



Cálculo eficiencia empírica e híbrida

Algoritmos: Burbuja, Inserción, Mergesort,
Quicksort, Floyd, Dijkstra, Fibonacci y Hanoi





PROCEDIMIENTO SEGUIDO
EN TODOS LOS ALGORITMOS



Cálculo eficiencia empírica - clock

Usamos la función clock y capturamos el tiempo antes y después de la llamada a cada algoritmo para posteriormente restar y obtener el tiempo empleado en cada uno.

```
clock_t tini=clock();    // Anotamos el tiempo de inicio

//Llamada algoritmo

clock_t tfin=clock();    // Anotamos el tiempo de finalización

// Mostramos resultados
cout << Tam << "\t" << (tfin-tini)/(double)CLOCKS_PER_SEC << endl;
```

Cálculo eficiencia empírica - scripts

Para facilitar la tarea de ejecutar los algoritmos numerosas veces y poder obtener el tiempo de ejecución de cada tamaño, hacemos uso de un script:

```
#!/bin/csh
@ inicio = //tamaño de inicio
@ fin = //tamaño final
@ incremento = //en cuanto incrementamos en cada iteración

@ i = $inicio
echo "" >> algoritmo.dat
while ( $i <= $fin )
    echo tam = $i
    echo `./algoritmo $i` >> algoritmo.dat
    @ i += $incremento
end
```

Cálculo eficiencia empírica - tablas y gráficas

Con los archivos .dat obtenidos generamos una tabla para cada algoritmo. Cada entrada de la tabla tiene el siguiente formato:

tamañoEjecución	tiempo(seg)
-----------------	-------------

Para los algoritmos con la misma eficiencia generamos una sola tabla con el siguiente formato:

tamañoEjecución	tiempoAlgoritmo1(seg)	tiempoAlgoritmo2(seg)
-----------------	-----------------------	-----------------------

Estas tablas nos servirán para posteriormente representar las gráficas de cada algoritmo con gnuplot.

Cálculo eficiencia híbrida

Para obtener el valor de las constantes ocultas definimos una función $f(x)$ que represente la eficiencia teórica por una constante y realizamos el ajuste por regresión con el comando fit en gnuplot.

```
gnuplot> fit f(x) "tiempos_algoritmo.dat" via a
```

Donde a es la constante oculta definida en $f(x)$ y tiempos_algoritmo.dat el fichero de datos con los resultados de medir el tiempo de ejecución.

Además para ver cómo varía la eficiencia híbrida hemos ejecutado los algoritmos con diferente nivel de optimización, concretamente con opciones de compilación -O2 y -O3.



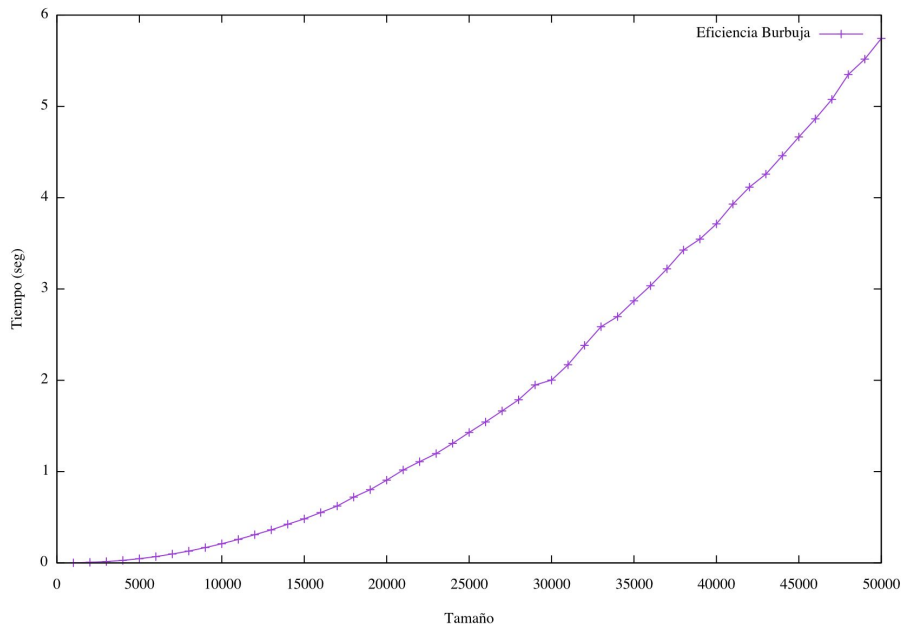
RESULTADOS OBTENIDOS

Algoritmos de inserción y burbuja



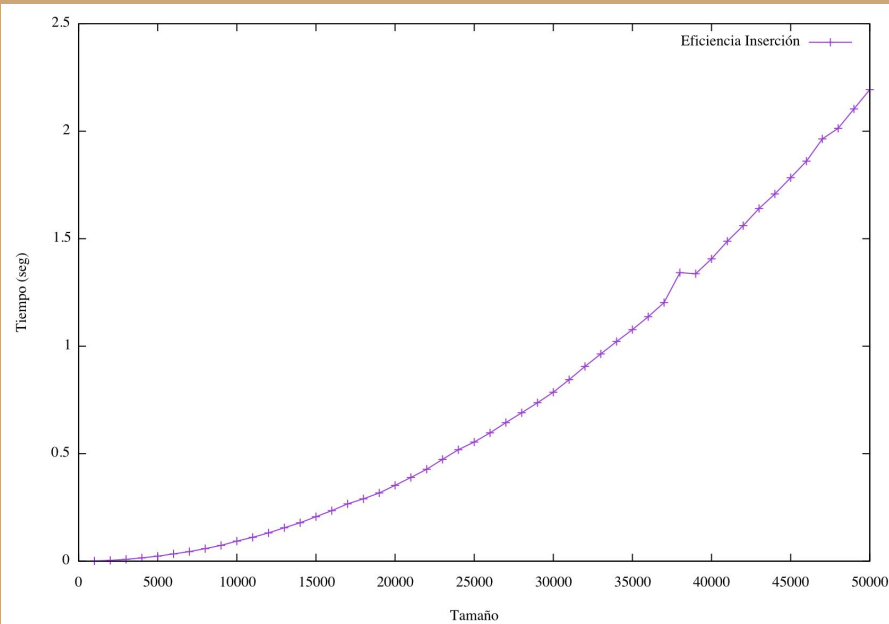
EFICIENCIA

BURBUJA

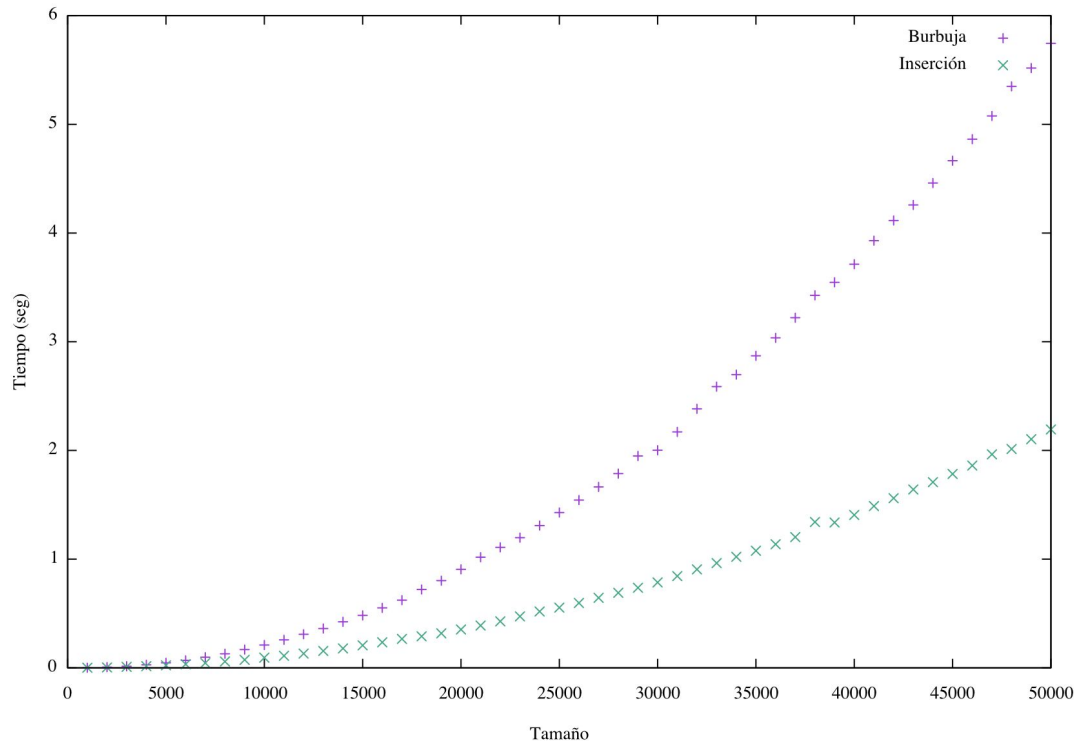


EMPÍRICA

INSERCIÓN

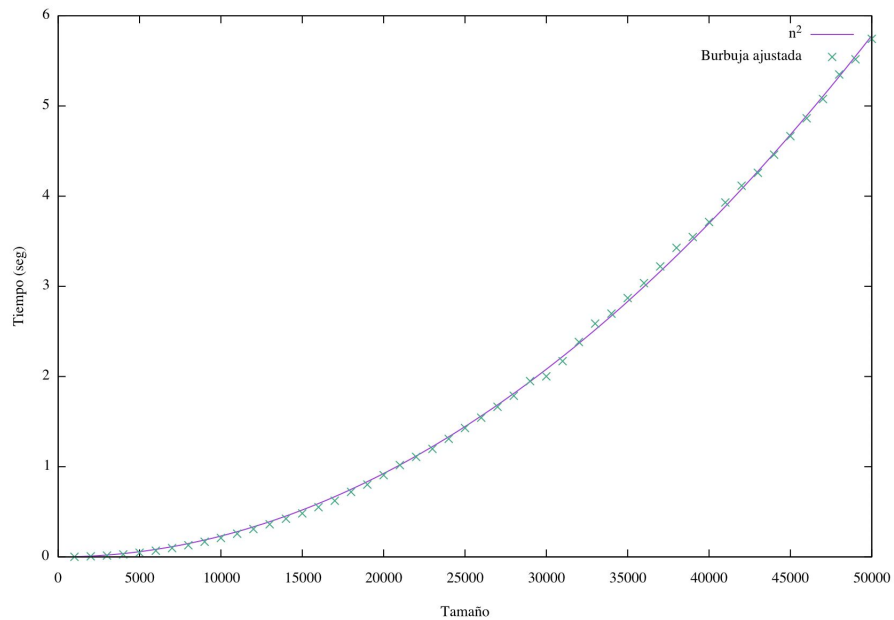


Comparación Burbuja e Inserción empírica



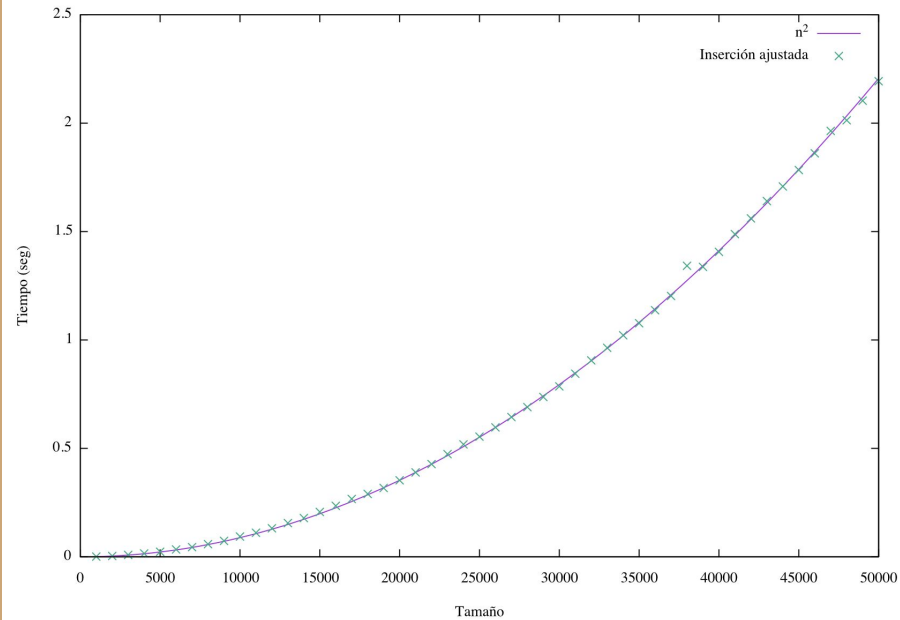
EFICIENCIA

BURBUJA



HÍBRIDA

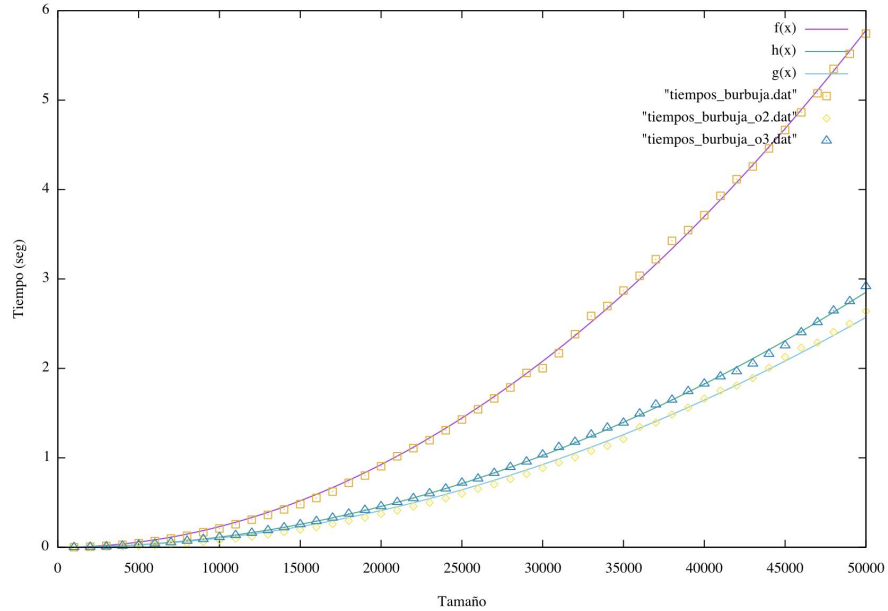
INSERCIÓN



EFICIENCIA

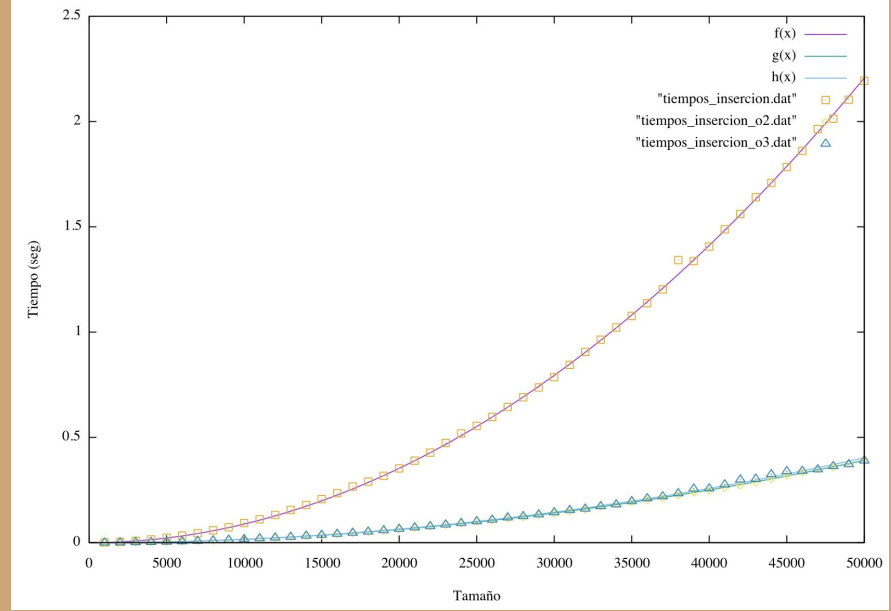
BURBUJA

(con optimización)



HÍBRIDA

INSERCIÓN





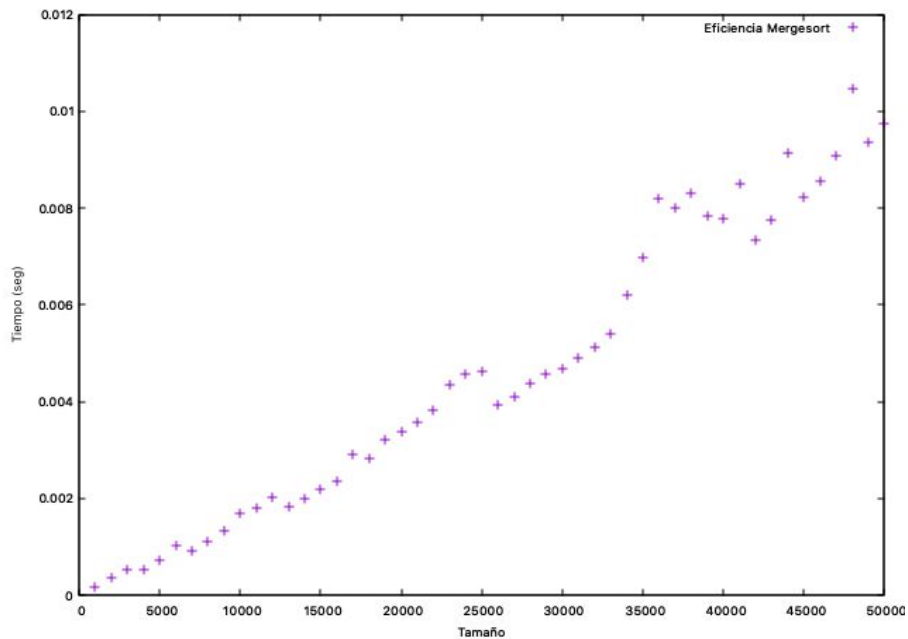
RESULTADOS OBTENIDOS

Algoritmos de mergesort y quicksort



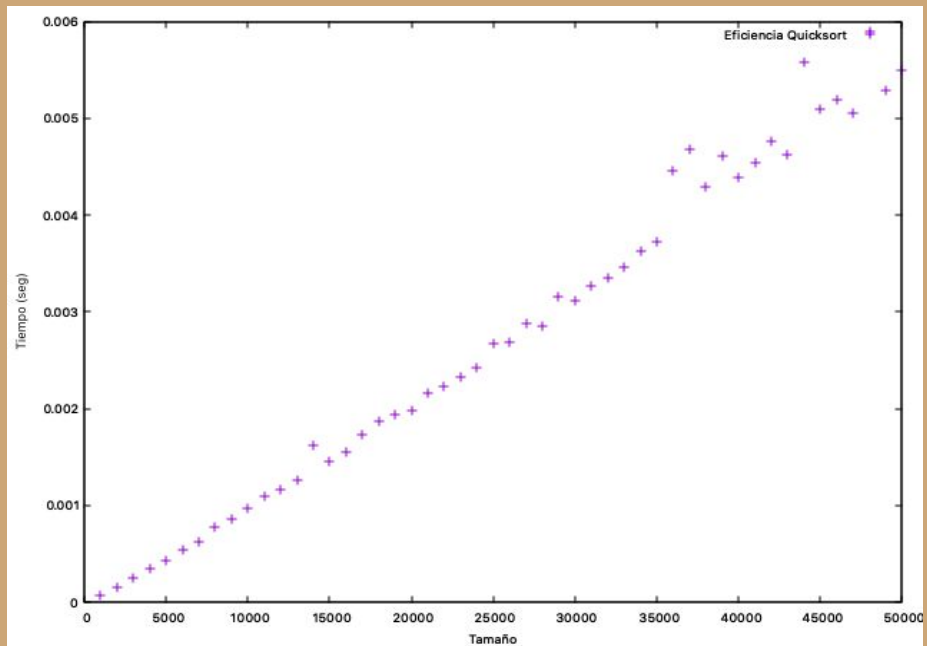
EFICIENCIA

MERGESORT

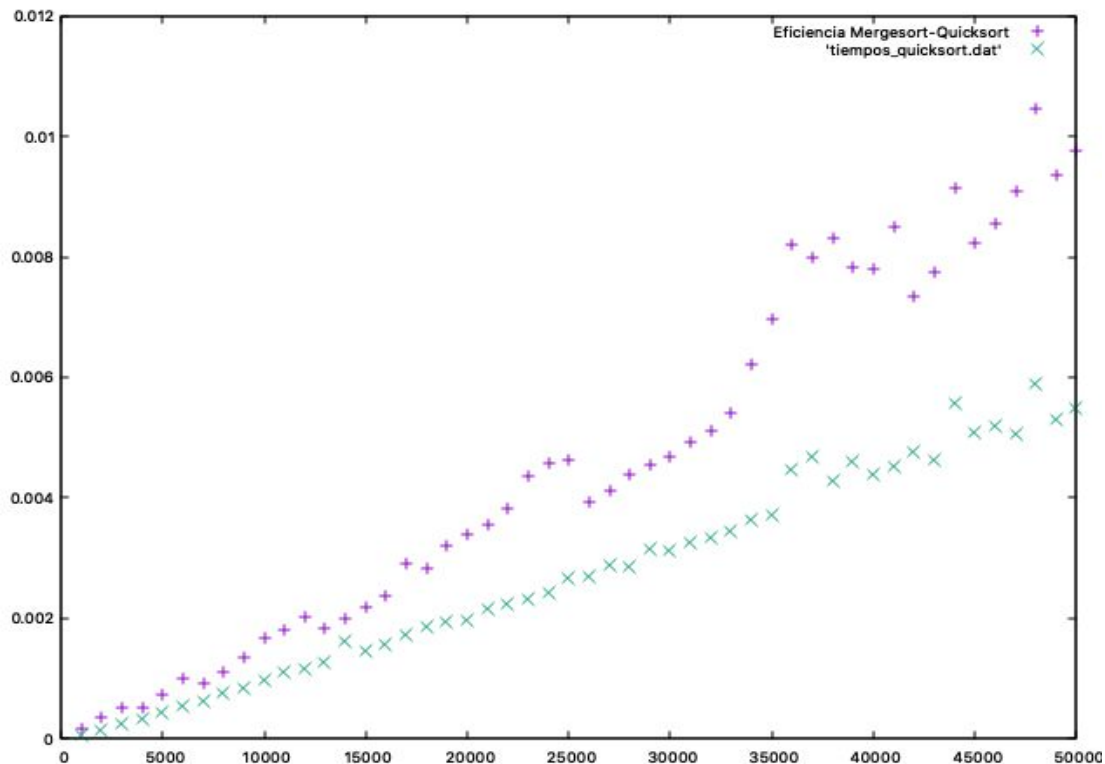


EMPÍRICA

QUICKSORT

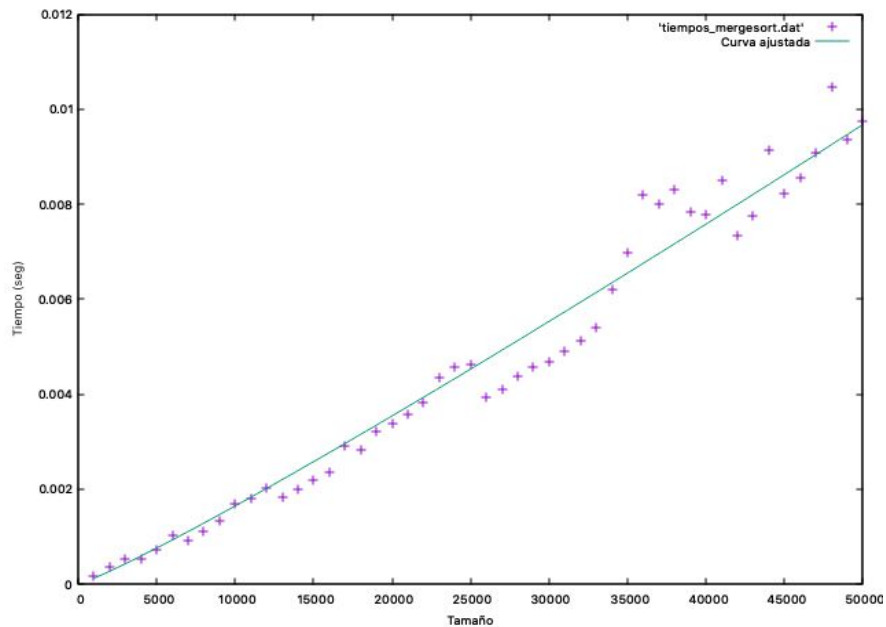


Comparación Mergesort y Quicksort empírica



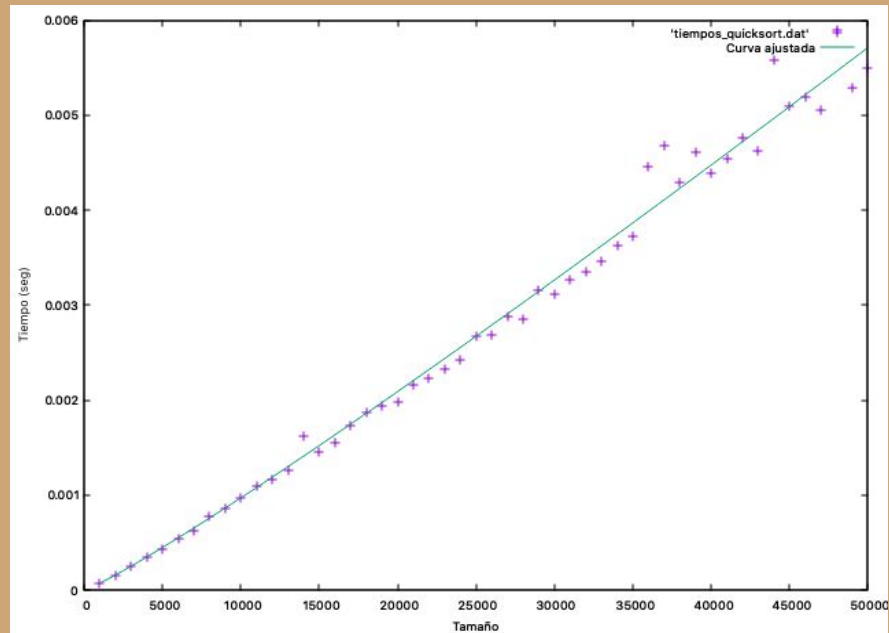
EFICIENCIA

MERGESORT



HÍBRIDA

QUICKSORT

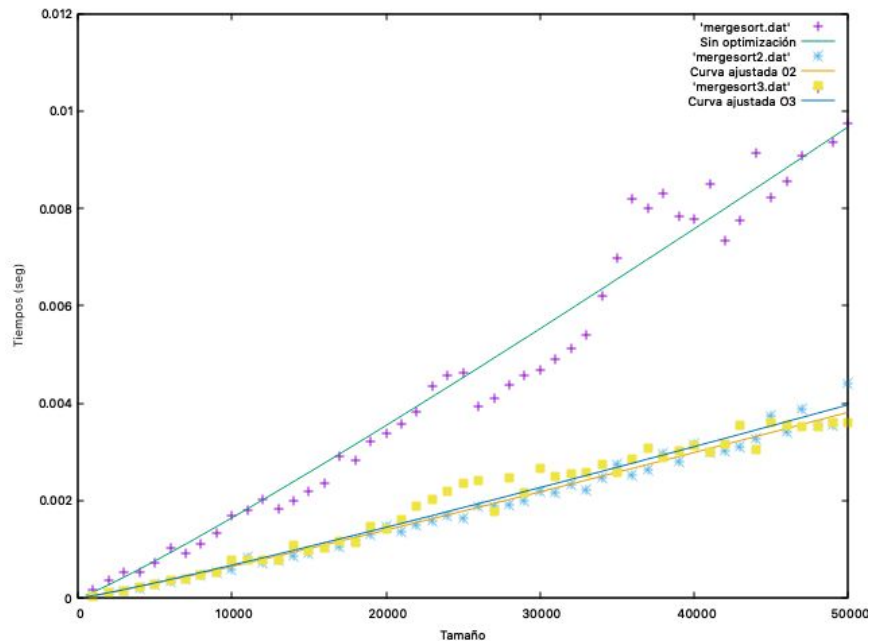


EFICIENCIA

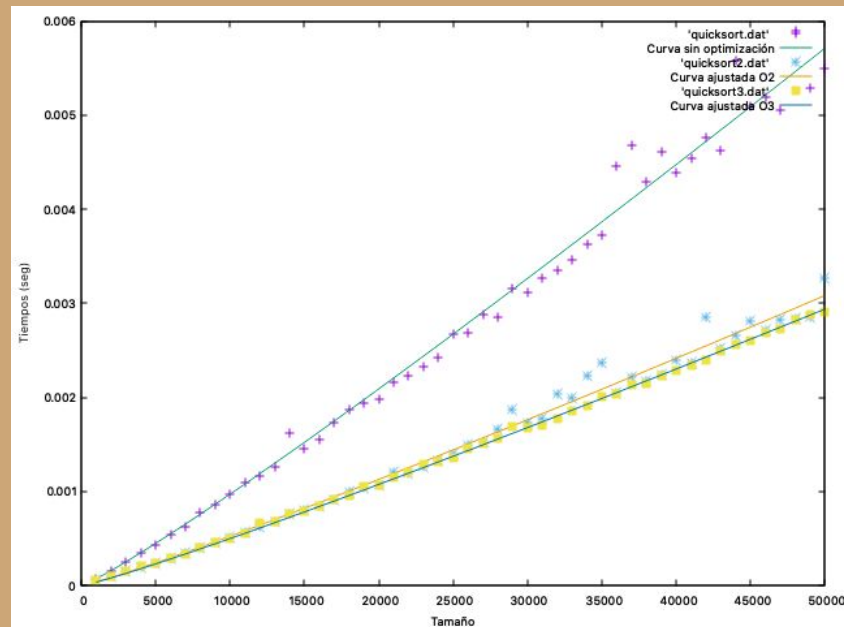
HÍBRIDA

(con optimización)

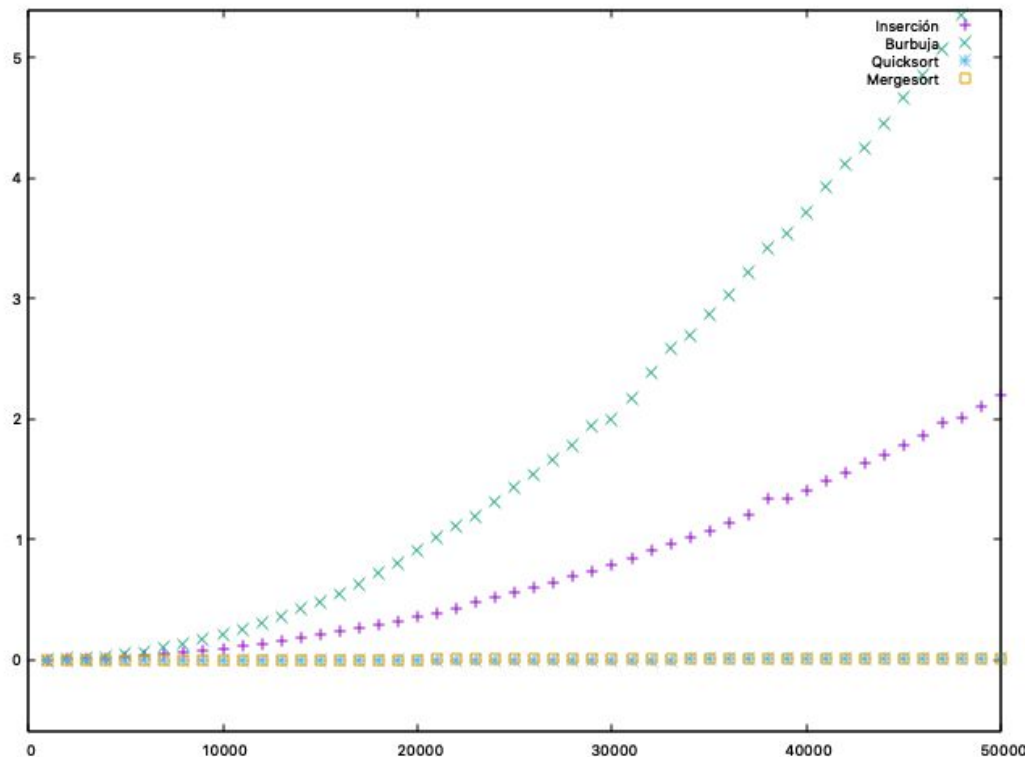
MERGESORT



QUICKSORT



Comparación de los cuatro algoritmos de ordenación





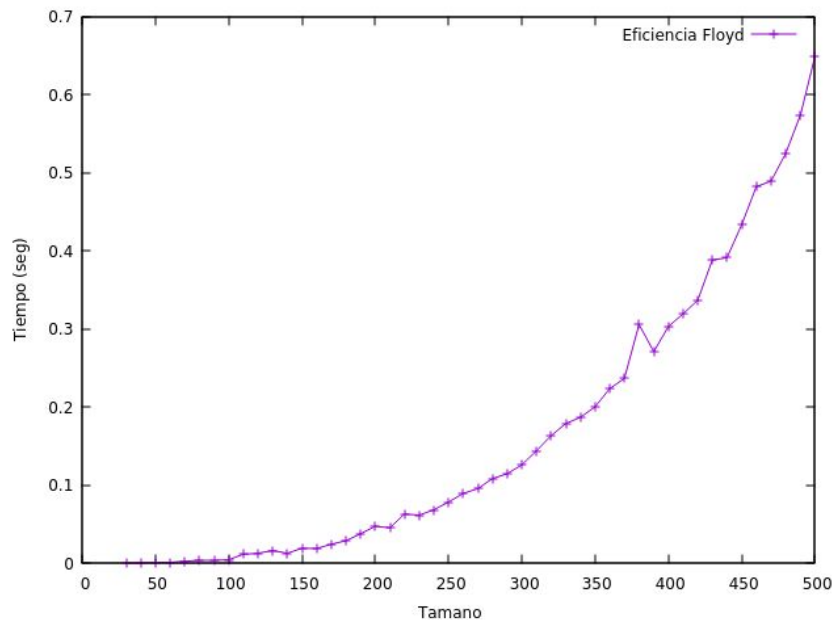
RESULTADOS OBTENIDOS

Algoritmos de Floyd y Dijkstra



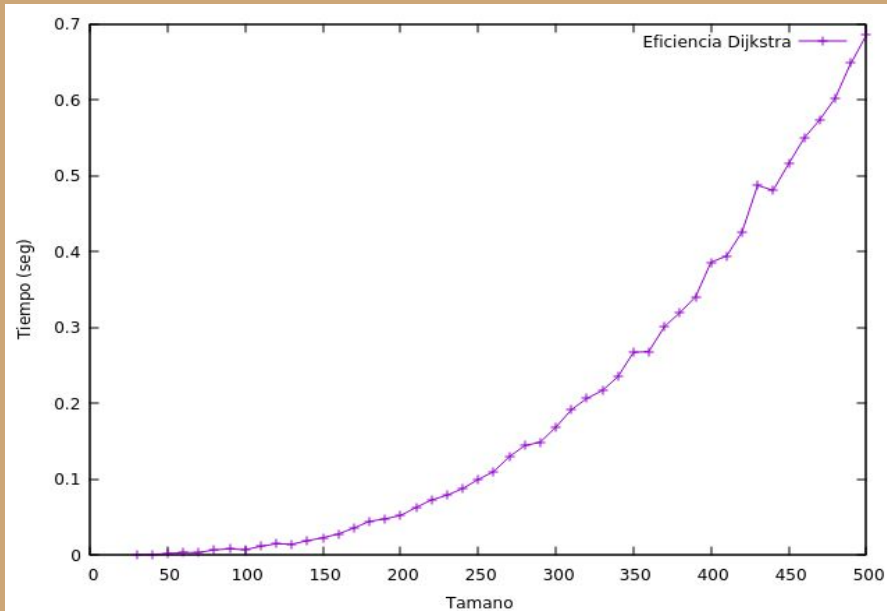
EFICIENCIA

FLOYD

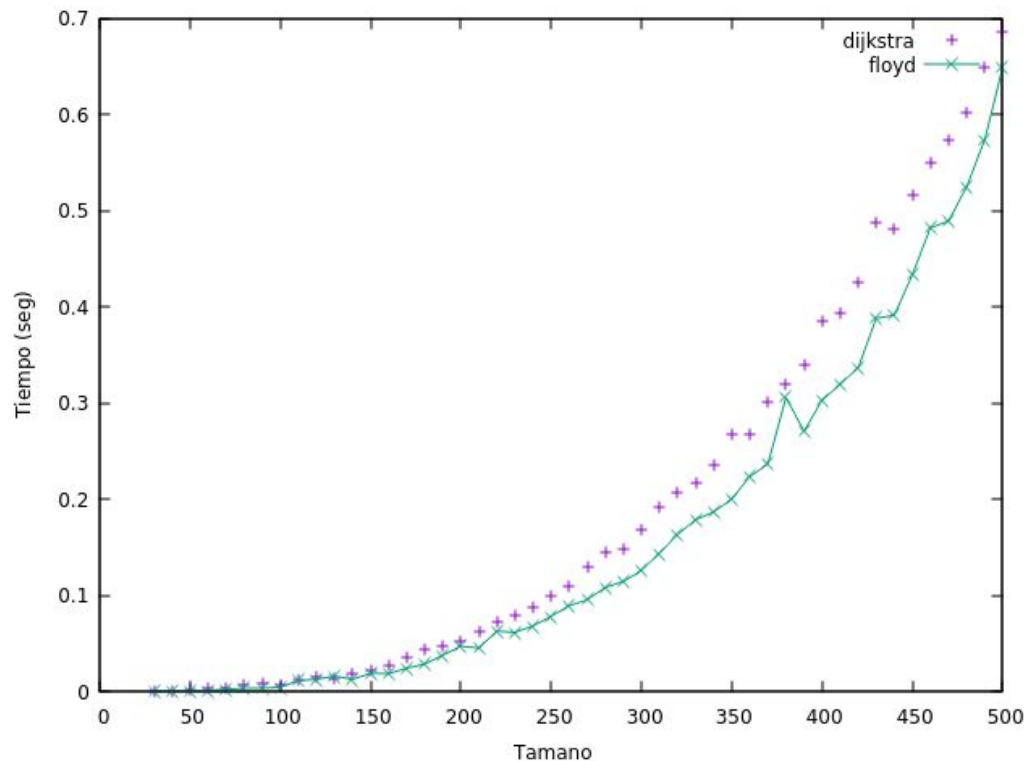


EMPÍRICA

DIJKSTRA

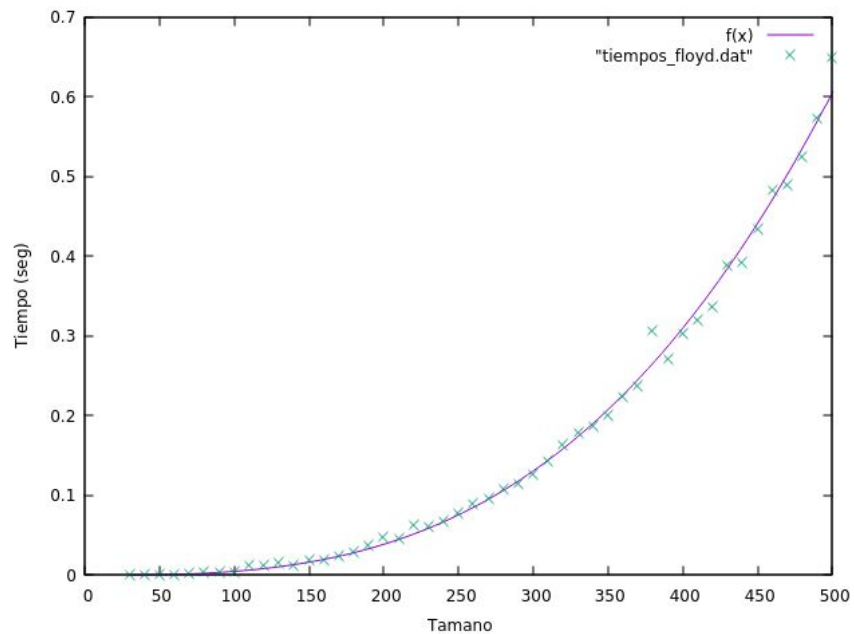


Comparación Floyd y Dijkstra empírica



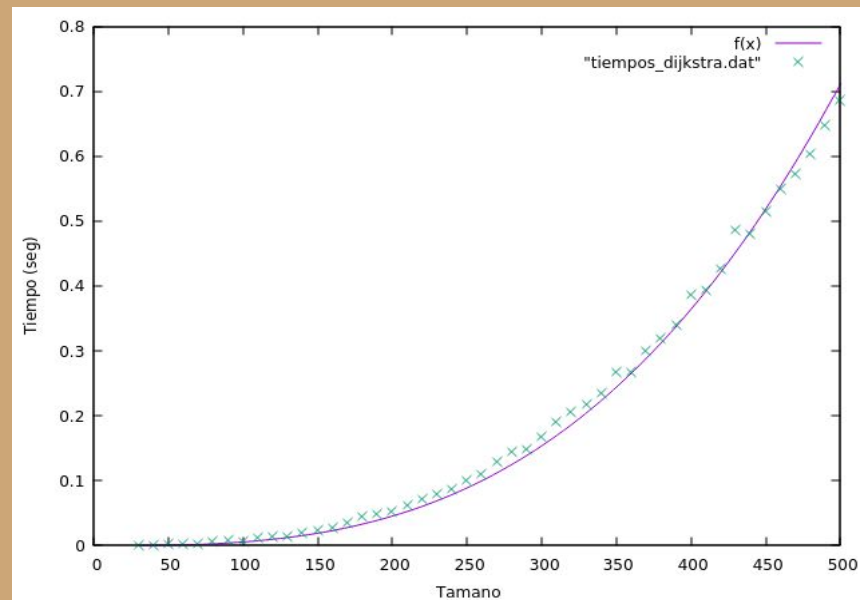
EFICIENCIA

FLOYD



HÍBRIDA

DIJKSTRA

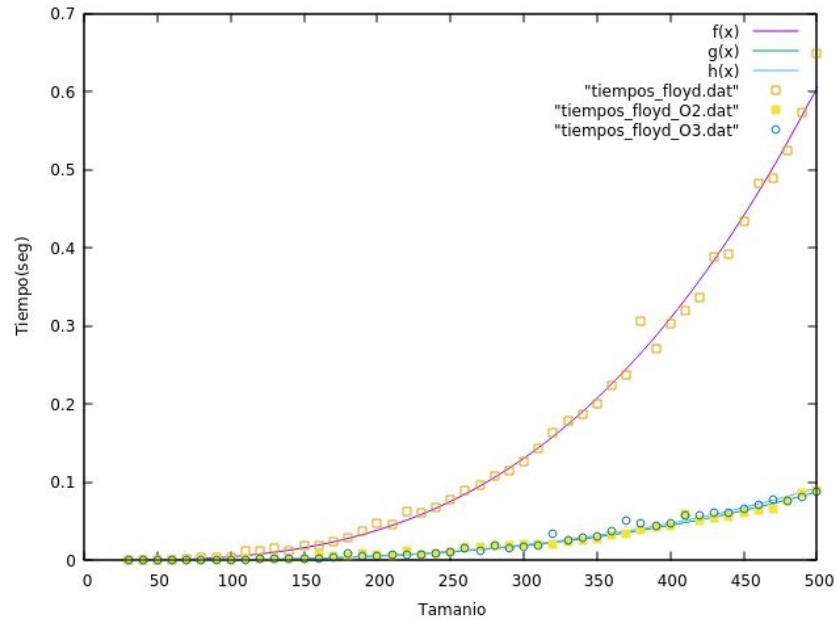


EFICIENCIA

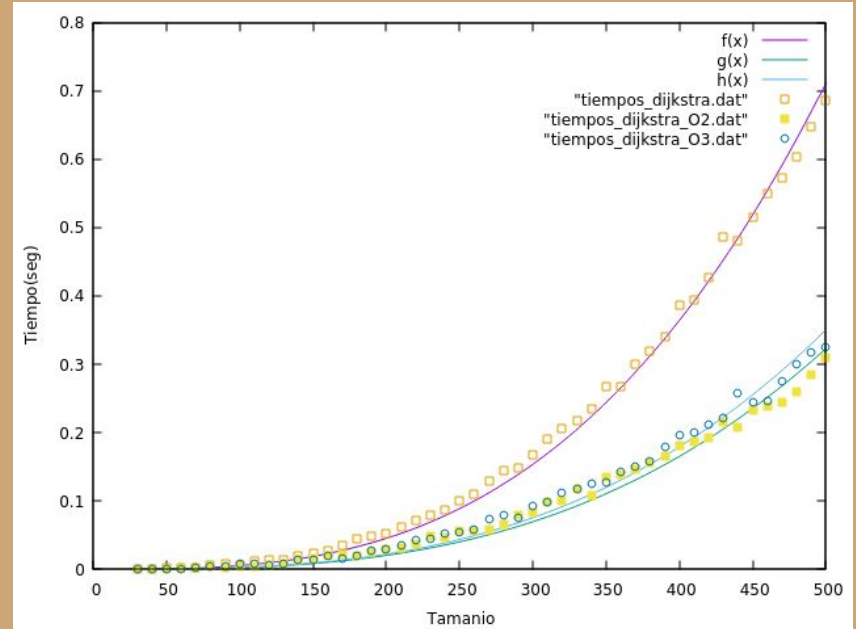
HÍBRIDA

(con optimización)

FLOYD



DIJKSTRA





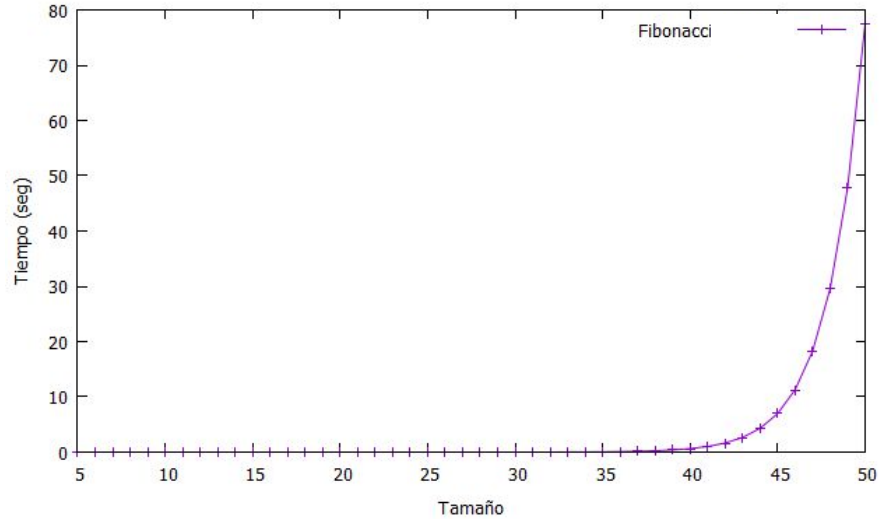
RESULTADOS OBTENIDOS

Algoritmo de Fibonacci

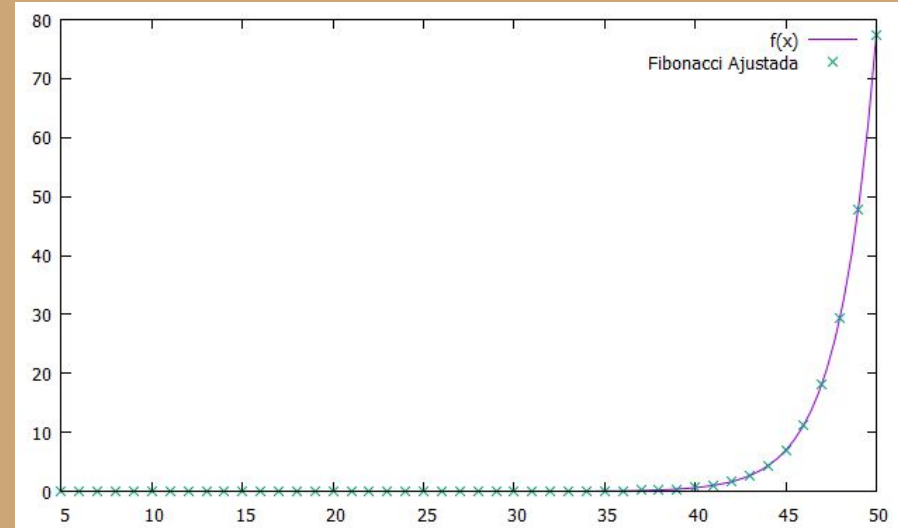


EFICIENCIA FIBONACCI

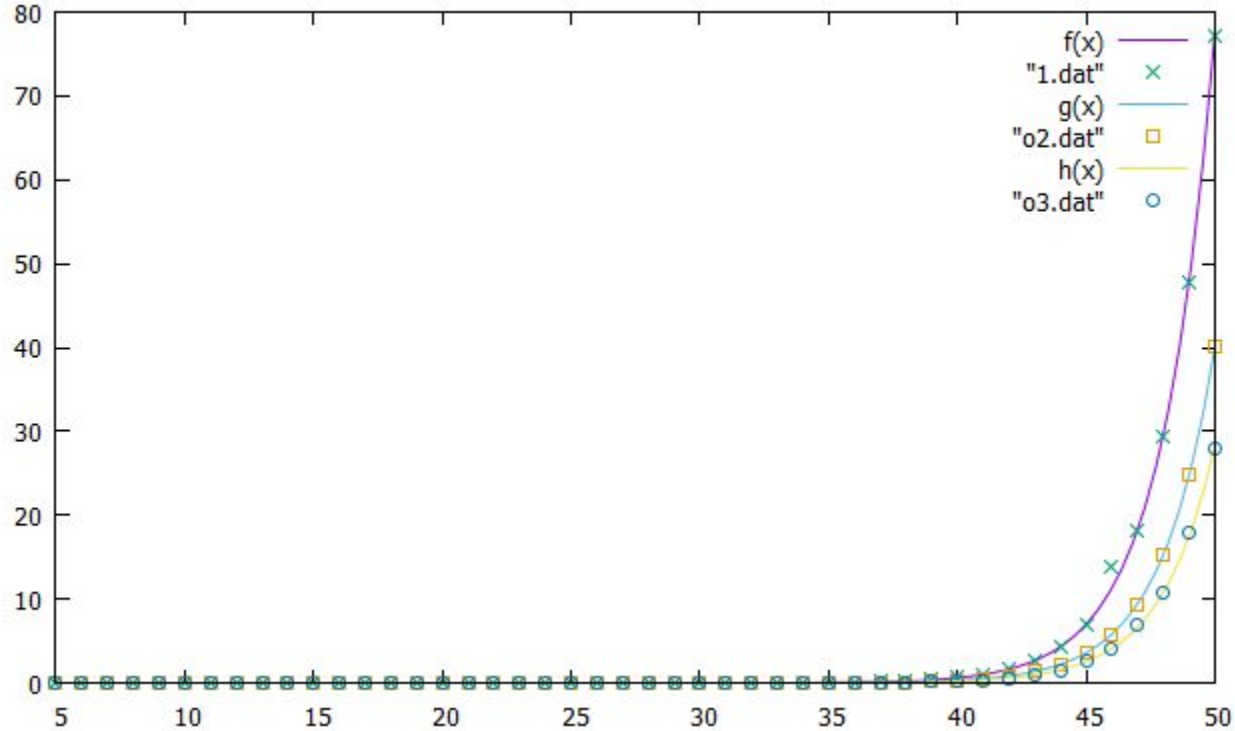
Empírica




Híbrida



Eficiencia híbrida con optimización Fibonacci





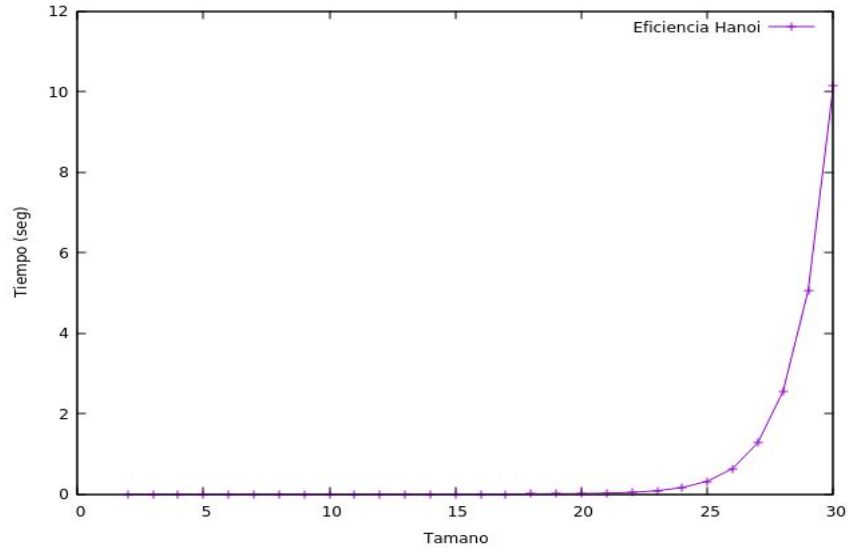
RESULTADOS OBTENIDOS

Algoritmo de Hanoi

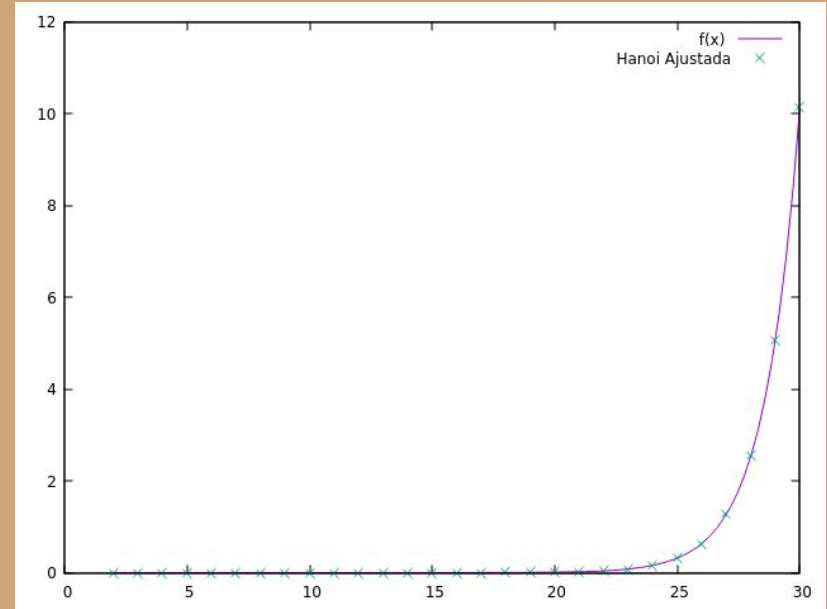


EFICIENCIA HANOI

Empírica



Híbrida



Eficiencia híbrida con optimización Hanoi

