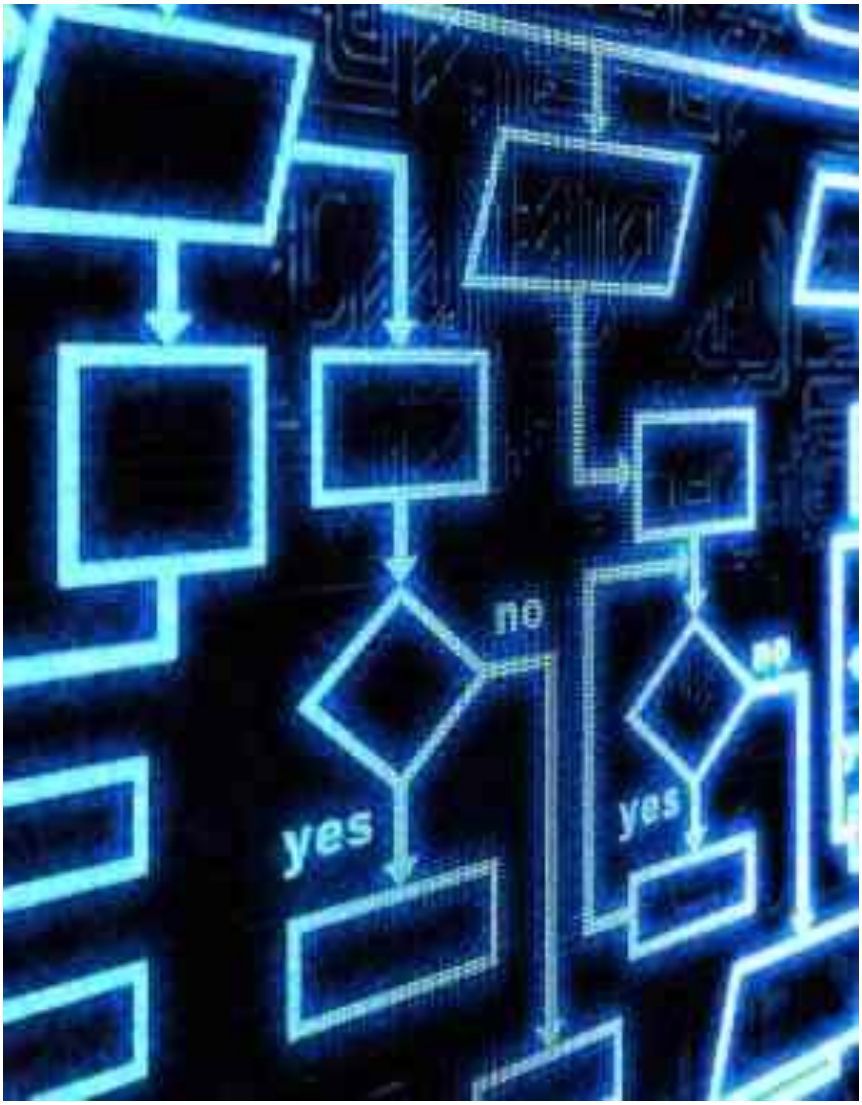


Análisis de un algoritmo de busqueda secuencial

FAA

Por: Ismael Da Palma Fernández

ÍNDICE

3. Introducción. Algoritmo de búsqueda secuencial

El objetivo de esta práctica es realizar el estudio teórico y experimental de un algoritmo de búsqueda secuencial, para la parte experimental usaremos un lenguaje de programación.

El lenguaje que se ha escogido es el de C++ y el entorno de desarrollo integrado Visual Studio 2019. Adicionalmente hemos utilizado Gnuplot para graficar los resultados.

4. Cálculo del tiempo teórico:

4.1. Pseudocódigo (código) y análisis de coste

El algoritmo se dio directamente implementado en C++. Sus parámetros formales son: un array de enteros (T) que contiene un conjunto de valores, un entero (n) que indica cuantos elementos hay en la tabla y un entero (valor) que será la clave de búsqueda.
Esta función devuelve la posición dentro del array de la clave de búsqueda; en caso de haber varias, se queda con la primera ocurrencia. Si la clave no se encuentra, devuelve -1 .

Las sentencias del algoritmo son:

líneas	
	`int BusquedaSecuencial(int T[],int n,int valor)`
	`{`
1.	`int i=0;`
2.	`while (T[i] != valor && i<n) {`
3.	`i=i+1;`
4.	`}`
5.	`if (T[i]==valor)`
6.	`return i;`
7.	`else return -1;`
	`}`

Inicia un contador "i" a 0, después itera hasta que se acabe el array o encuentre la clave.

Al final comprueba si salió del bucle con "i" siendo el índice conveniente, en ese caso regresa "i", en caso contrario devuelve -1.

Conteo de operaciones elementales ==> $TBSecuencial(n) = T_{Asig} + T_{Bucle} + T_{Si}$

TAsig -> Se realiza en la línea 1 una operación de asignación $(+1)$

TBucle -> (En las lineas 2 y 3) La condición del bucle tiene 4 operaciones, el cuerpo del bucle tiene 2. En total son 6 operaciones por repetición y otras 4 de la condición del bucle para la salida. (4+6n)

TSi -> La condición es de 2 operaciones y en ambos casos del if las sentencias tienen 1 operación. (2+1)

Haciendo recuento, el coste temporal de este algoritmo es de 6 operaciones de repetición y 8 operaciones fijas (constantes). Si la Key no está en la tabla, el bucle se repetirá tantas veces como elementos tenga el array. Si la Key está al principio, no llega a entrar al bucle, el coste es siempre el mismo.

Mediante el análisis del algoritmo, se obtienen las expresiones que determinan el número de operaciones elementales necesarias para cada uno de los casos

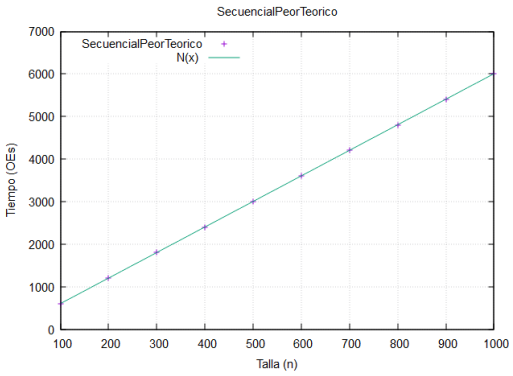
- **Caso peor:** la key no está en el array $T(n) = 8 + 6n \quad T(n) \in O(n)$
- **Caso medio:** la key está en el centro del array $T(n) = 8 + \frac{6}{2}n \quad T(n) \in O(n)$
- **Caso mejor:** la key está en el primer elemento $T(n) = 8 \quad T(n) \in O(1)$

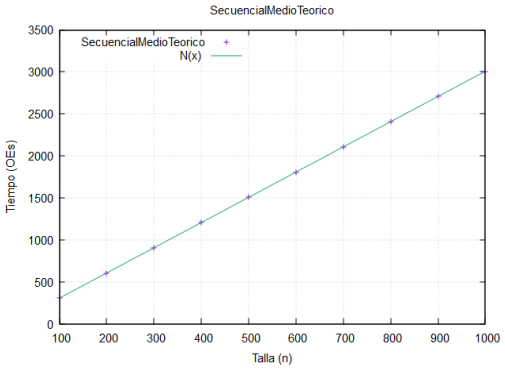
4.2. Tablas y Gráficas de coste

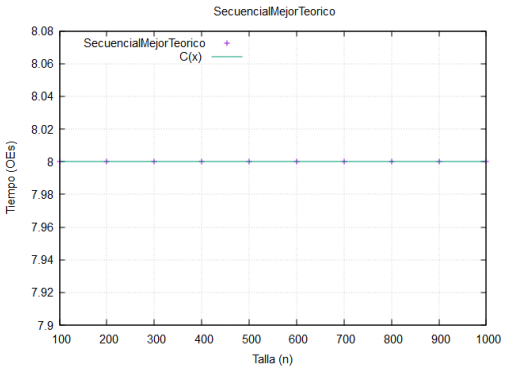
```
C:\Users\ismae\source\repos\Practica_1_FAA\Debug\Practica_1_FAA.exe

Busqueda SecuencialSecuencialPeor TeoricoTiempos de ejecucion

       Talla          Tiempo (oe)

       100              6.1e+02
       200              1.2e+03
       300              1.8e+03
       400              2.4e+03
       500                3e+03
       600              3.6e+03
       700              4.2e+03
       800              4.8e+03
       900              5.4e+03
       1000               6e+03

Datos guardados en el fichero SecuencialPeorTeorico.dat

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

```
C:\Users\ismae\source\repos\Practica_1_FAA\Debug\Practica_1_FAA.exe

Busqueda SecuencialSecuencialMedio TeoricoTiempos de ejecucion

       Talla          Tiempo (oe)

       100              3.1e+02
       200              6.1e+02
       300              9.1e+02
       400              1.2e+03
       500              1.5e+03
       600              1.8e+03
       700              2.1e+03
       800              2.4e+03
       900              2.7e+03
       1000               3e+03

Datos guardados en el fichero SecuencialMedioTeorico.dat

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

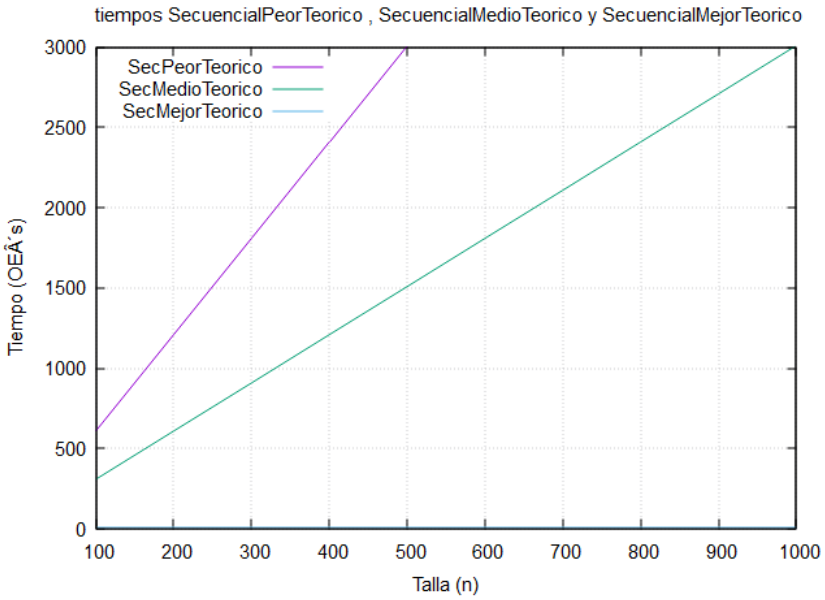
```
C:\Users\ismae\source\repos\Practica_1_FAA\Debug\Practica_1_FAA.exe

Busqueda SecuencialSecuencialMejor TeoricoTiempos de ejecucion

       Talla          Tiempo (oe)

       100                8
       200                8
       300                8
       400                8
       500                8
       600                8
       700                8
       800                8
       900                8
       1000               8

Datos guardados en el fichero SecuencialMejorTeorico.dat

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








```
C:\Users\ismae\source\repos\Practica_1_FAA\Debug\Practica_1_FAA.exe

Busqueda Secuencial Teorico. Tiempos de ejecucion

       Talla          Tiempo (oe)

       100              6.1e+02          3.1e+02                8
       200              1.2e+03          6.1e+02                8
       300              1.8e+03          9.1e+02                8
       400              2.4e+03          1.2e+03                8
       500                3e+03          1.5e+03                8
       600              3.6e+03          1.8e+03                8
       700              4.2e+03          2.1e+03                8
       800              4.8e+03          2.4e+03                8
       900              5.4e+03          2.7e+03                8
       1000               6e+03            3e+03                8
```



4.3. Conclusiones

El coste temporal del algoritmo depende de la talla del array y de la instancia en concreto. Búsquedas con la misma talla tienen costes diferentes, dependiendo de dónde se encuentre la clave.

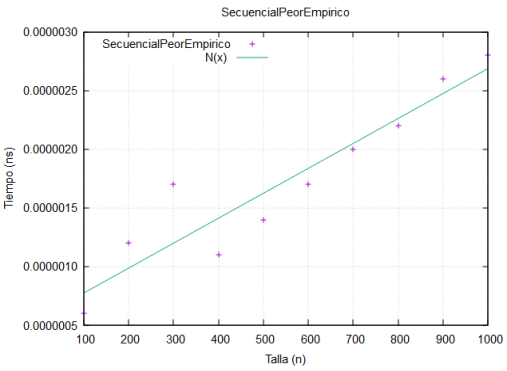
Aún en el peor caso, este algoritmo sigue un modelo de crecimiento lineal. Este crecimiento puede verse representado como rectas en la gráfica. La función complejidad es lineal.

5. Cálculo del tiempo experimental:

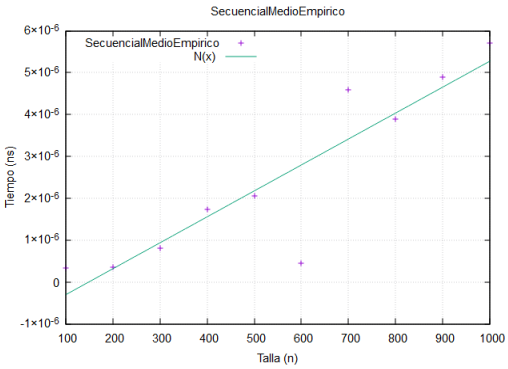
Para cada medición se generan nuevos conjuntos de datos aleatorios, pero solo se mide el tiempo de búsqueda. En el caso medio este proceso se repite varias veces y se toma la media.

5.1. Tablas y Gráficas de coste

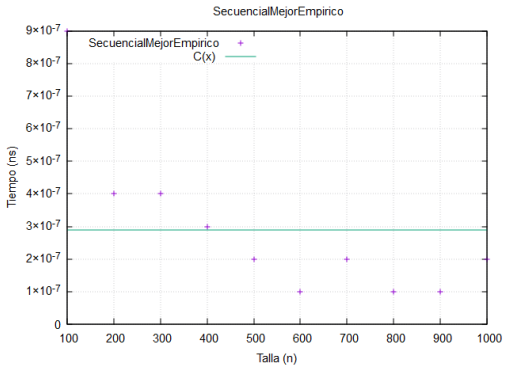
```
C:\Users\ismae\source\repos\Practica_1_FAA\Debug\Practica_1_FAA.exe

Busqueda SecuencialSecuencialPeor EmpiricoTiempos de ejecucion

        Talla           Tiempo (ns)

        100                 6e-07
        200               1.2e-06
        300               1.7e-06
        400               1.1e-06
        500               1.4e-06
        600               1.7e-06
        700                 2e-06
        800               2.2e-06
        900               2.6e-06
        1000              2.8e-06

Datos guardados en el fichero SecuencialPeorEmpirico.dat

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



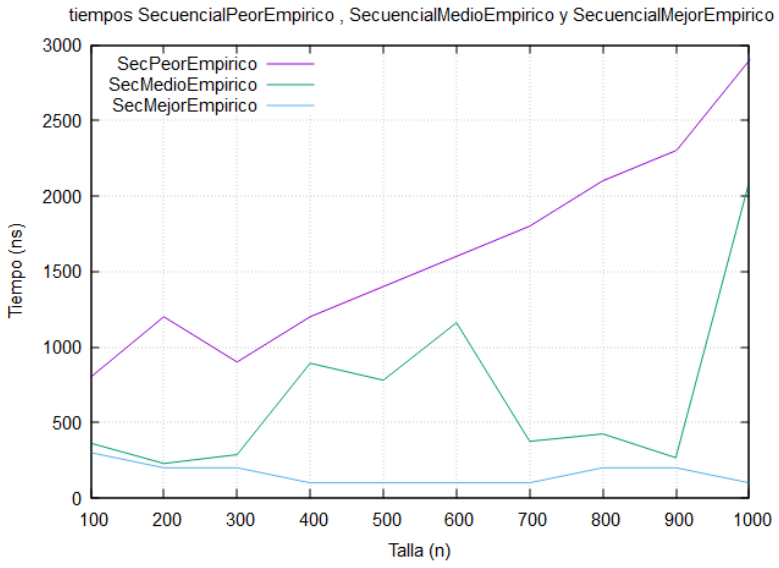

```
C:\Users\ismae\source\repos\Practica_1_FAA\Debug\Practica_1_FAA.exe

Busqueda SecuencialSecuencialMedio EmpiricoTiempos de ejecucion

        Talla           Tiempo (ns)

        100               3.5e-07
        200               3.6e-07
        300               8.2e-07
        400               1.7e-06
        500               2.1e-06
        600               4.6e-07
        700               4.6e-06
        800               3.9e-06
        900               4.9e-06
        1000              5.7e-06

Datos guardados en el fichero SecuencialMedioEmpirico.dat
```




```
C:\Users\ismae\source\repos\Practica_1_FAA\Debug\Practica_1_FAA.exe

Busqueda SecuencialSecuencialMejor EmpiricoTiempos de ejecucion

        Talla           Tiempo (ns)

        100               9e-07
        200               4e-07
        300               4e-07
        400               3e-07
        500               2e-07
        600               1e-07
        700               2e-07
        800               1e-07
        900               1e-07
        1000              2e-07

Datos guardados en el fichero SecuencialMejorEmpirico.dat

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



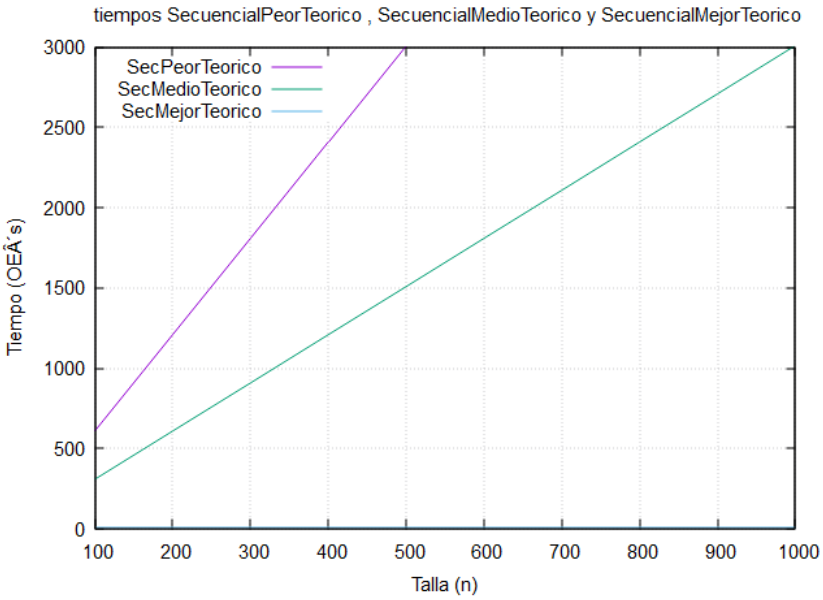

```
C:\Users\ismae\source\repos\Practica_1_FAA\Debug\Practica_1_FAA.exe

Busqueda Secuencial Empirico. Tiempos de ejecucion

        Talla           Tiempo (ns)

        100                 8e+02           3.6e+02         3e+02
        200               1.2e+03           2.3e+02         2e+02
        300                 9e+02           2.9e+02         2e+02
        400               1.2e+03           8.9e+02         1e+02
        500               1.4e+03           7.8e+02         1e+02
        600               1.6e+03           1.2e+03         1e+02
        700               1.8e+03           3.8e+02         1e+02
        800               2.1e+03           4.2e+02         2e+02
        900               2.3e+03           2.7e+02         2e+02
        1000              2.9e+03           2.1e+02         1e+02
```



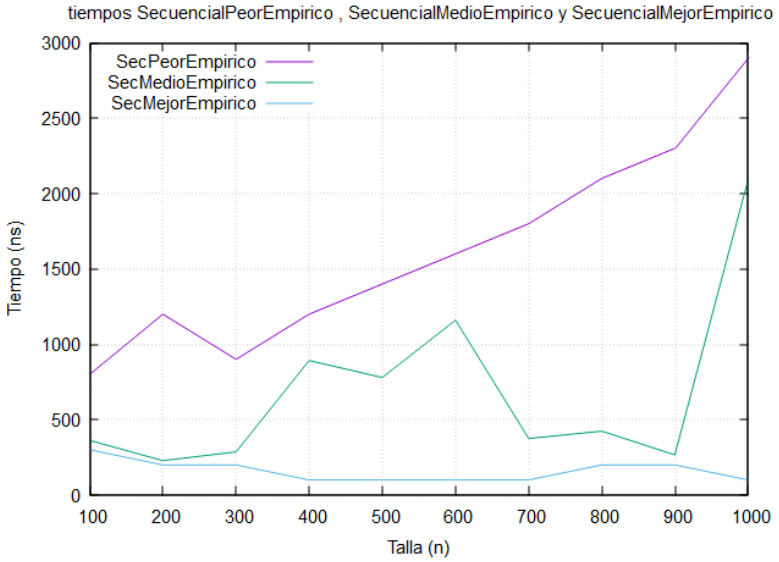
5.2. Conclusiones

Los resultados individuales muestran un buen grado de dispersión, sin elementos anómalos (en algunas ocasiones hay). Siendo esta dispersión más apreciable en el caso medio. En el caso mejor por separado, aparenta una gran dispersión; pero la diferencia es mínima y se debe al límite de precisión con la que se mide el tiempo transcurrido.

6. Comparación de los resultados teóricos y experimental



El modelo de coste temporal analizado describe correctamente al algoritmo, puesto que las fórmulas se ajustan a los resultados obtenidos.



En el caso medio del empírico suele haber cierta tendencia a la dispersión cuando aumenta la talla, llegando en algunas tallas a ser más eficiente que el caso mejor y más ineficiente que el caso peor.

La comparativa empírica presenta la misma relación que la comparativa teórica, luego el modelo de coste analizado se ajusta correctamente a los resultados empíricos.

7. Diseño de la aplicación

El método principal con la interfaz de menús está en el Principal.cpp. Desde allí se llama a los métodos de TestAlgoritmo, la clase que realiza los cálculos teóricos y empíricos. Además, TestAlgoritmo usa la clase ConjuntoInt para la ejecución del algoritmo a estudiar y Mtime para medir el tiempo transcurrido.

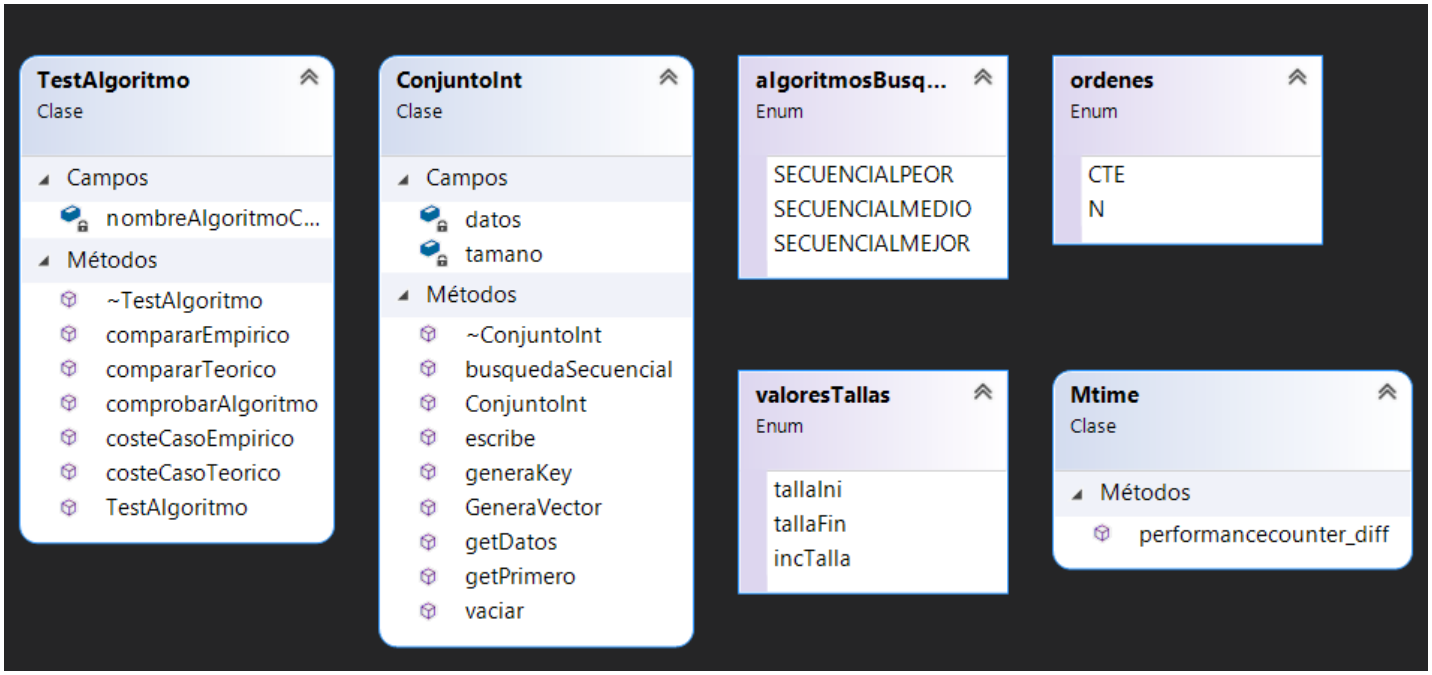
ConjuntoInt permite crear arrays de enteros aleatorios y, posteriormente, buscar un elemento dada una clave especificada. Adicionalmente, puede generar claves de búsqueda (generarKey).

Mtime permite determinar el tiempo de procesador que tarda en ejecutarse el algoritmo para el caso empírico, puede haber variaciones de tiempo debido a la dependencia de carga de trabajo del ordenador.

Constantes contiene una serie de constantes útiles para el desarrollo de la práctica.

TestAlgoritmo tiene los siguientes métodos principales:

- **costeCasoTeorico:** Recibe por parámetro el caso a estudiar (numerocaso) y lo calcula de forma teórica. Muestra por pantalla y guarda en un archivo los valores que toma la función complejidad para distintas tallas y casos.
- **compararTeorico:** Recibe por parámetro los tres casos y los comparara teóricamente. Muestra por pantalla y guarda en un archivo los valores de la función complejidad para todos los casos y diferentes tallas
- **costeCasoEmpirico:** Recibe por parámetro (numerocaso) el caso a estudiar empíricamente. Genera un conjunto de datos aleatorios basado en el tiempo que transcurre entre el inicio y el fin de la llamada al algoritmo de búsqueda. En el caso medio, repite varias veces el proceso para una misma talla y hace una media. Muestra los resultados por pantalla y los guarda en un fichero.
- **compararEmpirico**: Recibe por parámetro los tres casos y los compara empíricamente. Realiza las mismas mediciones de los tres casos, las muestra por pantalla y las guarda en un archivo.



8. Conclusiones y valoraciones personales de la práctica

Esta práctica ha sido de utilidad para comprender la utilidad del análisis de algoritmos, al confirmarse una clara relación entre los valores calculados y las lecturas obtenidas.

También se ha practicado la escritura de ficheros y el uso del diseño modular.