

Práctica 1. Análisis de Eficiencia de Algoritmos

Algorítmica

Shao Jie Hu Chen Mario Megías Mateo Jesús Samuel García Carballo

Equipo Rojo

1 de abril de 2022

Índice de contenidos

- 1 Introducción
- 2 Análisis de Eficiencia General
- 3 Análisis de Casos Particulares
- 4 Conclusiones
- 5 Referencias

Objetivos

Los objetivos de esta práctica son los siguientes:

- **Estudio teórico, empírico e híbrido y comparación** de los algoritmos de ordenación más empleados, verificando los resultados teóricos.
- **Estudio teórico, empírico e híbrido** de algoritmos de alta complejidad, poniendo especial énfasis en su **viabilidad** en diferentes equipos.
- Estudio del **aumento de eficiencia** de un mismo algoritmo para diferentes **tipos de optimización** del compilador.
- Determinación del algoritmo **más adecuado** para cada situación en función del estado de los datos.

Equipo

ASUS

- **Modelo:** ZenBook 15 UX534F
- **Procesador:** Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz5
- **Memoria Ram:** 16 GB DDR4 @ 2.133 MHz.
- **Sistema Operativo** Ubuntu 20.04.2 LTS

HP

- **Modelo:** Pavilion Gaming Laptop 15-dk0xxx
- **Procesador:** Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
- **Memoria RAM:** 32 GB DDR4
- **Sistema Operativo:** Ubuntu 20.04.4 LTS

LENOVO

- **Modelo:** YOGA 530-14IKB
- **Procesador:** Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
- **Memoria RAM:** 8 GB DDR4
- **Sistema Operativo:** Ubuntu 20.04.4 LTS

Metodología

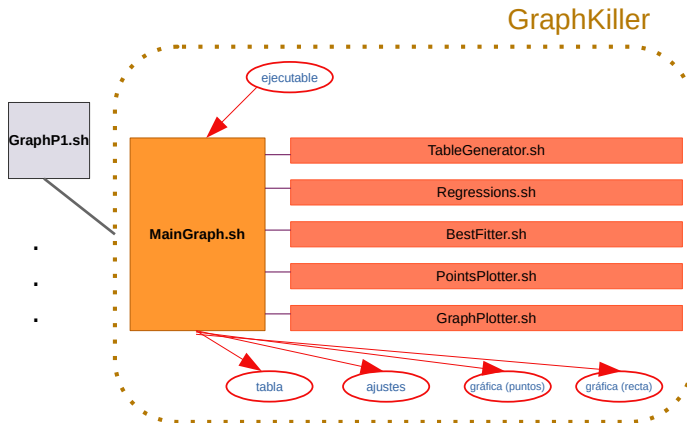


Figura: Esquema del funcionamiento de GraphKiller

Eficiencia teórica

Algoritmo HeapSort

$$T(n) \in \mathcal{O}(n \cdot \log(n))$$

Algoritmo de Inserción

$$T(n) \in \mathcal{O}(n^2)$$

Algoritmo de Floyd

$$T(n) \in \mathcal{O}(n^3)$$

Algoritmo QuickSort

$$T(n) \in \mathcal{O}(n \cdot \log(n))$$

Algoritmo de Selección

$$T(n) \in \mathcal{O}(n^2)$$

Algoritmo de Hanoi

$$T(n) \in \mathcal{O}(2^n)$$

Más **detalles** en la memoria de la práctica.

Eficiencia empírica: Caso $T(n) \in \mathcal{O}(n \cdot \log(n))$

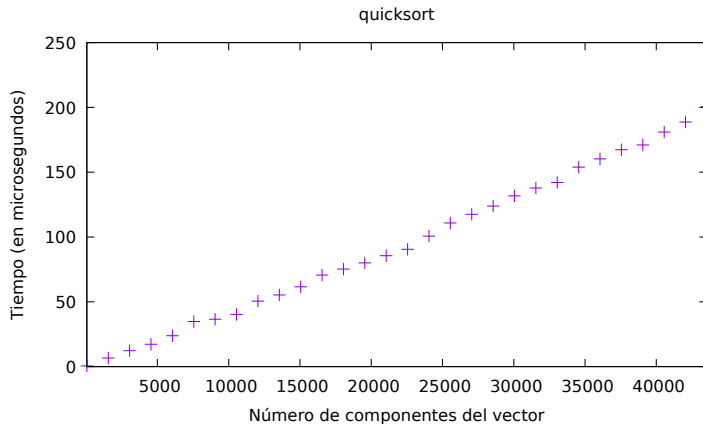


Figura: Gráfica de tiempos de ejecución (en μs) puntos del algoritmo QuickSort (HP)

Eficiencia empírica: Caso $T(n) \in \mathcal{O}(n^2)$

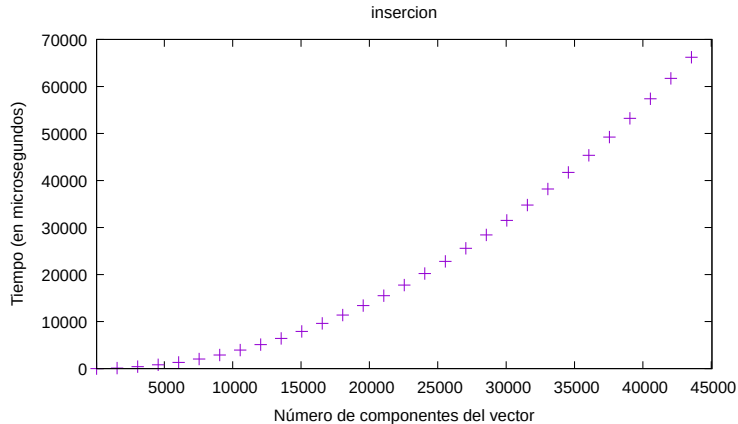


Figura: Gráfica de tiempos de ejecución (en μs) puntos del algoritmo de Inserción (Lenovo)

Eficiencia empírica: Caso $T(n) \in \mathcal{O}(n^3)$

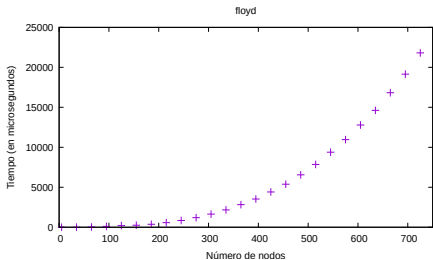


Figura: Gráfica de tiempos de ejecución (en μs) puntos del algoritmo de Floyd (ASUS)

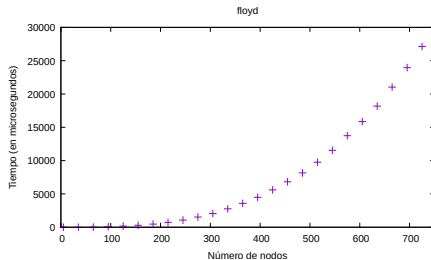
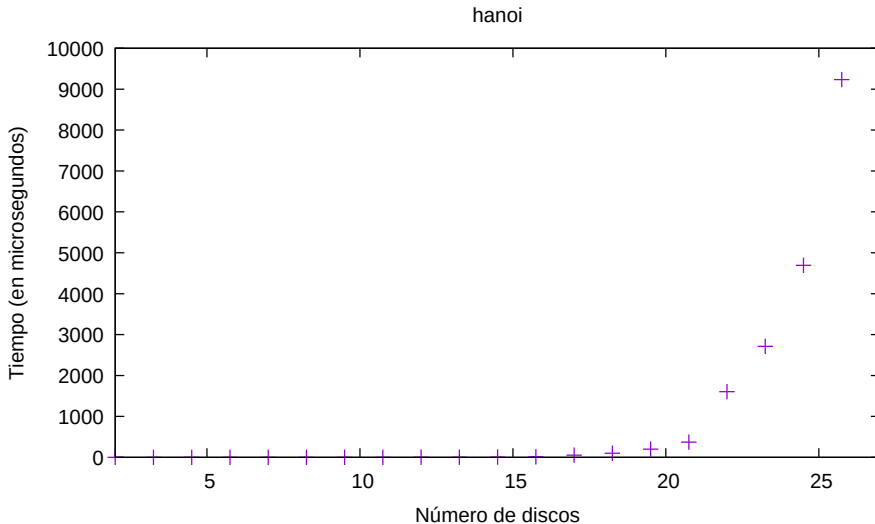
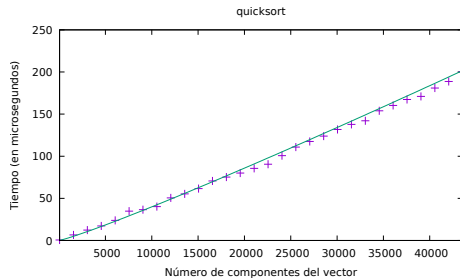


Figura: Gráfica de tiempos de ejecución (en μs) puntos del algoritmo de Floyd (Lenovo)

Eficiencia empírica: Caso $T(n) \in \mathcal{O}(2^n)$



Eficiencia híbrida: Algoritmo QuickSort



Función de ajuste

$$f(x) = 3,66 \cdot 10^{-4}x \log(x) + 50$$

Coefficiente de determinación

$$R^2 = 0,9957$$

Figura: Gráfica de tiempos de ejecución (en μs) puntos del algoritmo de Inserción (HP) con función de ajuste

Eficiencia híbrida: Algoritmo Inserción

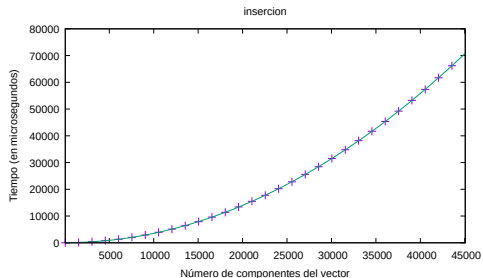


Figura: Gráfica de tiempos de ejecución (en μs) puntos del algoritmo de Inserción (Lenovo) con función de ajuste

Función de ajuste

$$f(x) = 3,48955 \cdot 10^{-5}x^2 - 0,000620098x + 48,0855$$

Coefficiente de determinación

$$R^2 = 0,99999$$

Eficiencia híbrida: Algoritmo Floyd

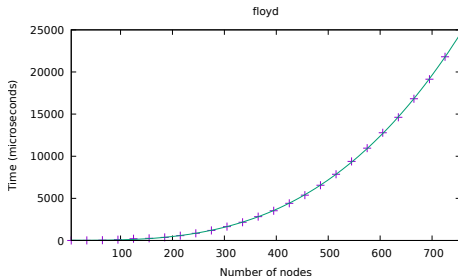


Figura: Gráfica de tiempos de ejecución (en μs) puntos del algoritmo de Floyd (Asus) con función de ajuste

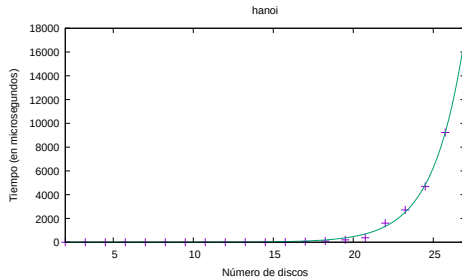
Función de ajuste

$$f(x) = 5,57676 \cdot 10^{-5} x^3 + 0,00138722 x^2 - 0,315202 x + 34,0467$$

Coefficiente de determinación

$$R^2 = 1,0000$$

Eficiencia híbrida: Algoritmo Hanoi



Función de ajuste

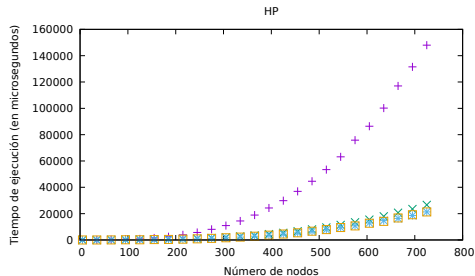
$$f(x) = 0,0158 \cdot e^{0,503814x}$$

Coefficiente de determinación

$$R^2 = 0,9921$$

Figura: Gráfica de tiempos de ejecución (en μs) puntos del algoritmo de Hanoi (Lenovo) con función de ajuste

Niveles de Optimización



- Morado: O0
- Verde: O1
- Amarillo: O2
- Azul: O3

Figura: Gráfica de tiempos de ejecución (HP) en diferentes niveles de optimización

Comparación de algoritmos de ordenación

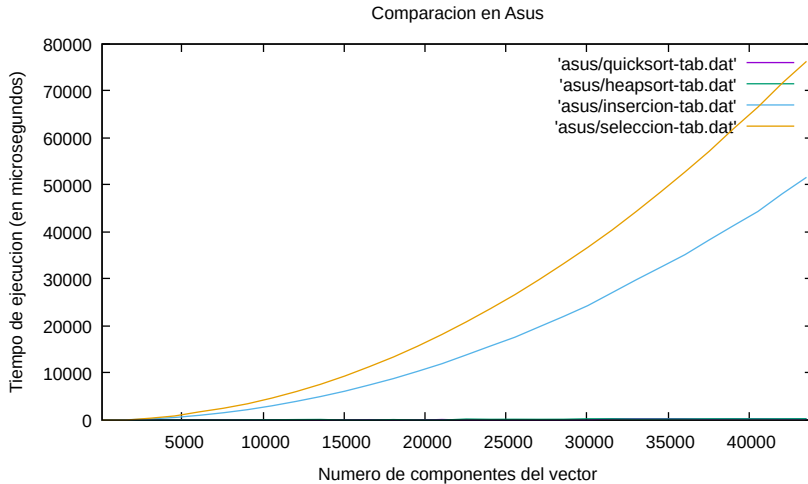


Figura: Gráfica de tiempos de ejecución (ASUS) para diferentes algoritmos de ordenación

Mejor y peor caso para Inserción

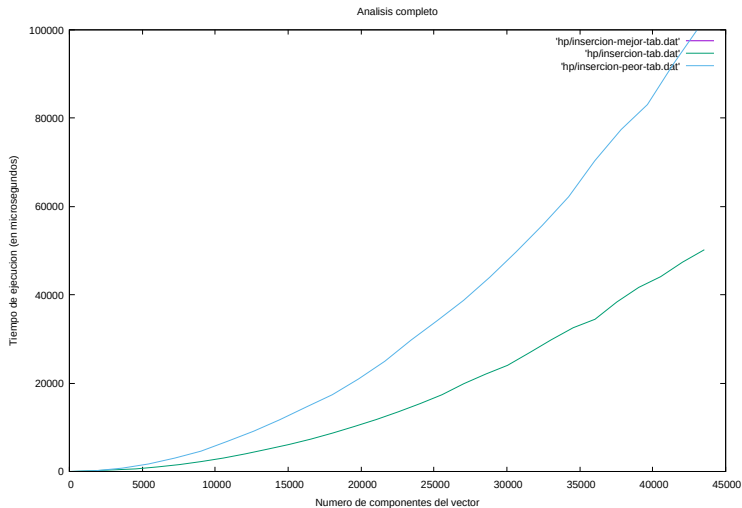


Figura: Gráfica de tiempos de ejecución (HP) para Inserción

Mejor y Peor Caso para Selección

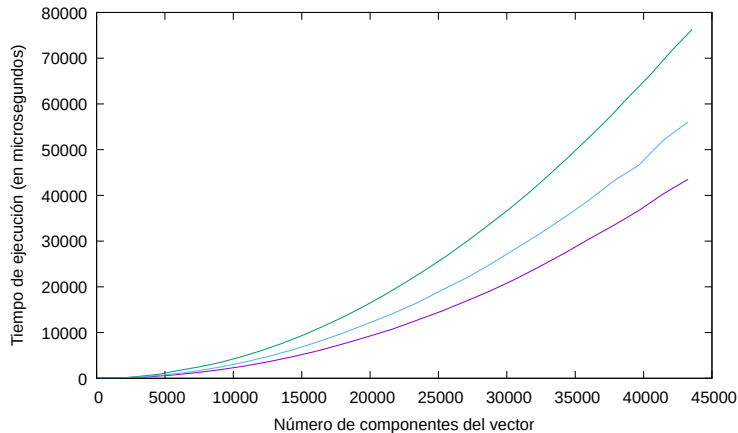





Figura: Gráfica de tiempos de ejecución (ASUS) para Selección

Conclusiones

Los aspectos tratados más relevantes son:

- **Comprobar que los análisis empíricos verifican los resultados teóricos esperados.** Calculado los coeficientes de determinación comprobamos que la función de regresión que mejor se ajusta en cada caso coincide con el tipo de función dado por el análisis teórico.
- **Comparar los resultados experimentales en equipos distintos.** Hemos ilustrado que los tiempos de ejecución de un mismo algoritmo se reducen con un mejor hardware.
- **Cómo elegir un algoritmo de ordenación.** Para grandes tamaños de problema conviene escoger los de mejor orden de eficiencia mientras que para tamaños inferiores es indiferente.
- **Análisis del peor y mejor caso en Selección e Inserción.** El tiempo de ejecución se reduce considerablemente en vectores ordenados ascendentemente, y aumenta en vectores ordenados descendentemente, sobre todo en Inserción.
- **Comparar los tiempos de ejecución para optimizaciones distintas.** Para tamaños pequeños las diferencias no son apreciables, en cambio para valores mayores estas se acentúan.

-  Verdegay Galdeano. (2017). Lecciones de Algorítmica / José Luis Verdegay. Técnica Avicam.
-  Cormen. (2017). Introduction to algorithms / Thomas H. Cormen... [et al.] (3rd ed.). PHI Learning.
-  Garrido Carrillo. (2018). Estructuras de datos avanzadas: con soluciones en C++ / A. Garrido. Universidad de Granada.