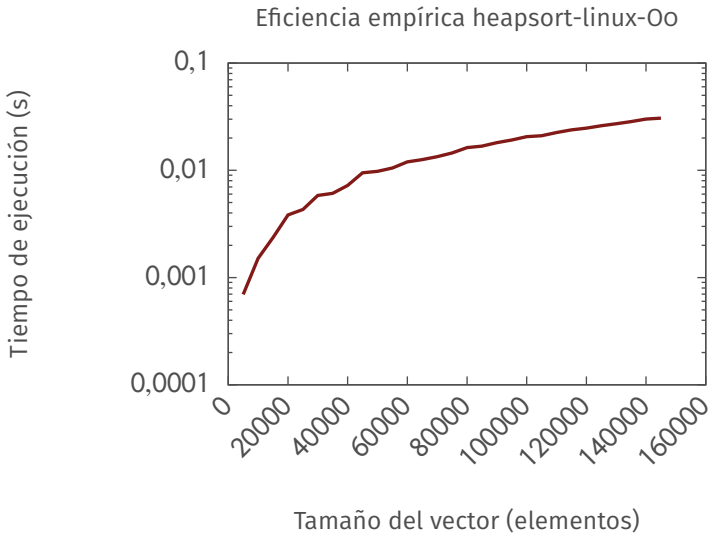


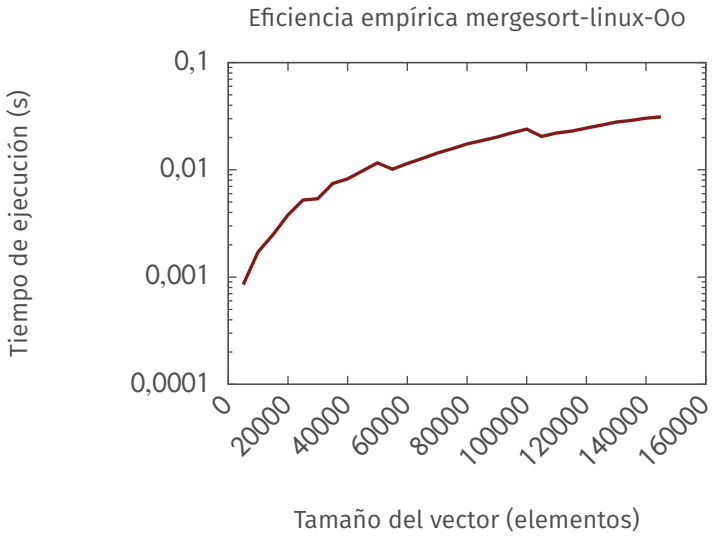


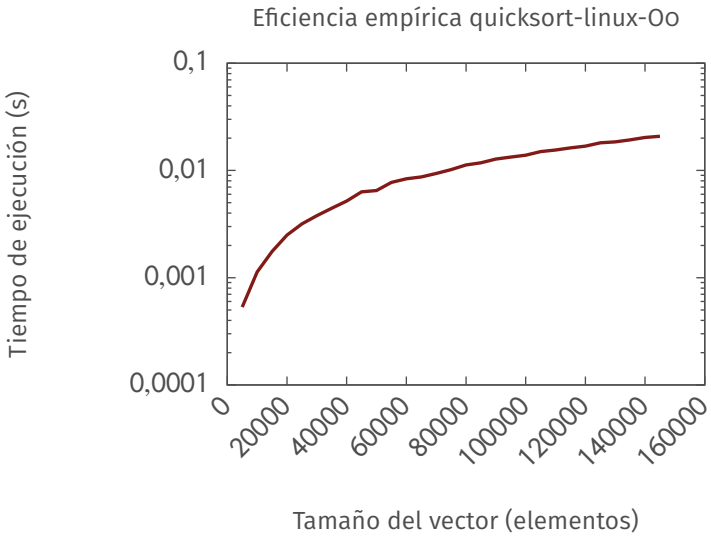

Eficiencia empírica seleccion-linux-O0
Tiempo de ejecución (s)
3,5
3
2,5
2
1,5
1
0,5
0
0
5000
10000
15000
20000
25000
30000
35000
40000
45000
50000
Tamaño del vector (elementos)

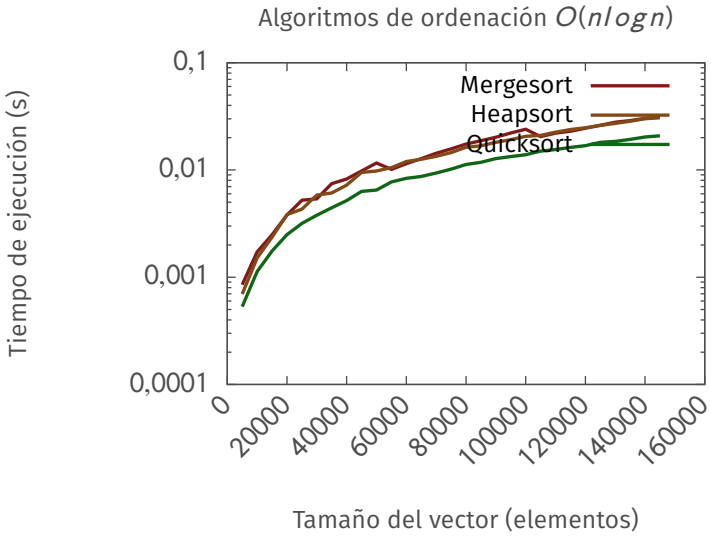

Algoritmos de ordenación $O(n^2)$
Tiempo de ejecución (s)
8
7
6
5
4
3
2
1
0
Burbuja
Inserción
Selección
0
5000
10000
15000
20000
25000
30000
35000
40000
45000
50000
Tamaño del vector (elementos)

Algoritmos que son $O(n \log n)$

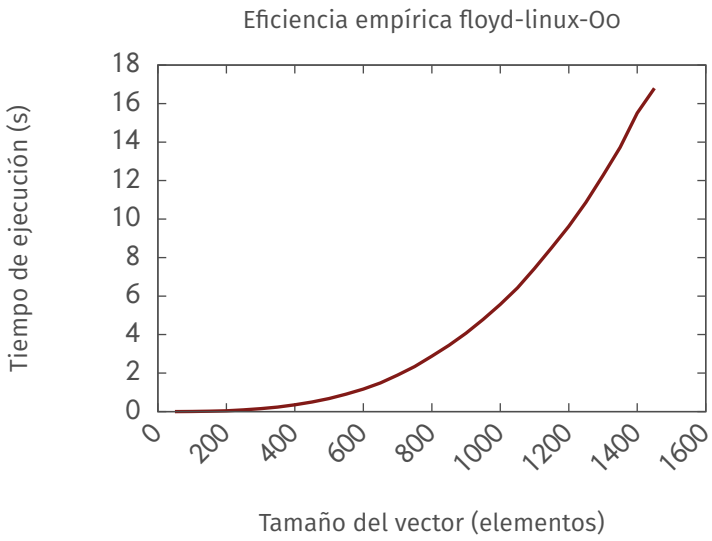




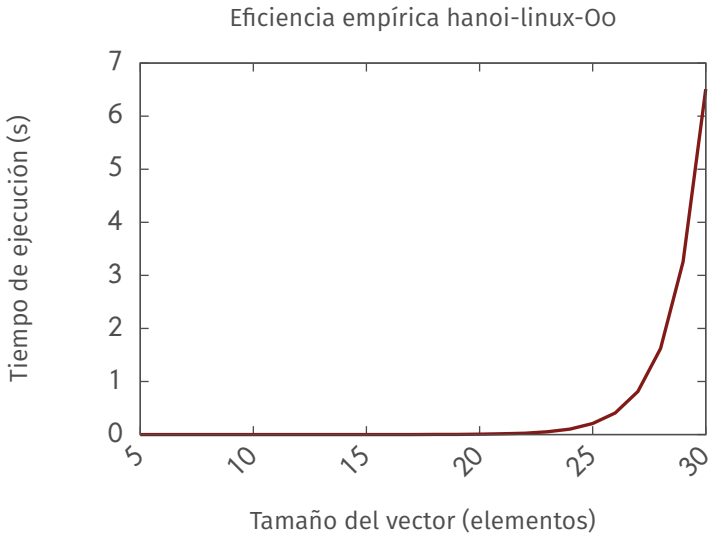

Eficiencia empírica quicksort-linux-O0
Tiempo de ejecución (s)
0,1
0,01
0,001
0,0001
0
20000
40000
60000
80000
100000
120000
140000
160000
Tamaño del vector (elementos)


Algoritmos de ordenación O(nlogn)
Tiempo de ejecución (s)
0,1
0,01
0,001
0,0001
Mergesort
Heapsort
Quicksort
0
20000
40000
60000
80000
100000
120000
140000
160000
Tamaño del vector (elementos)

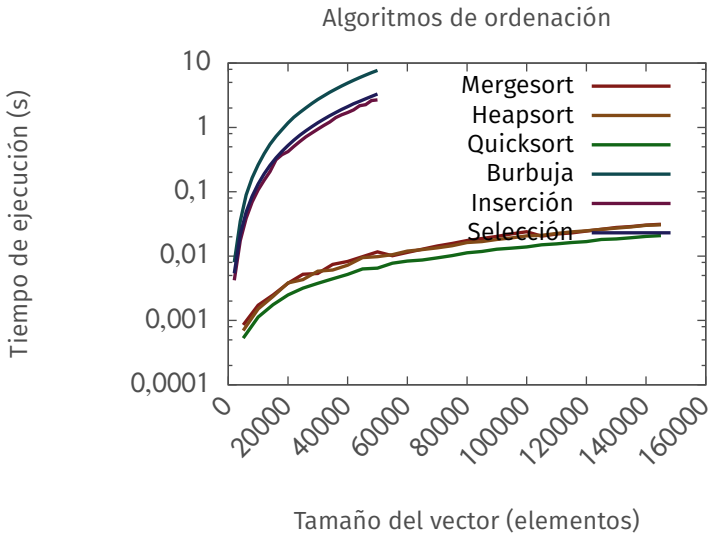
Algoritmos que son $O(n^3)$



Algoritmos que son $O(2^n)$



Algoritmos de ordenación



En este último gráfico, puede observarse claramente la tendencia de los algoritmos cuyo orden de eficiencia es $O(n \log n)$ a ser más rápidos que aquellos que son $O(n^2)$. Sin embargo, no puede apreciarse totalmente en el gráfico, puesto que los valores de entrada son distintos para ambos grupos de algoritmos.