Práctica 1

Análisis Empírico e Híbrido de Eficiencia de Algoritmos

Germán Castilla López
Jorge Gangoso Klöck
Pedro Morales Leyva
Clara Mª Romero Lara

Índice

1 Introducción

- El objetivo de esta práctica es analizar eficiencias de los algoritmos de manera empírica e híbrida
- Para ello, hemos ejecutado algoritmos con diferentes cantidades de datos y los hemos comparado.
- Hemos usado la librería Mtime de Windows destinada a la medición precisa de tiempos.
- Se ha automatizado la creación de scripts de GNUplot.

2 Complejidad o(n^2)

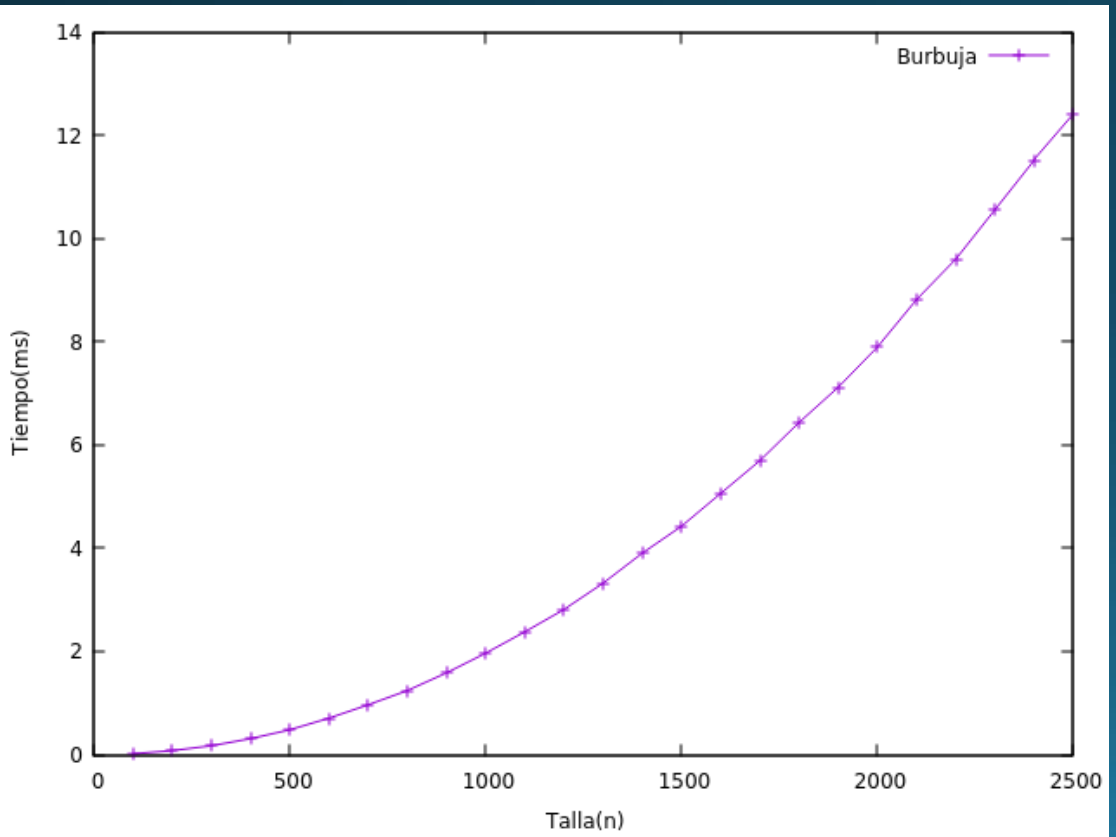
Estos algoritmos son más simples pero no son muy eficientes.

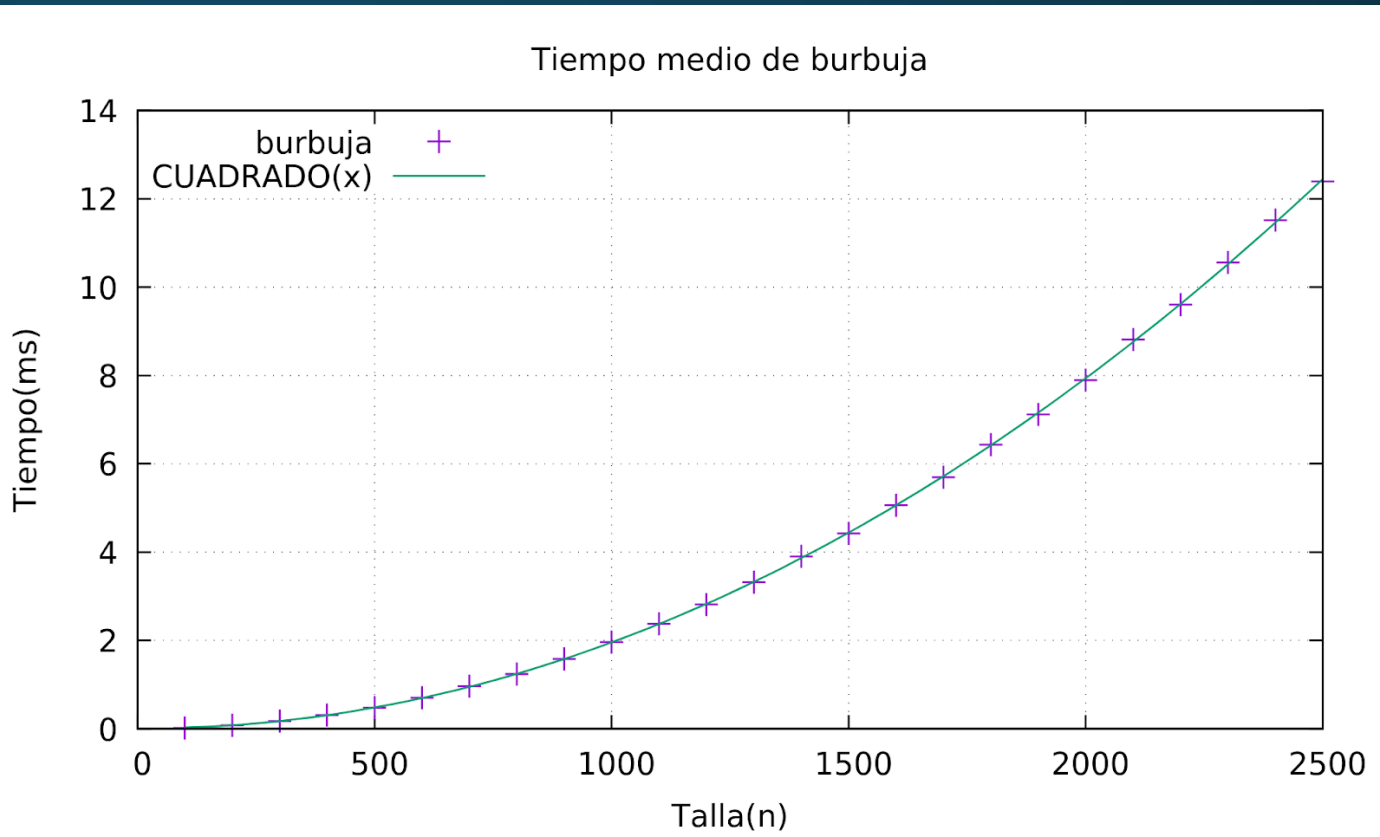
Vamos a ver los siguientes algoritmos:

- Burbuja
- Inserción
- Selección

2.1 Burbuja

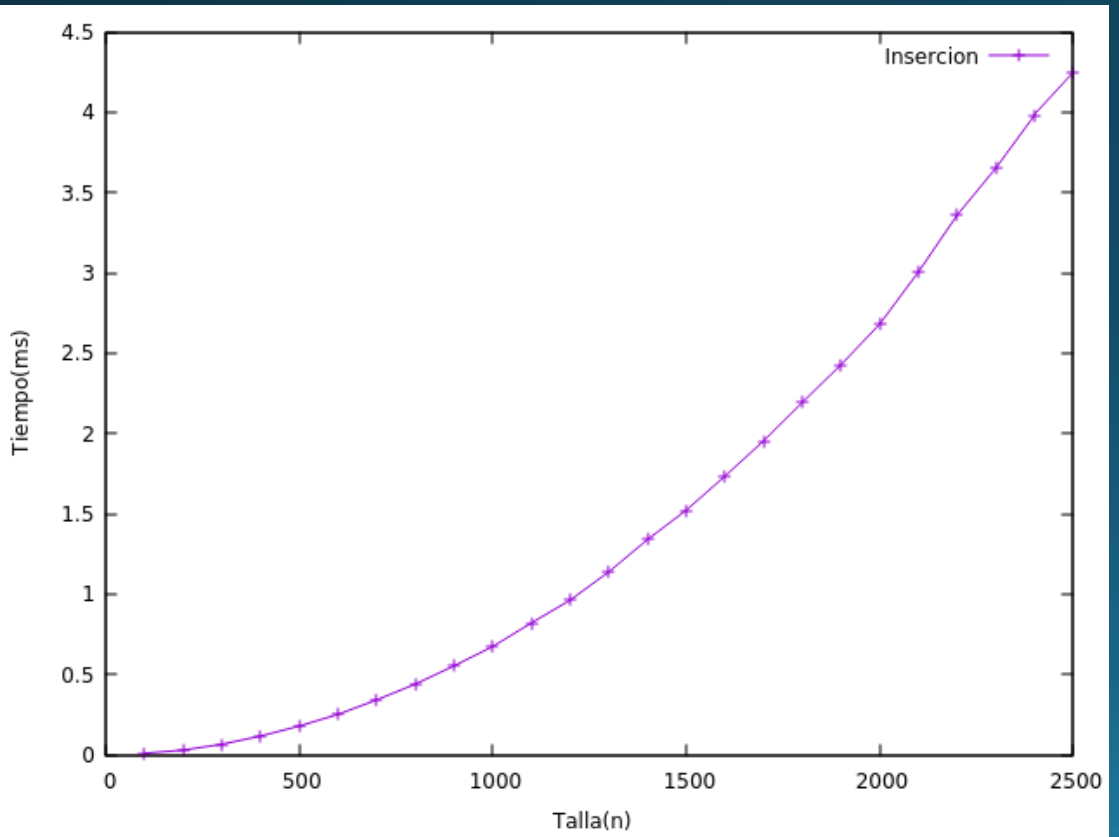
Consiste en comparar cada elemento con el siguiente e intercambiarlos si es necesario y repetirlo hasta que el vector esté ordenado

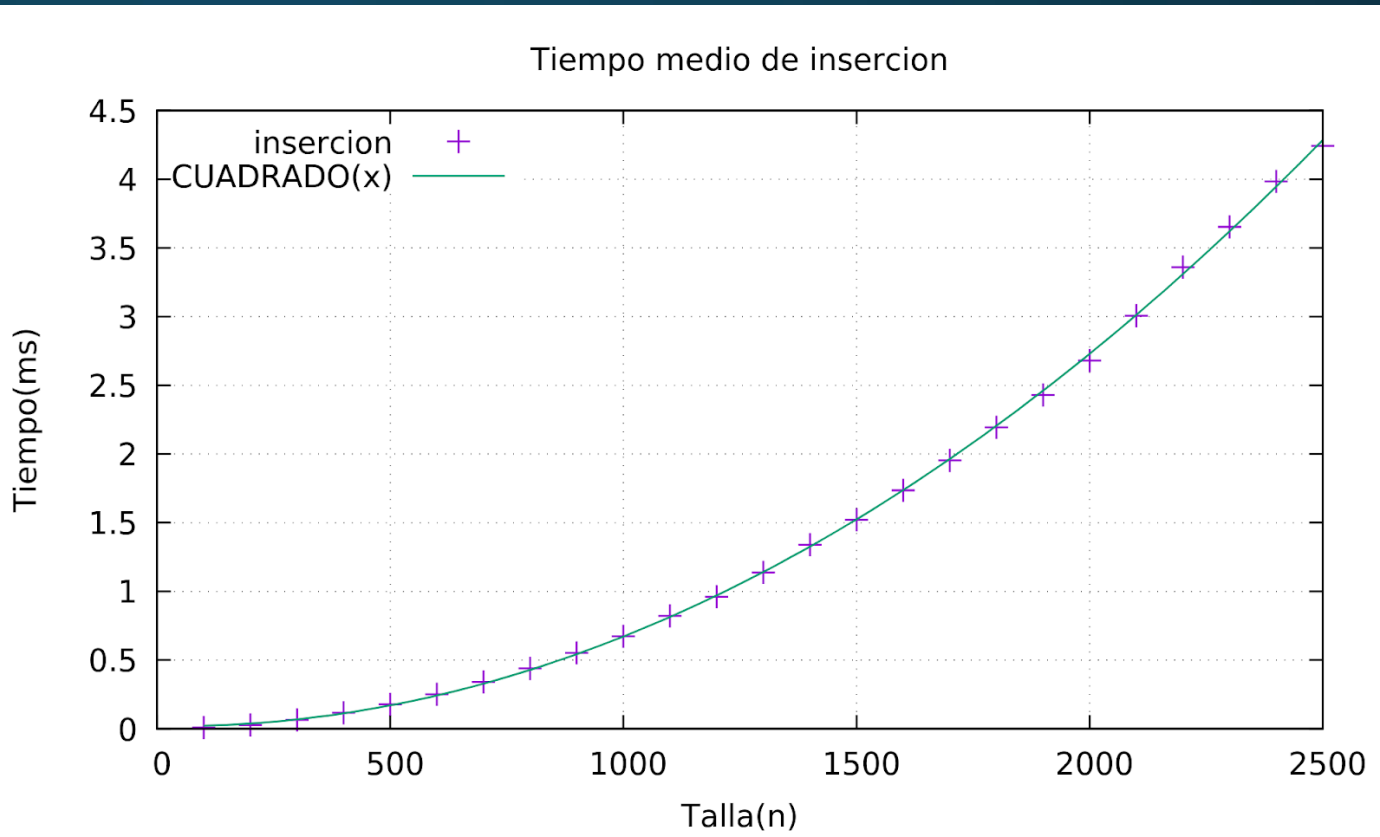




2.2 Inserción

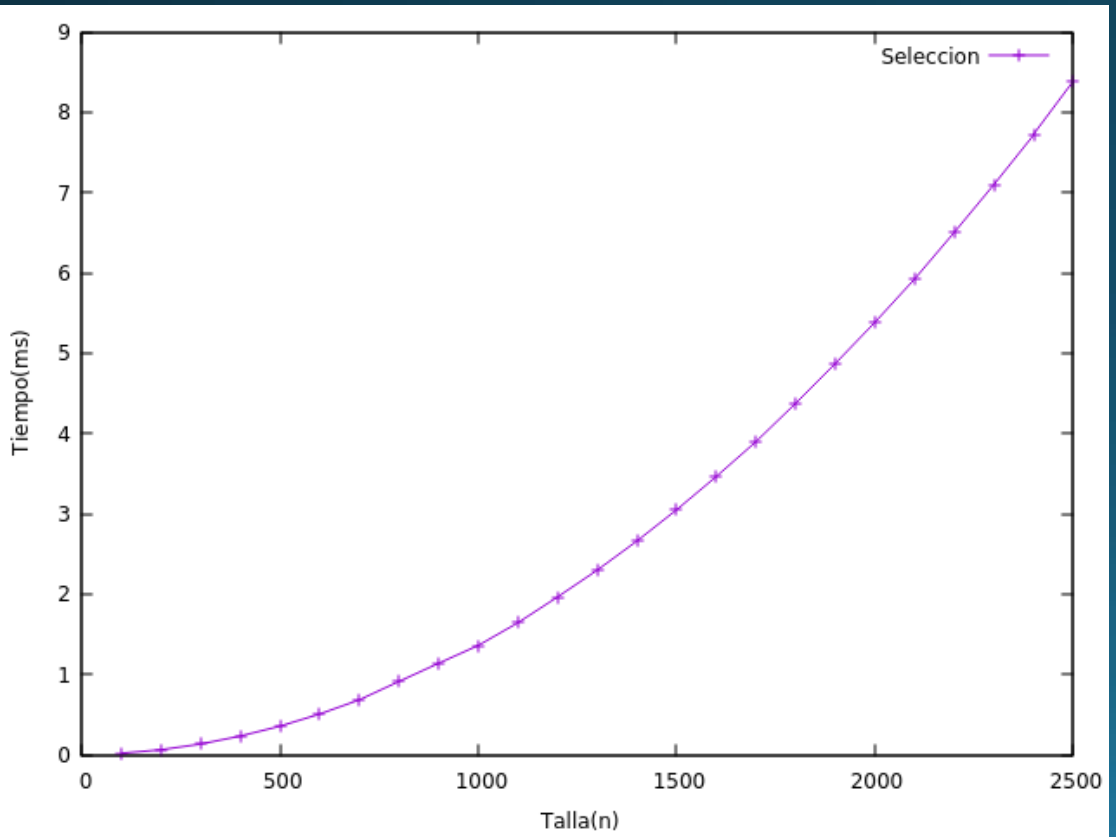
Se seleccionan los elementos uno a uno de principio a fin de vector y se compara con los elementos anteriores, dejándolo en el sitio si es el mayor o poniéndose donde corresponda si no

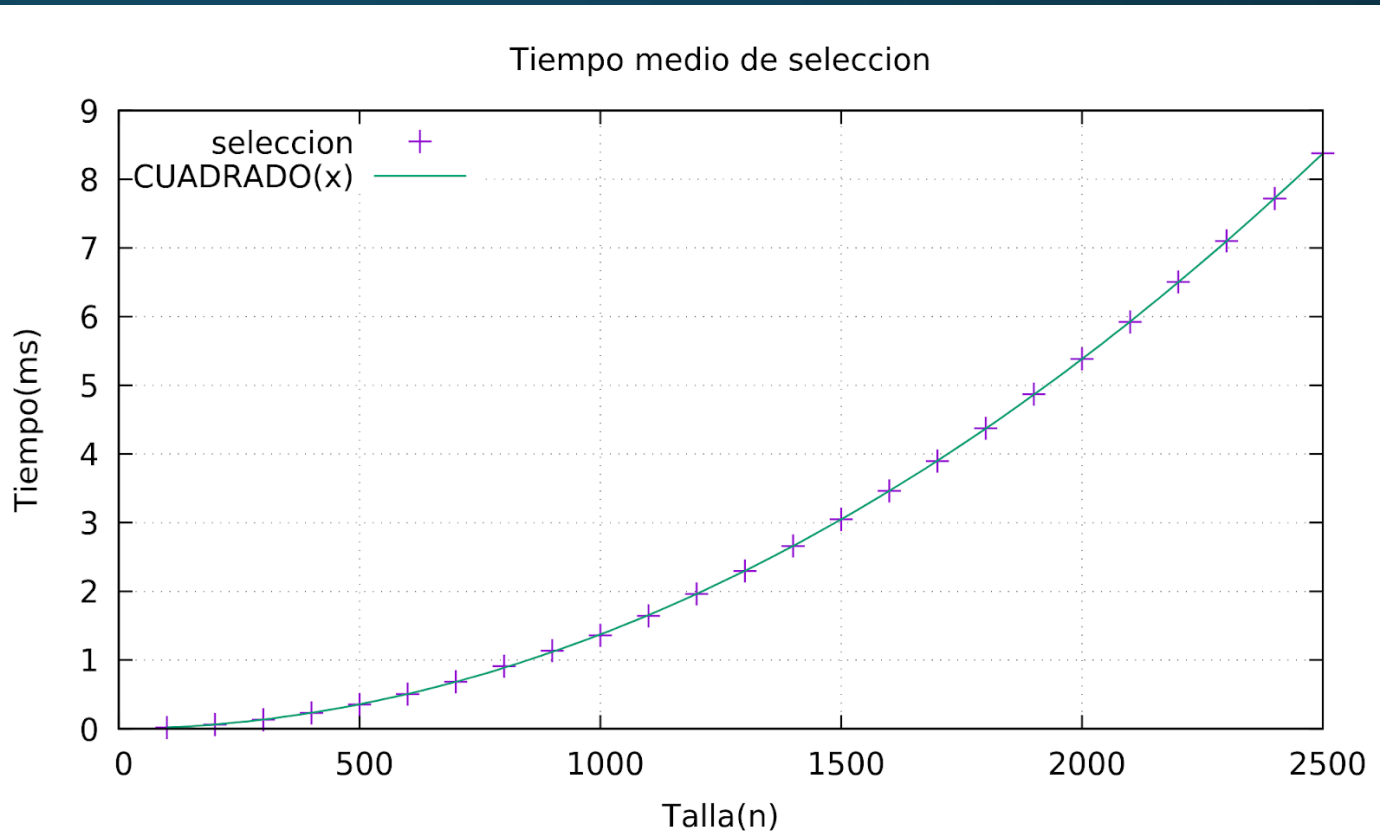




2.3 Selección

Consiste en buscar en cada iteración el menor elemento del vector y fijarlo en la menor posición. Se repite tantas veces como elementos halla en el vector.





3 Complejidad O(nlog(n))

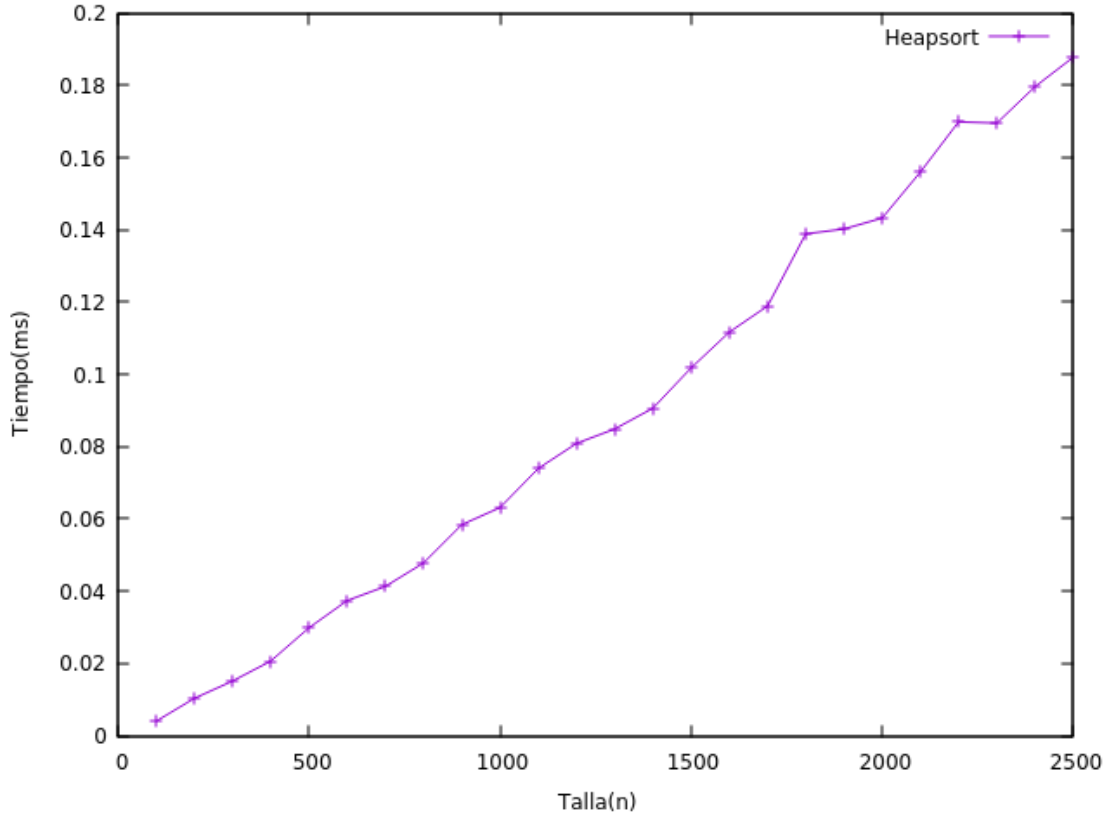
Estos algoritmos tienen una mayor complejidad pero son más eficientes

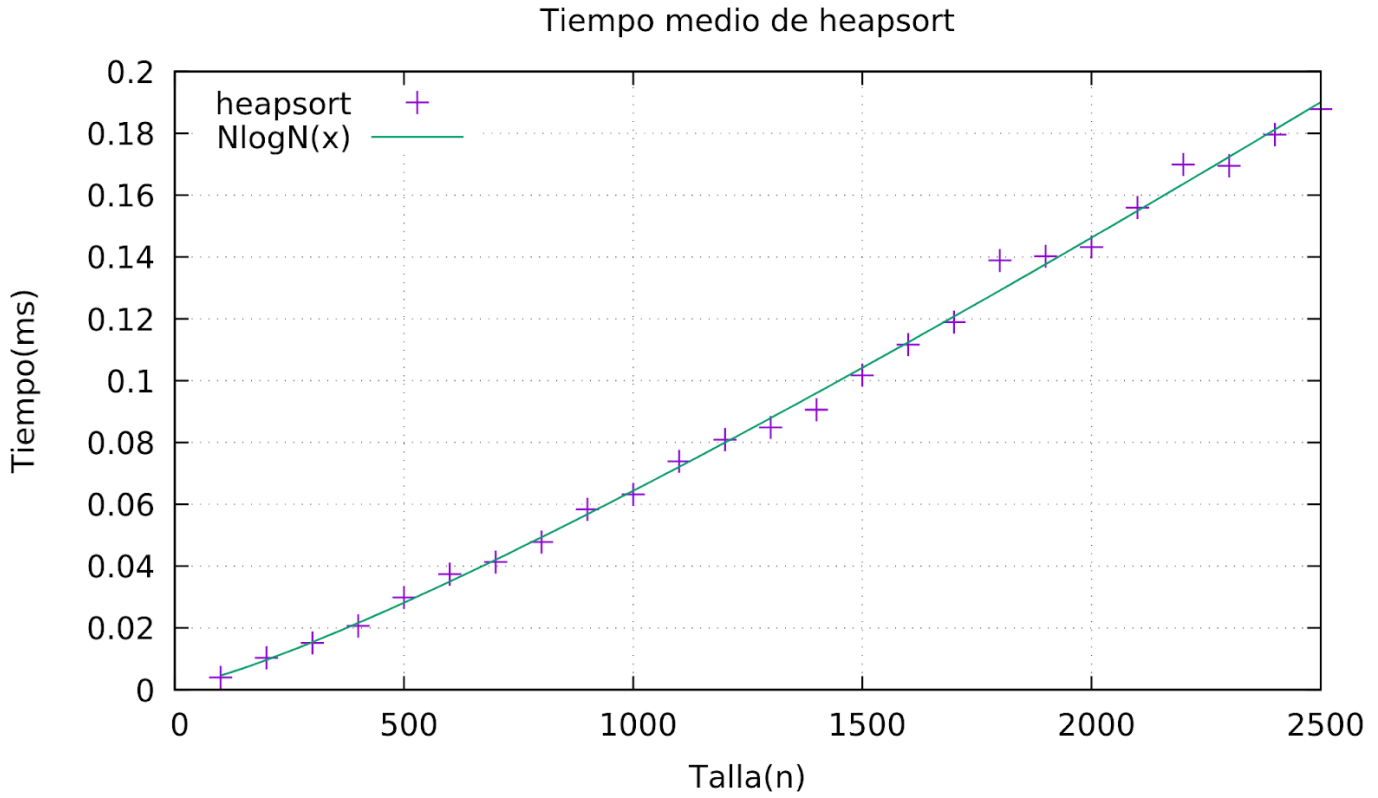
Vamos a ver los siguientes algoritmos:

- Heapsort
- Mergesort
- Quicksort

3.1 Heapsort

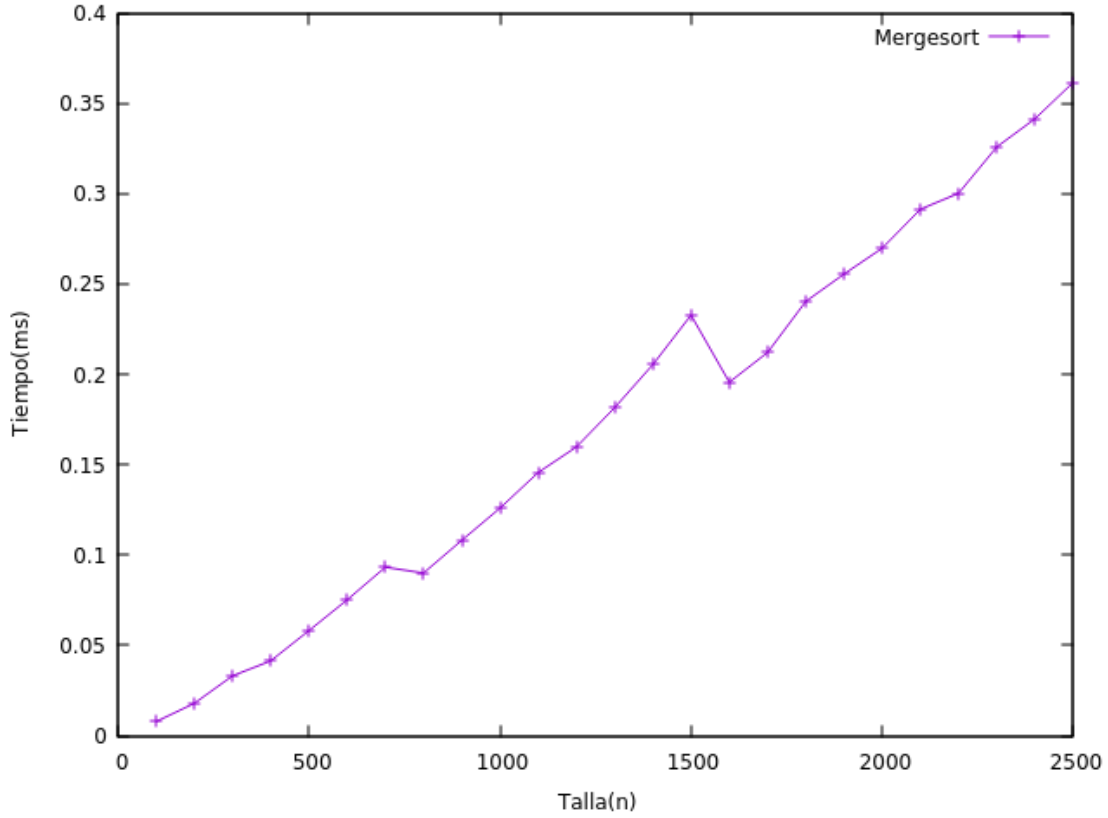
Este algoritmo coloca los elementos en una estructura de árbol y se establece que un "nodo" padre nunca debe ser menor que los hijos. Una vez hecho esto podemos asegurar que el nodo raíz, es el mayor elemento y podemos almacenarlo en la mayor posición del vector.
Esta reordenación se realizará tantas veces como elementos tenga el vector.

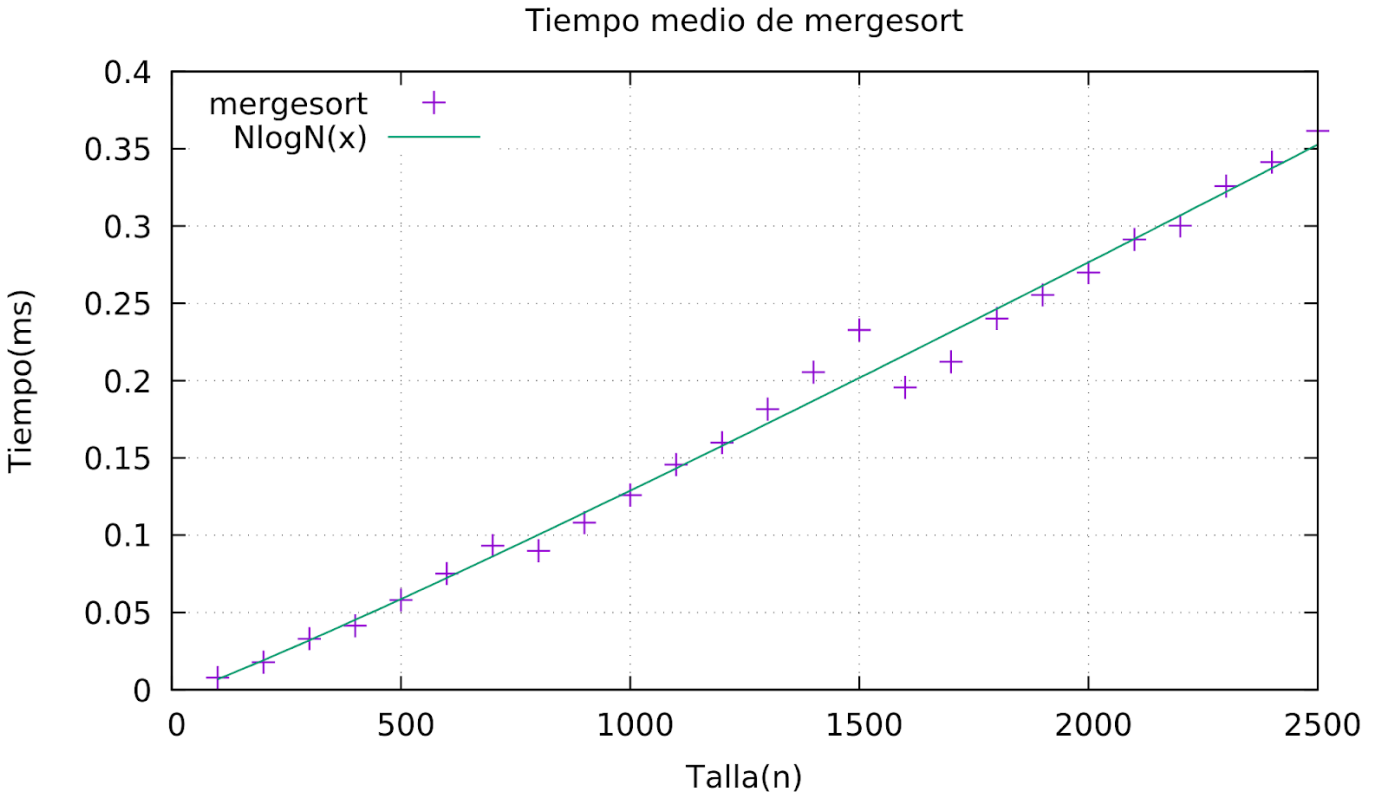




3.2 Mergesort

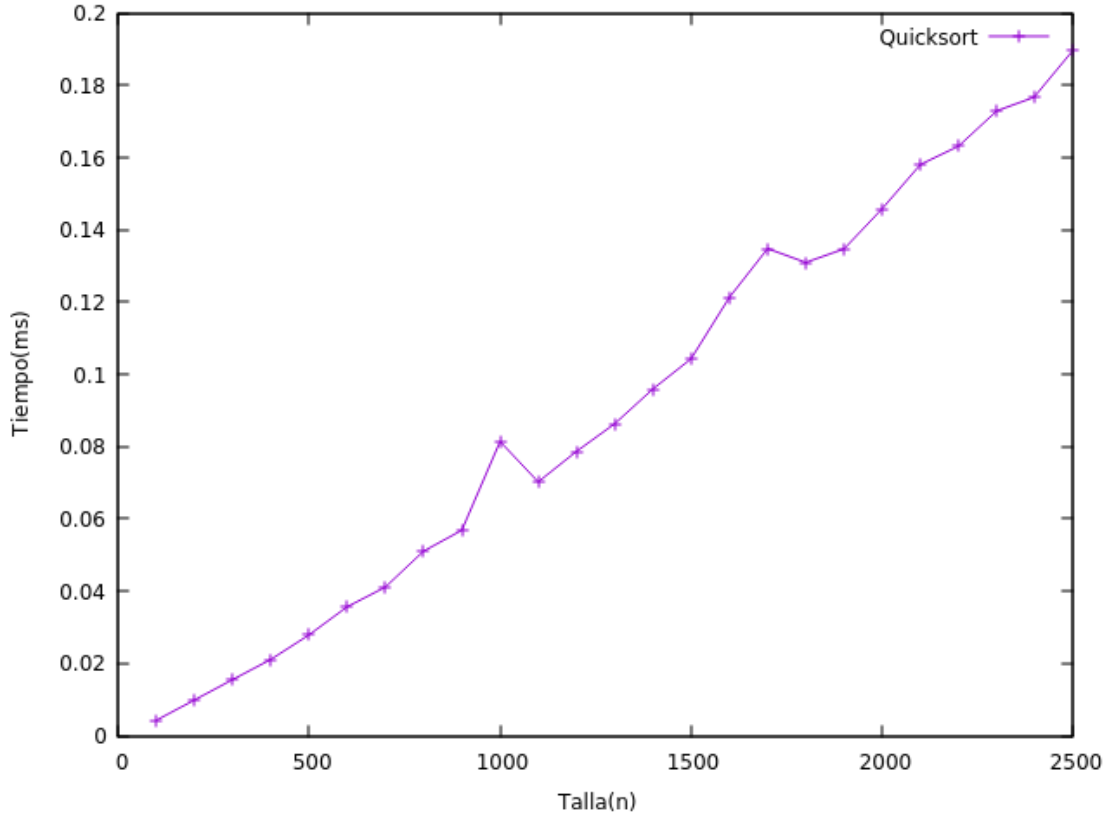
El vector inicial se divide en dos subvectores y se llama a la función de ordenación de forma recursiva. Cuando los están ordenados se llama a una función que va comparando los primeros elementos de cada subvector y colocándolos en su posición, aumentando el puntero y continuando hasta que el vector final ordenado.

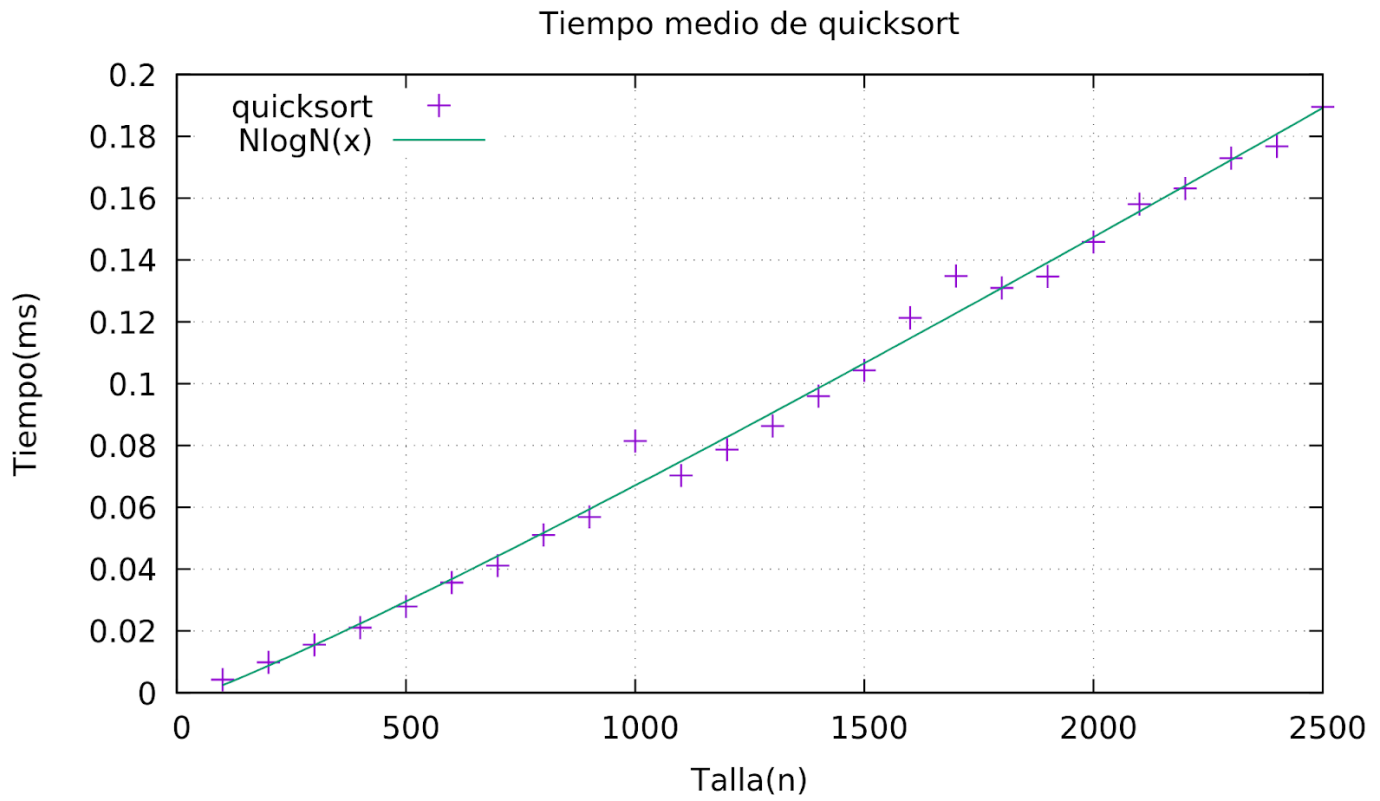




3.3 Quicksort

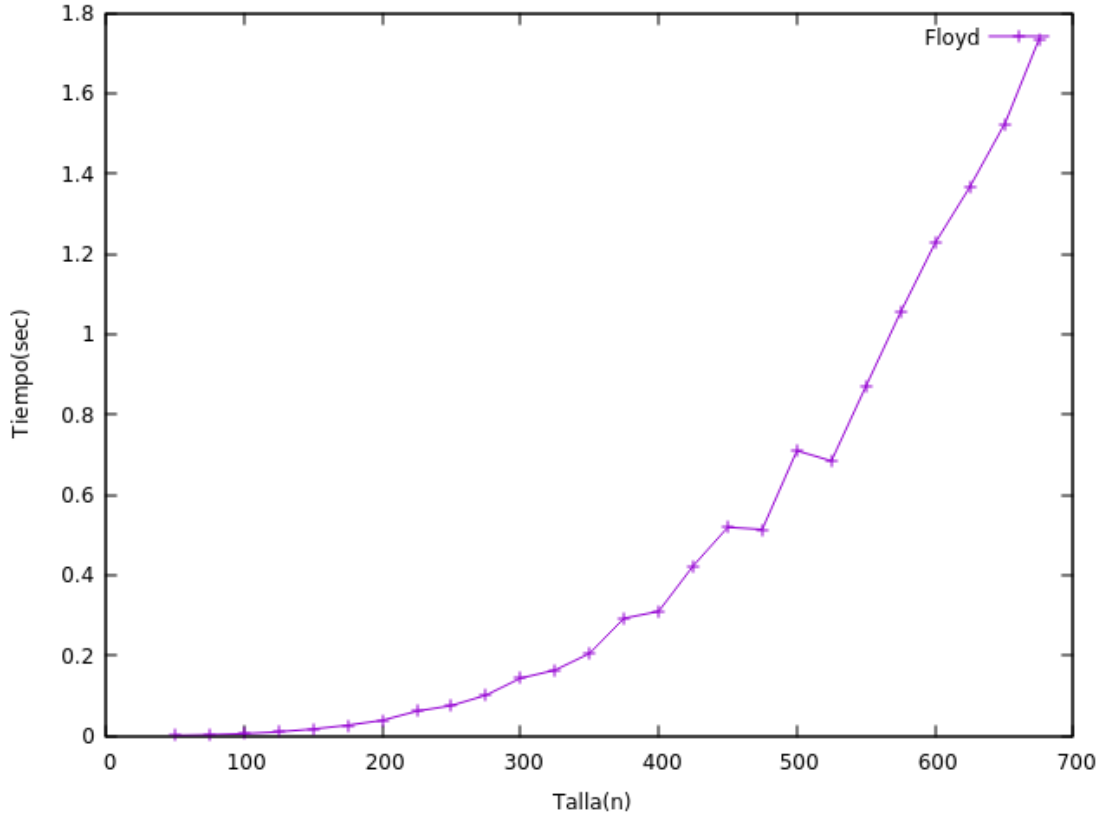
Para ordenar el vector decidimos mediante un criterio un elemento pivote a partir del cual deberemos cumplir la regla de que todo elemento a la izquierda del pivote debe ser menor y todo elemento a la derecha debe ser mayor a éste.
Esto se realiza mediante llamadas recursivas al algoritmo Quicksort.

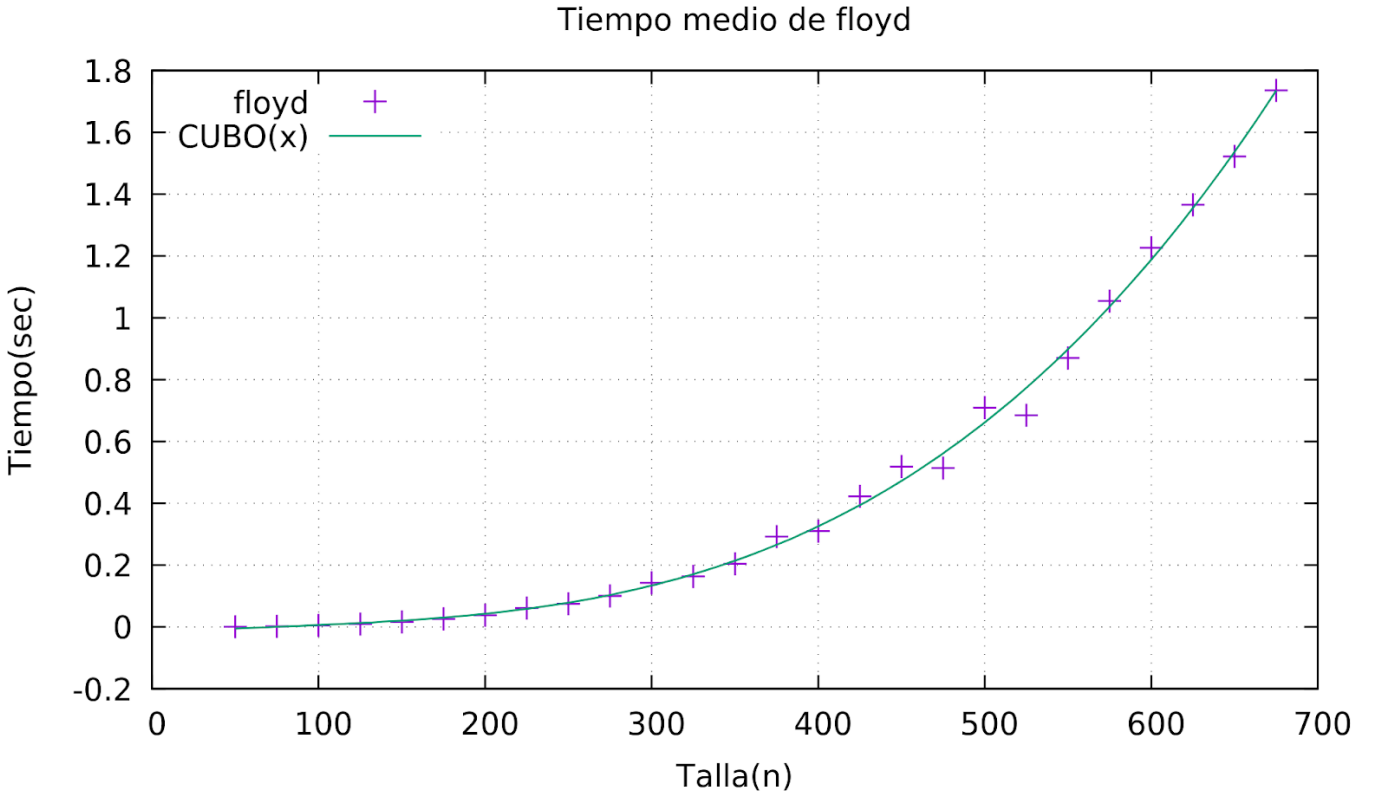




4 Floyd

Este algoritmo encuentra el camino mínimo entre dos vértices de un grafo ponderado mediante una matriz que crea y rellena. Calcula todos los caminos mínimos de los posibles vértices del grafo.





5 Hanoi

El algoritmo resuelve el problema de las torres de Hanoi de forma recursiva.
La función se llama a sí misma dando el número de discos que se desean mover la columna en la que están dichos discos y la columna a la cual se desean llevar.