

Algorítmica

Curso 2023-2024

Grupo Viterbi



PRÁCTICA 1-ANÁLISIS DE EFICIENCIA DE ALGORITMOS

Integrantes:

Miguel Ángel De la Vega Rodríguez	miguevrod@correo.ugr.es
Alberto De la Vera Sánchez	joaquin724@correo.ugr.es
Joaquín Avilés De la Fuente	adelaveras01@correo.ugr.es
Manuel Gomez Rubio	e.manuelgmez@go.ugr.es
Pablo Linari Perez	e.pablolinari@go.ugr.es

Facultad de Ciencias UGR
Escuela Técnica Ingeniería Informática UGR
Granada
2023-2024

Índice

1	Participación	2
2	Equipo de trabajo	2
3	Objetivos	2
4	Diseño del estudio	3
4.1	Algoritmos de ordenación de vectores	3
5	Algoritmos	3
5.1	Burbuja	4
5.2	Selección	4
5.3	Inserción	5
5.4	Hanoi	5
5.5	Fibonacci	6
5.6	Floyd	6
6	Conclusiones	6

Participación

- **Miguel Ángel De la Vega Rodríguez:** 20%
 - Plantilla y estructura del documento \LaTeX
 - Cómputo de la eficiencia de los algoritmos (Resultados y Ajuste)
- **Joaquín Avilés De la Fuente:** 20%
 - Descripción del Objetivo de la práctica
 - Diseño del estudio
- **Alberto De la Vera Sánchez:** 20%
- **Manuel Gomez Rubio** 20%
- **Pablo Linari Perez:** 20%

Equipo de trabajo

- **Miguel Ángel De la Vega Rodríguez:** (Ordenador donde se ha realizado el computo)
 - AMD Ryzen 7 2700X 8-Core
 - 16 GB RAM DDR4 3200 MHz
 - NVIDIA GeForce GTX 1660 Ti
 - 1 TB SSD NVMe

Objetivos

En esta práctica, se han implementado los siguientes algoritmos de ordenación: **quicksort**, **mergesort**, **heapsort**, **inserción**, **burbuja**, y **selección**. Además, se han implementado los algoritmos de **Floyd**, que calcula el costo del camino mínimo entre cada par de nodos de un grafo dirigido, de **Fibonacci**, que calcula los números de la sucesión de Fibonacci, y de **Hanoi**, que resuelve el famoso problema de las torres de Hanoi. Se ha aplicado la siguiente metodología:

- En primer lugar, aunque tenemos la eficiencia teórica de estos algoritmos, se realizarán los cálculos necesarios para demostrar como se obtiene dicha eficiencia utilizando los distintos métodos estudiados en teoría.
- En segundo lugar, se pasará al estudio empírico de los algoritmos de ordenación de vectores para distintos tipos de datos, es decir, para datos tipo **int**, **float**, **double** y **string**. Posteriormente, se creará las gráficas para cada algoritmo en las que visualizaremos el tiempo de ejecución en función del tamaño del vector y del tipo de dato. Finalmente para esta parte, se hará un cálculo de **eficiencia híbrida** que se basa en ajustar la gráfica obtenida a la función de su eficiencia teórica por mínimos cuadrados, obteniendo por tanto los literales de dicha función que ajustan la gráfica.

- En tercer lugar, se hará el estudio de los otros tres algoritmos de forma similar, es decir, se estudiará la eficiencia de estos de modo empírica, cuyo estudio se mostrará en las gráficas, y se calculará la eficiencia híbrida de estos, a partir de la eficiencia teórica.

Diseño del estudio

Los estudios empíricos han sido realizados en el ordenador con las características mencionadas anteriormente. Además, hemos realizado el estudio empírico de forma aislada para el algoritmo de ordenación de vectores quicksort en los distintos ordenadores de los participantes del grupo para ver como afectan las características hardware de cada ordenador en el tiempo de ejecución, cuyas gráficas se mostrarán en la sección de Algoritmos. En ambos casos se ha hecho uso del sistema operativo Linux, concretamente de Debian, y se ha utilizado el compilador gcc para la compilación de los programas con el flag -Og para la optimización.

4.1. Algoritmos de ordenación de vectores

Para la facilitación del estudio empírico de los algoritmos se ha hecho uso de varios scripts, entre ellos:

- **[AutoCompile.sh]** Este script se encarga de compilar todos los ficheros en una misma carpeta con las mismas opciones de compilación, para garantizar la máxima igualdad posible entre cada algoritmo y organizar la estructura de ficheros.
- **[AutoFinal.sh]** Este script es el encargado de ejecutar todos los algoritmos varias veces con las opciones respectivas para cada uno, el resultado se pasa por un programa AutoMedia.py que se encarga de realizar la media de las ejecuciones de los algoritmos, este resultado es guardado en una carpeta llamada Resultados de la que posteriormente el mismo script genera las graficas de cada algoritmo.
- **[AutoIndividual.sh]** Este script es como el descrito previamente pero unicamente ejecuta un script, esto ha sido útil para hacer pruebas sin la necesidad de esperar la gran cantidad de tiempo que requiere la ejecución de todos los algoritmos.

Algoritmos

Esta sección esta dedicada a mostrar los resultados obtenidos en el estudio de los algoritmos, la estructura seguida para mostrar los resultados consiste en mostrar, para cada algoritmo, los tiempos de ejecución, junto con las gráficas obtenidas y los ajustes correspondientes. Previo a ello, se analizará en cada caso teoricamente la eficiencia prevista para cada algoritmo.

5.1. Burbuja

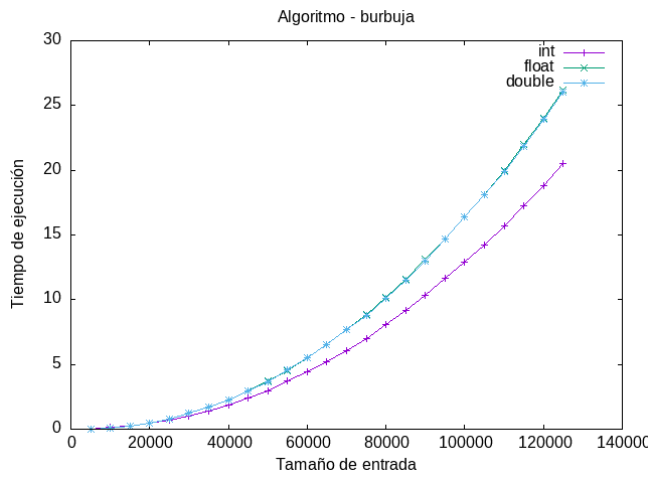


Figura 1: Ejecución algoritmo burbuja

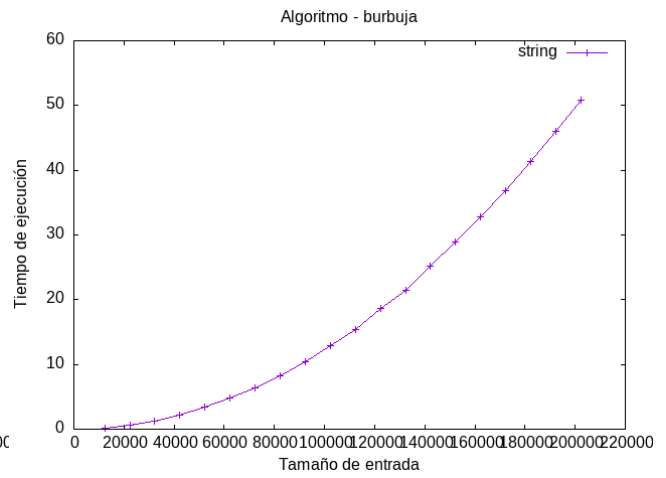


Figura 2: Ejecución algoritmo burbuja con string

5.2. Selecccion

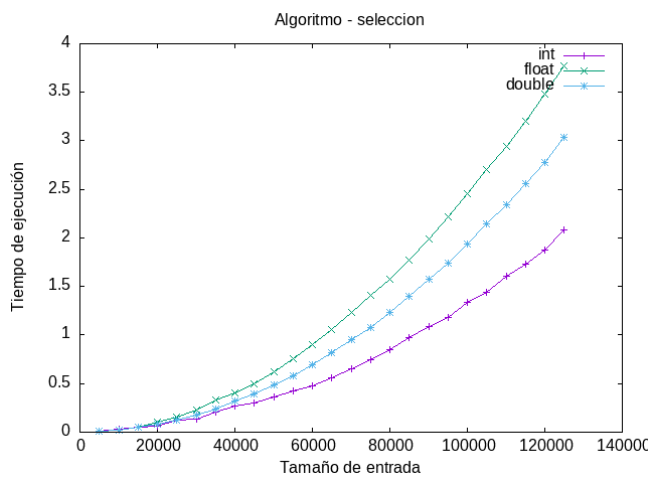


Figura 3: Ejecución algoritmo seleccion

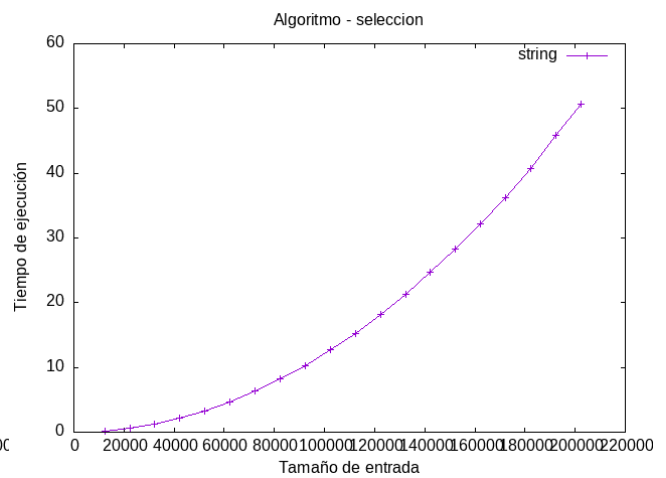


Figura 4: Ejecución algoritmo seleccion con string

5.3. Inserción

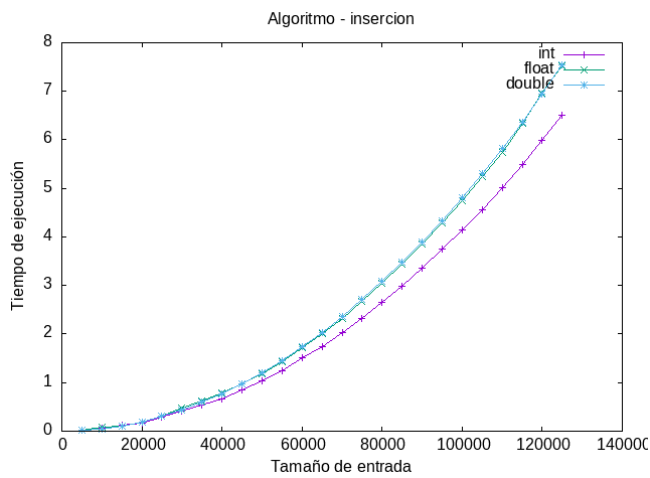


Figura 5: Ejecución algoritmo insercion

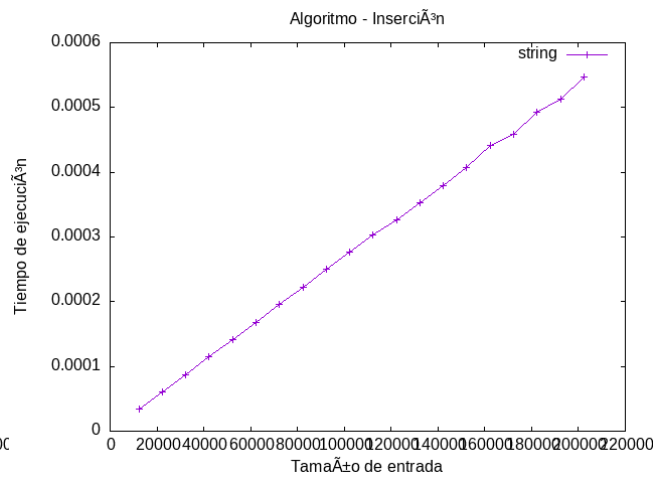


Figura 6: Ejecución algoritmo inserción con string

5.4. Hanoi

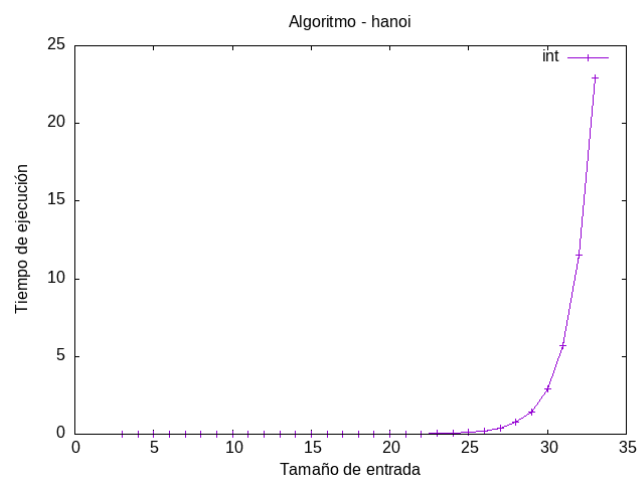


Figura 7: Ejecución algoritmo Hanoi

5.5. Fibonacci

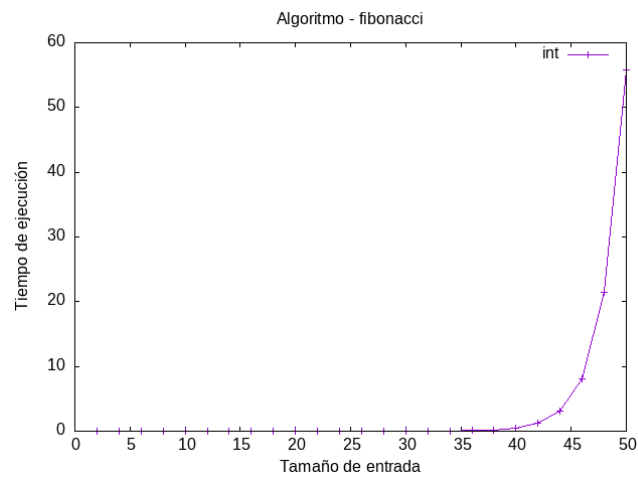


Figura 8: Ejecución algoritmo Fibonacci

5.6. Floyd

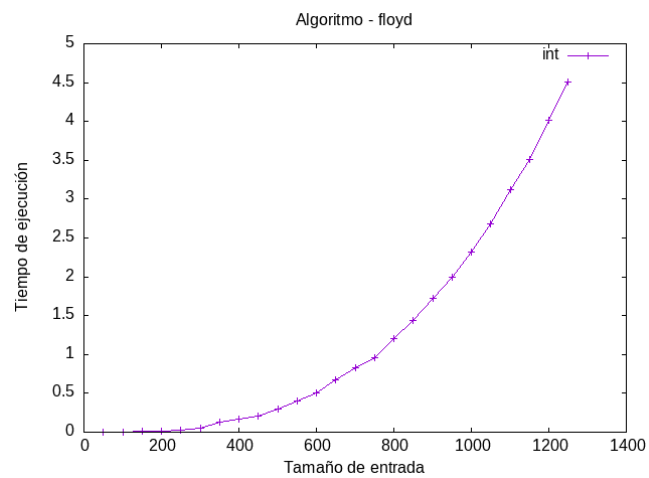


Figura 9: Ejecución algoritmo Floyd

Conclusiones