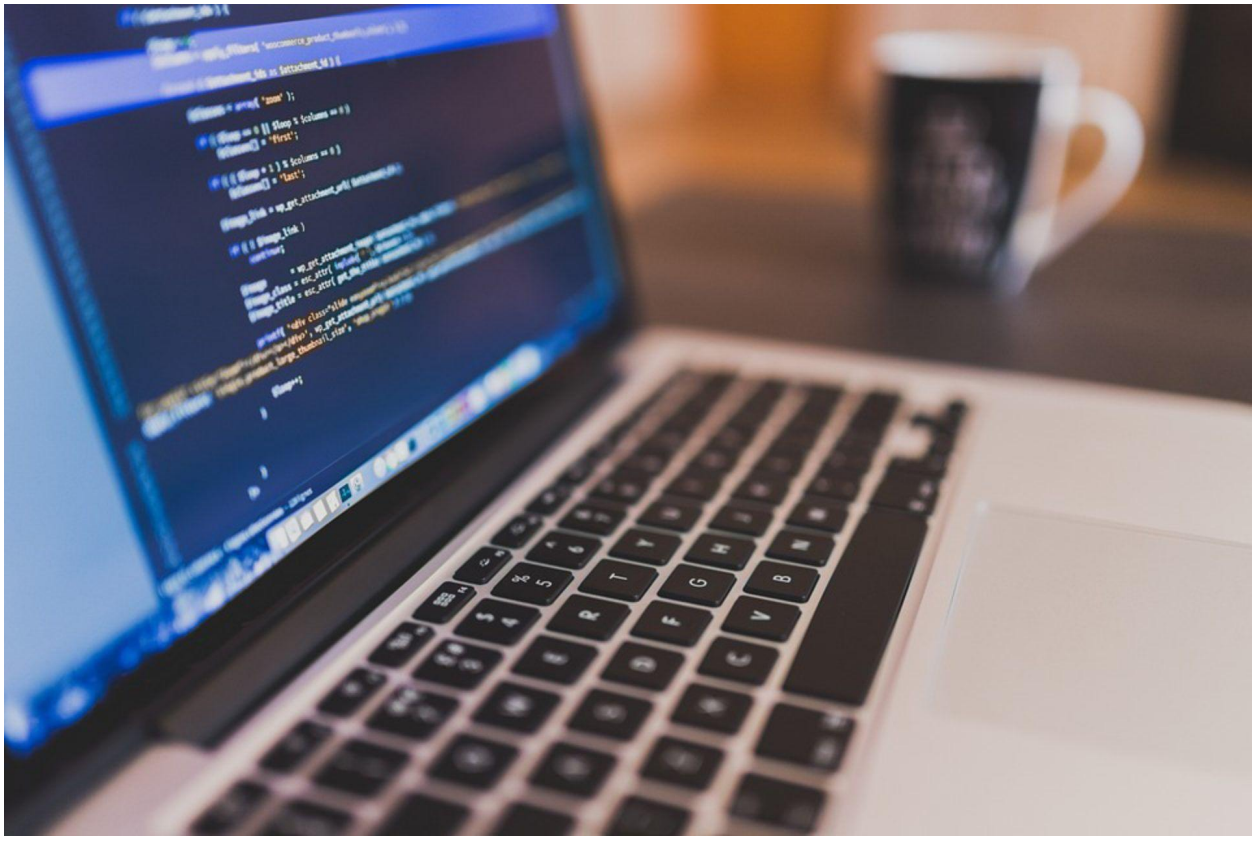
Memoria de la práctica



Antonio Pablo García Sanabria y José María González Abad

Práctica 1

Fundamentos de Análisis de Algoritmos

Universidad de Huelva

18/04/2021

Índice

1. Introducción. Algoritmo de Búsqueda secuencial

Un algoritmo de búsqueda secuencial es aquel que reconoce cada valor de un vector individualmente y pasa al siguiente. Es uno de los algoritmos más básicos que se usan a la hora de encontrar valores en un arreglo.

2. Cálculo del tiempo teórico

A partir de la expresión del algoritmo, habrá que obtener la expresión de la función complejidad temporal (*T(n)*) correspondiente al algoritmo. Se aplicarán las reglas conocidas para contar el número de operaciones que realiza un algoritmo. Este valor será expresado como una función de *T(n)* que dará el número de operaciones requeridas para un caso concreto del problema caracterizado por tener un tamaño *n*. El análisis lo realizaremos para los casos mejor, peor y medio.

2.1. Pseudocódigo y análisis de coste

El algoritmo de búsqueda secuencial que vamos a usar es este (escrito en pseudocódigo), toma los parámetros *Valor* (valor a buscar), *A* (vector en el que buscar) y *tamano* (talla del vector).

```
Algoritmo BusquedaSecuencial(Valor; A; tamano);
        comienza
                i ← 1;
                mientras (A[i] != Valor) AND (i ≤ tamano) hacer
                        i ← i +1;
                fmientras;

                si (A[i] = Valor) hacer
                        BusquedaSecuencial ← i;
                si no hacer
                        BusquedaSecuencial ← -1;
                fsi;
        termina
```

Mirando al pseudocódigo del algoritmo podemos comprobar que ciertamente, tiene 8 operaciones fijas y además es necesario contar las 6 que se repiten en cada iteración dado el tamaño del arreglo. Sin embargo, como en este experimento tenemos diferentes caso debido a la naturaleza del algoritmo nos quedarían diferentes fórmulas de operaciones

elementales según los casos: peor, medio y mejor. La diferencia de estos casos es la cantidad de valores del vector que debe recorrer y reconocer el algoritmo:

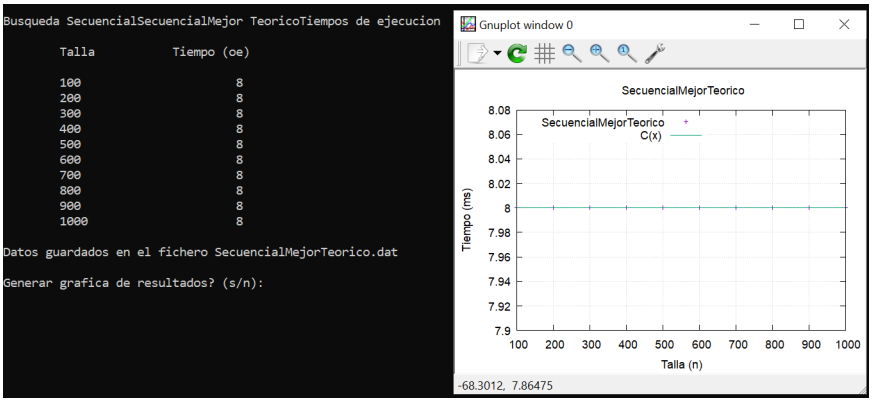
En caso de que sea el peor (se debe recorrer el vector entero), la complejidad se calcula por: $T(n) = 6n + 8$

En caso de que sea el medio (cada valor tiene la misma probabilidad de ser el buscado), la complejidad se calcula por: $T(n) = 3n + 8$

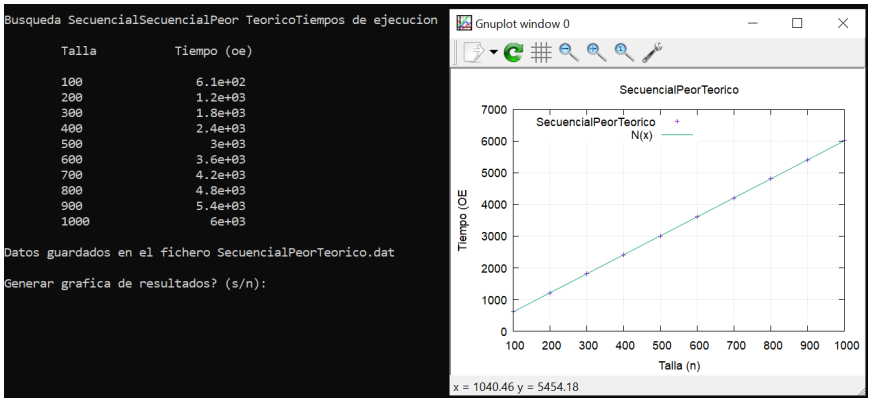
En caso de que sea el mejor (recorre solo el primer elemento), la complejidad se calcula por: $T(n) = 8$

2.2. Tablas y Gráficas de coste

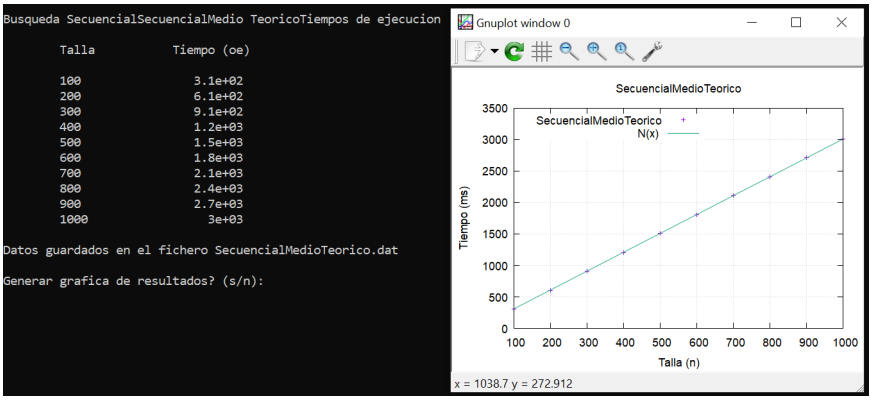
Caso mejor



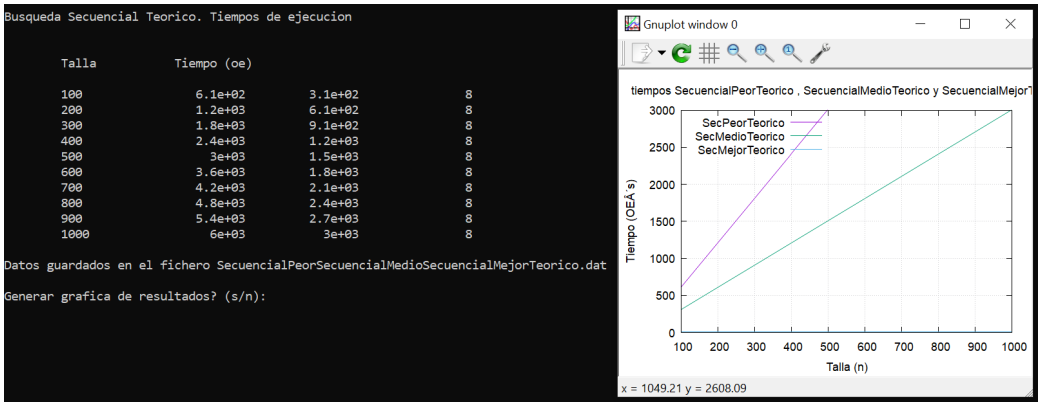
Caso peor



Caso medio



Comparación de los casos teóricos



2.3. Conclusiones

Podemos afirmar sin equivocarnos que el algoritmo sigue un modelo de crecimiento lineal. El coste temporal del algoritmo depende del tamaño del problema, pero incluso búsquedas del mismo tamaño pueden tener diferentes costes dependiendo de la ubicación del elemento a buscar. Aun así, al ser la función complejidad lineal, el algoritmo siempre sigue su modelo de crecimiento lineal, independientemente del caso en el que nos encontremos.

3. Cálculo del tiempo experimental

3.1. Tablas y Gráficas de coste

Caso mejor

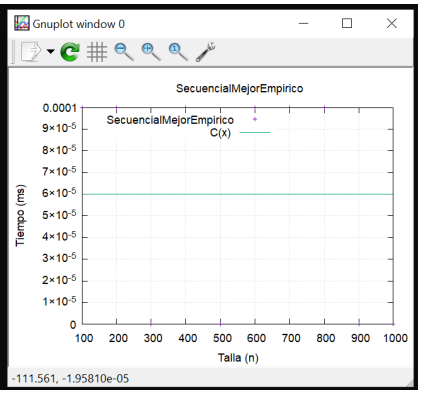
```
Busqueda SecuencialSecuencialMejor EmpÝricoTiempos de ejecucion

        Talla           Tiempo (milisegundos)

        100                0.0001
        200                0.0001
        300                     0
        400                0.0001
        500                     0
        600                0.0001
        700                0.0001
        800                0.0001
        900                     0
        1000                    0

Datos guardados en el fichero SecuencialMejorEmpirico.dat

Generar grafica de resultados? (s/n): s
```



Caso peor

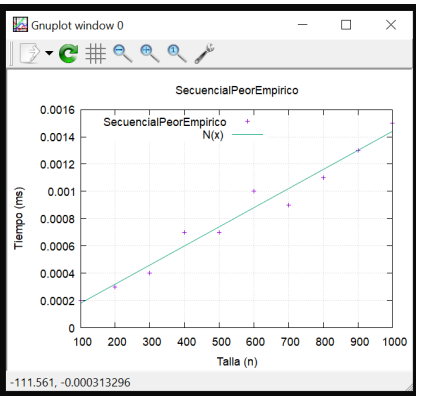
```
Busqueda SecuencialSecuencialPeor EmpÝricoTiempos de ejecucion

        Talla           Tiempo (milisegundos)

        100                0.0002
        200                0.0003
        300                0.0004
        400                0.0007
        500                0.0007
        600                 0.001
        700                0.0009
        800                0.0011
        900                0.0013
        1000               0.0015

Datos guardados en el fichero SecuencialPeorEmpirico.dat

Generar grafica de resultados? (s/n): s
```



Caso medio

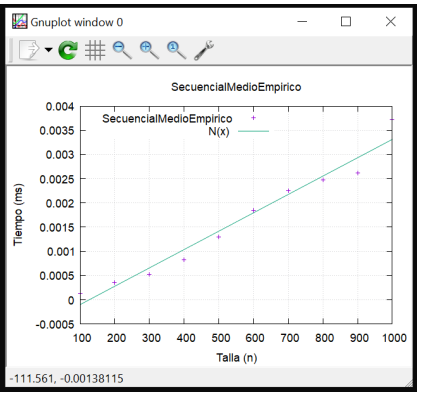
```
Busqueda SecuencialSecuencialMedio EmpÝricoTiempos de ejecucion

        Talla           Tiempo (milisegundos)

        100               0.00013
        200               0.00036
        300               0.00053
        400               0.00083
        500                0.0013
        600                0.0019
        700                0.0023
        800                0.0025
        900                0.0026
        1000               0.0037

Datos guardados en el fichero SecuencialMedioEmpirico.dat

Generar grafica de resultados? (s/n): s
```



Comparación de los casos empíricos

3.2. Conclusiones

Los resultados en el caso experimental parecen desproporcionados, la explicación de esto es por la cola de operaciones que tiene el CPU por delante de las operaciones de nuestro algoritmo. Un ejemplo sería las operaciones que está ejecutando nuestro IDE o los procesos background de nuestro sistema operativo.

4. Comparación de los resultados teórico y experimental

La diferencia general de los dos resultados es que estamos midiendo dos unidades diferentes: operaciones elementales (en el caso teórico) y tiempo (en el caso empírico). Ciertamente, estas operaciones elementales van a ser muchas más o menos según en el caso en el que nos encontremos y la talla del vector a procesar.

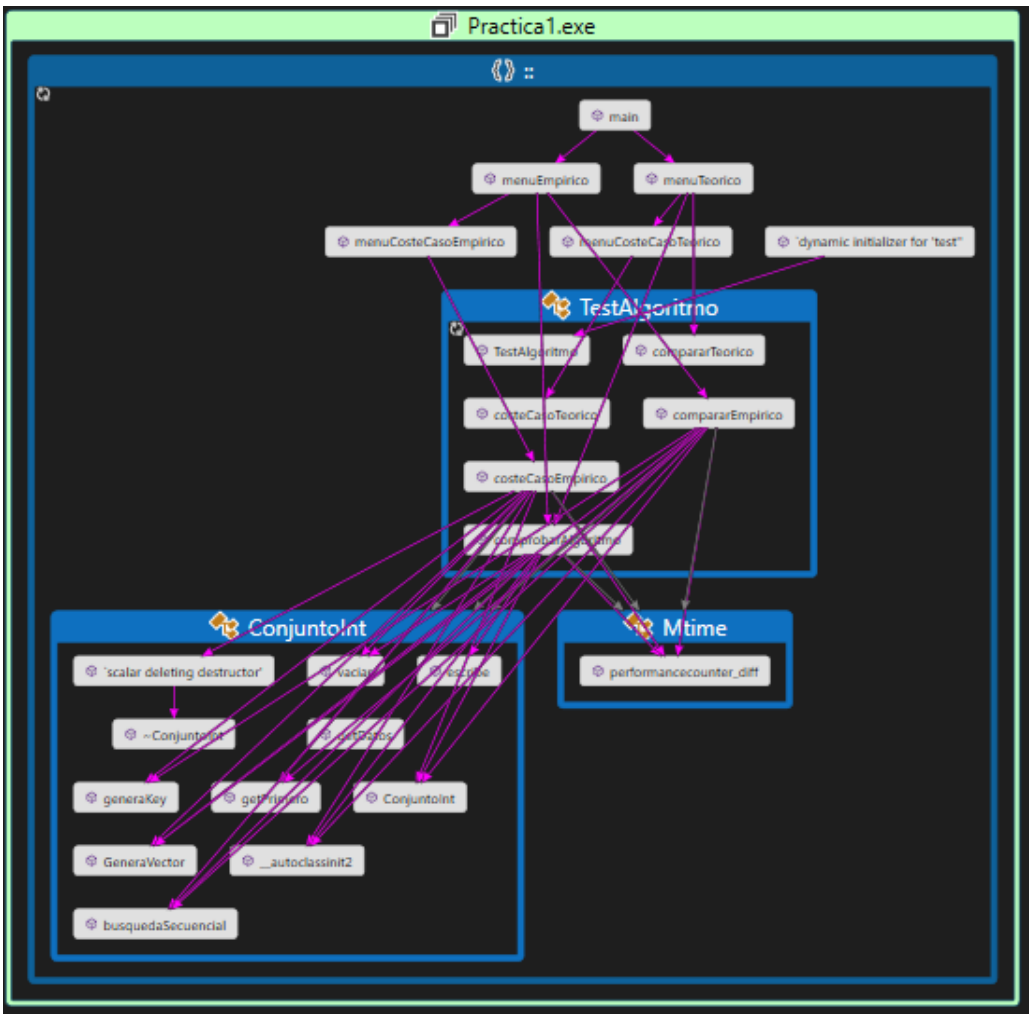
En los resultados teóricos podemos ver las cantidades de operaciones elementales que toma con cada caso, sin embargo, en el empírico podemos comprobar exactamente cuanto tiempo se ha tomado nuestro CPU en llevar a cabo dichas operaciones elementales.

5. Diseño de la aplicación

La aplicación muestra el estudio de las operaciones que realiza el algoritmo de búsqueda secuencial y el tiempo que tardan en ejecutarse según los diferentes casos. Está compuesta de los archivos de encabezado, de sus correspondientes archivos de origen, *ConjuntoInt.h/.cpp*, *Constantes.h*, *Mtime.h/.cpp*, *TestAlgoritmo.h/.cpp* y *Principal.cpp*, además de otra serie de archivos donde se recogen los datos obtenidos en la aplicación y las gráficas generadas a partir de dichos datos.

La clase *ConjuntoInt* nos ayuda a trabajar los vectores para el estudio, con diferentes métodos para poder trabajar con ellos, como generar o vaciar vectores, generar elementos aleatorios u obtener el primer elemento del vector generado. En la clase *TestAlgoritmo* implementamos los métodos necesarios para poder realizar los diferentes estudios de los casos. Mediante la clase *Mtime* calculamos el tiempo que tarda en ejecutarse cada caso.

En el fichero *Principal* se lleva a cabo la ejecución de la aplicación, mediante la navegación a través de varios menús en los que podemos decidir que estudio en concreto queremos realizar. A continuación, mostramos el mapa de código generado por la aplicación IDE.



6. Conclusiones y valoraciones personales de la práctica

La práctica sienta las bases para poder hacer un correcto uso de los algoritmos de búsqueda secuencial para futuras situaciones. En ella observamos las operaciones que se deben hacer para realizar este tipo de búsqueda, por otra parte podemos ver el tiempo real de ejecución del programa al resolver el problema.