Práctica 1: Comparación de Eficiencia de Algoritmos

Irene Huertas González
Javier Alcántara García
César Muñoz Reinoso

Índice

Máquinas empleadas

Antes de empezar con la comparación de eficiencia es necesario ver qué máquinas se han empleado para esta:

1. HP Pavilion, 2 año:
   - **Procesador:** Intel Core i7, 6ª generación.
   - **Memoria RAM:** 12,0 GB.
   - **Tipo de sistema:** Sistema operativo de 64 bits, procesador x64.
2. Dell XPS 15, 0.5 años:
   - **Procesador:** Intel Core i7, 7ª generación.
   - **Memoria RAM:** 8,0 GB.
   - **Tipo de sistema:** Sistema operativo de 64bits, procesador x64.
3. Xiaomi Mi NoteBook Pro, 1.5 años:
   - **Procesador:** Intel Core i7, 8ª generación.
   - **Memoria RAM:** 16,0 GB.
   - **Tipo de sistema:** Sistema operativo de 64bits, procesador x64.

Algoritmo 1 - teórica
Peor caso: que el pivote corresponda a un extremo. El vector está casi ordenado

$$T(n) = 2+3+1+\sum_{i=1}^{i>j}(\sum_{i=ini}^{n-t}(6)+\sum_{j=fin}^{t}(6)+max(7,0+1))+max(6,0)+1 =$$

$$6+\sum_{i}^{i>j}(\sum_{i}^{n-t}(6)+\sum_{j}^{t}(6)+8)+7 = 6+\sum_{i}^{i>j}(6(n-t-i)+6(t-j)+8)+7)$$

$$T(n) = 6+6(n-t-i)+6(t-j)+15$$

Concluimos que el orden de eficiencia es $O(n)$, eficiencia lineal

Algoritmo 1 - empírica

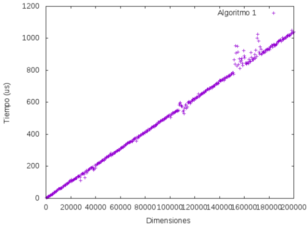

Figura: Algoritmo 1 Javier

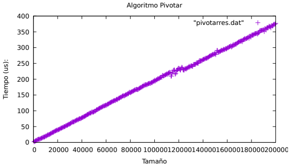

Figura: Algoritmo 1 César

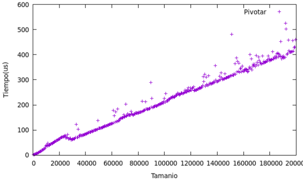

Figura: Algoritmo 1 Irene

Algoritmo 1 - híbrida

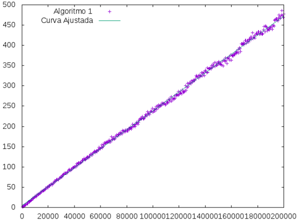

Figura: Algoritmo 1 Javier

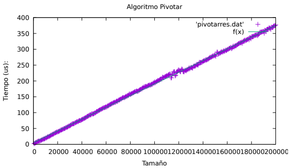

Figura: Algoritmo 1 César

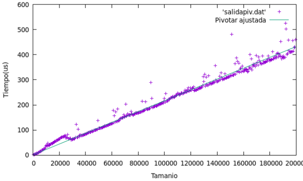

Figura: Algoritmo 1 Irene

Algoritmo 2 - teórica

Peor caso: Que no se encuentre el elemento que buscamos.

$$T(n) = 6 + 4 + \sum_{ini,fin}^{ini>fin \text{ ó } encontrado} (max(4,2) + 3 + 4) + max(2,0) + 1 =$$

$$10 + \sum_{0}^{n/2} (4 + 7) + 3 = 10 + 11\frac{n}{2} + 3$$

El orden de eficiencia es $O(log_2(n))$

Algoritmo 2 - empírica

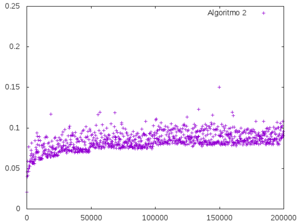

Figura: Algoritmo 2 Javier

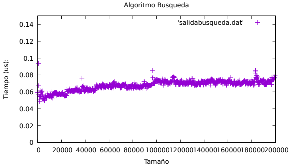

Figura: Algoritmo 2 César

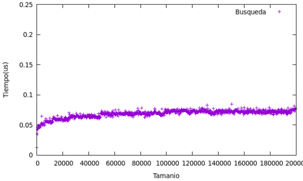

Figura: Algoritmo 2 Irene

Algoritmo 2 - híbrida

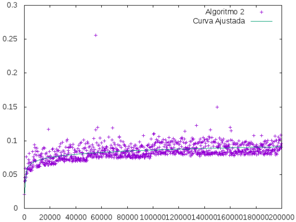

Figura: Algoritmo 2 Javier

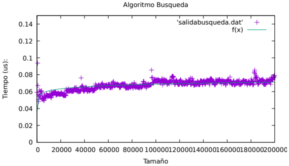

Figura: Algoritmo 2 César

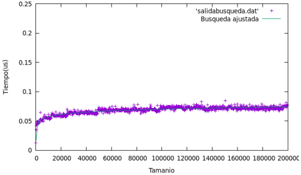

Figura: Algoritmo 2 Irene

Algoritmo 3 - teórica Peor caso se da cuando no hay repetidos.

$$T(n) = \sum_{i=0}^{nOriginal} (2 + \sum_{i+1}^{nOriginal} (max((3 + \sum_{i+2}^{nOriginal} 8 + 1), 1) + 1)) =$$

$$\sum_{i=0}^{n} (2 + \sum_{i+1}^{n} (1) + 1) =$$

$$T(n) = \sum_{i=0}^{n} (2 + 1(n-1) + 1) = n(3 + (n-1)) = n^2 - n + 3$$

Se nos queda asi un orden de eficiencia cuadratico $O(n^2)$

Algoritmo 3 - empírica

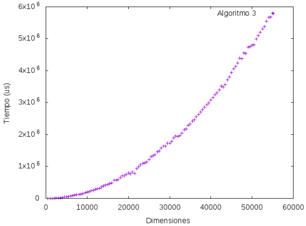

Figura: Algoritmo 3 Javier

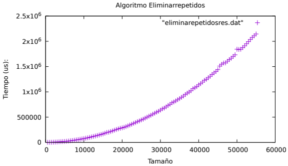

Figura: Algoritmo 3 César

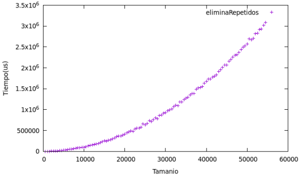

Figura: Algoritmo 3 Irene

Algoritmo 3 - híbrida

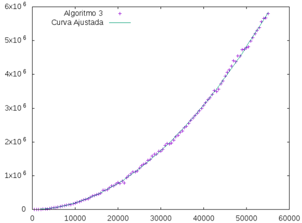

Figura: Algoritmo 3 Javier

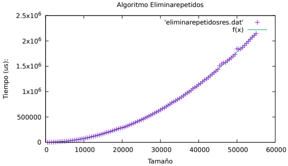

Figura: Algoritmo 3 César

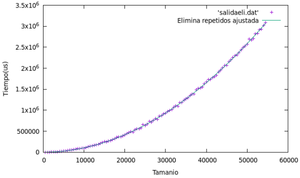

Figura: Algoritmo 3 Irene

Algoritmo 4 - teórica

El peor caso se da cuando el elemento a buscar no se encuentra en el vector, es decir, cuando tras dividir los elementos por analizar nos quedemos con un número menor a 1.

- n <= 1

$$T(n) = 1$$

- n >1

$$T(n) = i * (7 + T(n/2))$$

siendo i el número de iteraciones

Es por esto que el algoritmo de busqueda binaria tiene una complejidad de orden logarítmico $O(logn)$.

Algoritmo 4 - empírica

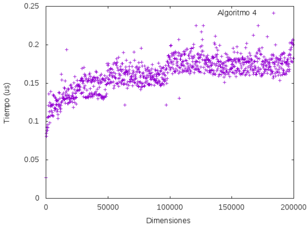

Figura: Algoritmo 4 Javier

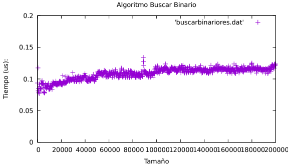

Figura: Algoritmo 4 César

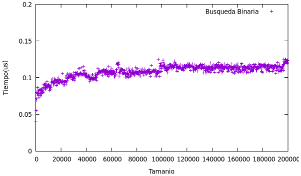

Figura: Algoritmo 4 Irene

Algoritmo 4 - híbrida

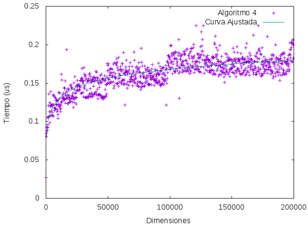

Figura: Algoritmo 4 Javier

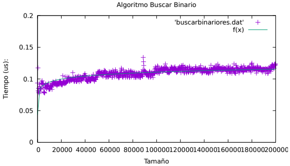

Figura: Algoritmo 4 César

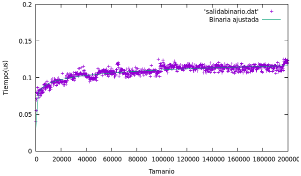

Figura: Algoritmo 4 Irene

Algoritmo 5 - teórica : $O(nlog(n))$

Algoritmo 5 - empírica

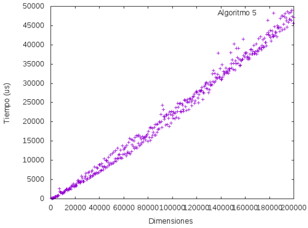

Figura: Algoritmo 5 Javier

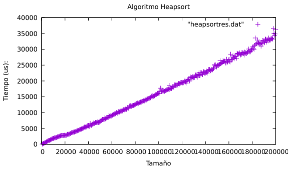

Figura: Algoritmo 5 César

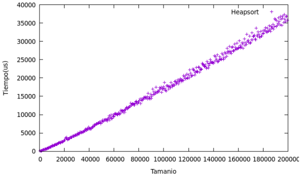

Figura: Algoritmo 5 Irene

Algoritmo 5 - híbrida

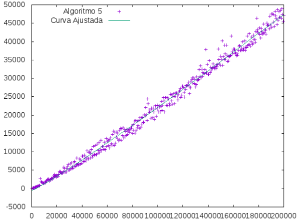

Figura: Algoritmo 5 Javier

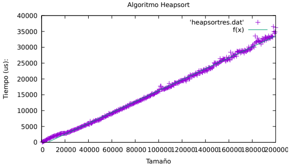

Figura: Algoritmo 5 César

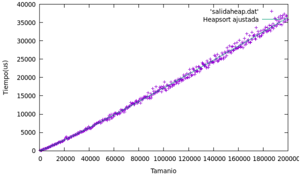

Figura: Algoritmo 5 Irene

Algoritmo Burbuja - teórica : $O(n^2)$

Algoritmo Burbuja - empírica

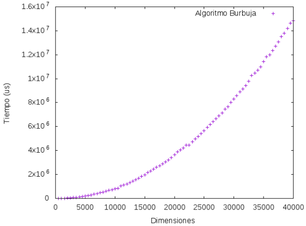

Figura: Burbuja Javier

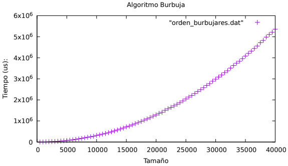

Figura: Burbuja César

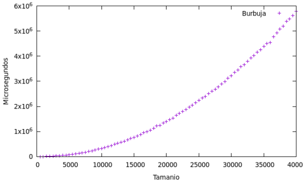

Figura: Burbuja Irene

Algoritmo Burbuja - híbrida

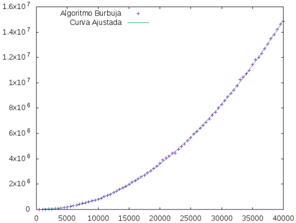

Figura: Burbuja Javier

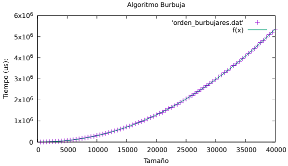

Figura: Burbuja César

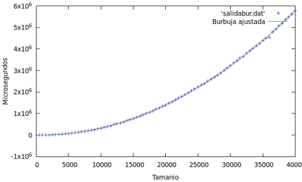

Figura: Burbuja Irene

Algoritmo Mergesort - teórica : $O(nlog(n))$

Algoritmo Mergesort - empírica

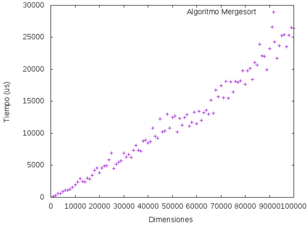

Figura: Mergesort Javier

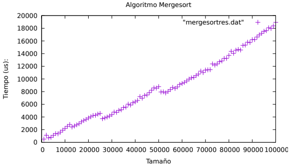

Figura: Mergesort César

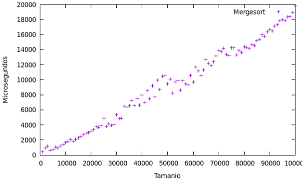

Figura: Mergesort Irene

Algoritmo Mergesort - híbrida

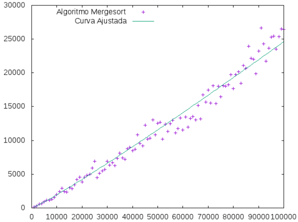

Figura: Mergesort Javier

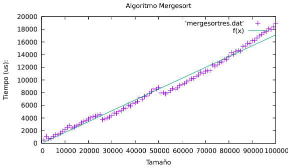

Figura: Mergesort César

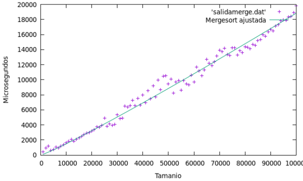

Figura: Mergesort Irene

Algoritmo Hanoi - teórica : $O(2^n)$

Algoritmo Hanoi - empírica

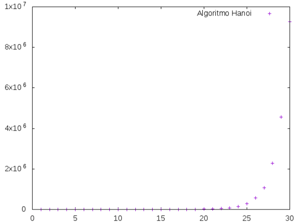

Figura: Hanoi Javier

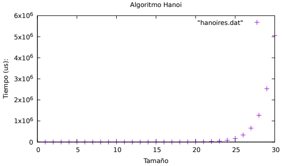

Figura: Hanoi César

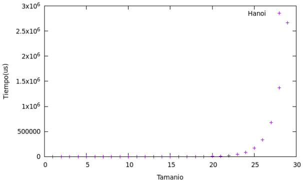

Figura: Hanoi Irene

Algoritmo Hanoi - híbrida

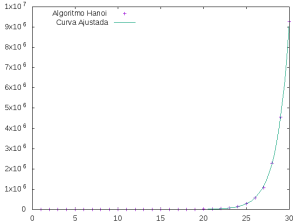

Figura: Hanoi Javier

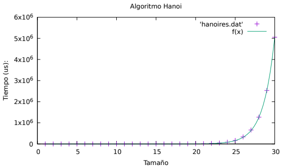

Figura: Hanoi César

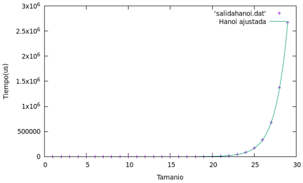

Figura: Hanoi Irene

Constantes K

IRENE
alg1: 0.0065
alg2: 0.02685741
alg3: 0.004957
alg4: 0.04651423
alg5: 0.2245189
burbuja: 0.003512
mergesort: 0.6547
hanoi: 0.0641523

JAVIER
alg1: 0.0058
alg2: 0.016234929
alg3: 0.0023218503
alg4: 0.0323592918
alg5: 0.1091367311
burbuja: 0.0046406
mergesort: 0.06282
hanoi: 0.016113281

CESAR
alg1: 0.0051
alg2: 0.02547895
alg3: 0.004884
alg4: 0.04030698
alg5: 0.22781818
burbuja: 0.00236641
mergesort: 0.558471
hanoi: 0.062551

Conclusión

Tras realizar esta práctica hemos llegado a las siguientes conclusiones:

1. Un código ejecutado en distintas máquinas siempre va a tener resultados de ejecución distintos.
2. Un código ejecutado en una misma máquina siempre va a tardar menos en ejecutarse si se ha compilado con optimización.
3. Una máquina menos eficiente que otra puede llegar a superarla con algoritmos optimizados.