PRÁCTICA 3

FAA – 1º Ingeniería Informática

Análisis experimental de algoritmos de búsqueda

Alba Márquez Rodríguez

3. Introducción. Algoritmos de Búsqueda.

En esta práctica implementaremos un algoritmo que tiene como objetivo el análisis y estudio de la eficiencia de algoritmos. Utilizaremos las prácticas hechas anteriormente, incluyendo algoritmos de ordenación. Lo que implementaremos nuevo será el estudio empírico correspondiente de 3 algoritmos de búsqueda: secuencial iterativa, binaria iterativa e interpolación iterativa. Por separado y comprándolos.

Los algoritmos de búsqueda consisten en algoritmos que buscan una clave en un array de elementos. Algunos algoritmos requieren que la tabla esté ordenada, como la búsqueda binaria y la interpolación. Además, se asume que los elementos están en una relación de orden < y almacenados en un contenedor lineal (vector o lista). El objetivo es obtener algoritmos de búsqueda lo más eficientes posibles (en tiempo de cálculo y memoria). El tiempo de ejecución viene dado por la función complejidad del número de elementos entre los que realiza la búsqueda (talla de 100 a 1000).

El estudio empírico consiste en el estudio experimental del comportamiento del algoritmo. Midiendo el tiempo en los tres posibles algoritmos. Este variará según la capacidad de cada ordenador y el uso que se esté haciendo en el momento de la memoria.

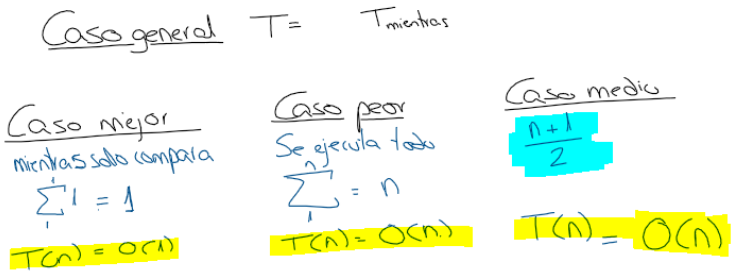
4. Cálculo de los tiempos teóricos:

Aplicando las reglas estudiadas en clase calcularemos el número de operaciones básicas del algoritmo y obtenemos el orden de complejidad. Lo pondremos como una función de T(n). Siendo n las veces que se ejecutará el bucle. Al depender el tiempo de ejecución de la talla, n=talla

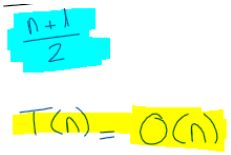
4.1. Pseudócódigos y análisis de coste

Este es el pseudocódigo que se nos da en el enunciado del ejercicio. Para todos los algoritmos cogeremos el caso medio ya que es el que nos interesa en esta práctica, para obtenerlo debemos calcular antes el caso mejor y el peor.

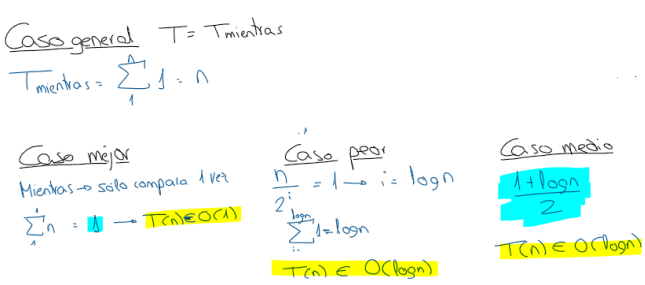
Secuencial Iterativa



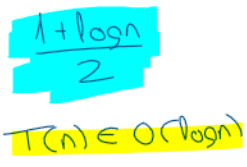
Calculamos y tenemos en cuenta que para el mejor caso el mientras solo realiza una comparación. El caso medio es:



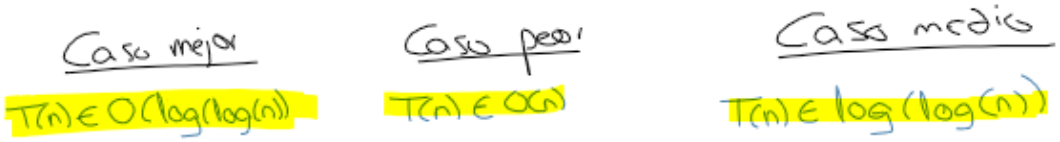
Binaria Iterativa



Para el mejor caso el mientras solo ejecuta una comparación, para el peor ejecuta n/2 comparaciones por lo que el orden es log(n). El caso medio es:



Interpolación Iterativa

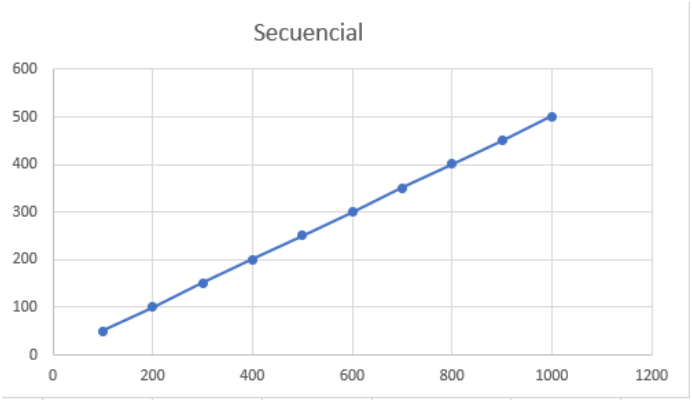


La eficiencia de este algoritmo dependerá de el incremento entre los valores de la tabla. Si estos valores tienen un incremento uniforme será el mejor caso siendo el orden el indicado. Sin embargo, si el incremento es menos uniforme (por ejemplo exponencial) el caso será peor y hará n comparaciones.

4.2. Tablas (ficheros) y Gráficas de coste

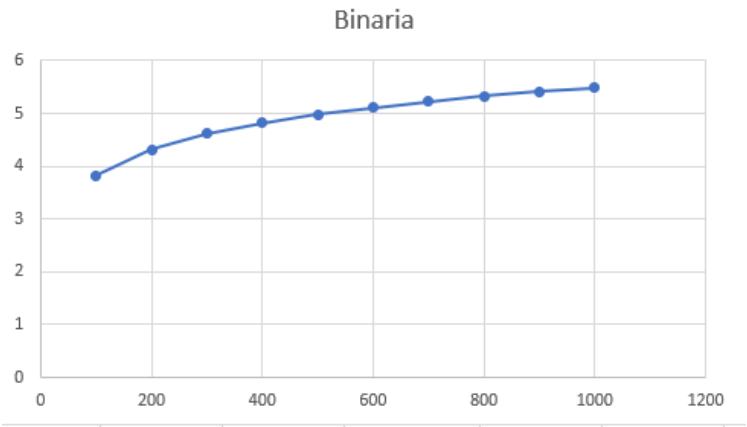
Secuencial

Talla	Secuencial T(n)
100	50,5
200	100,5
300	150,5
400	200,5
500	250,5
600	300,5
700	350,5
800	400,5
900	450,5
1000	500,5



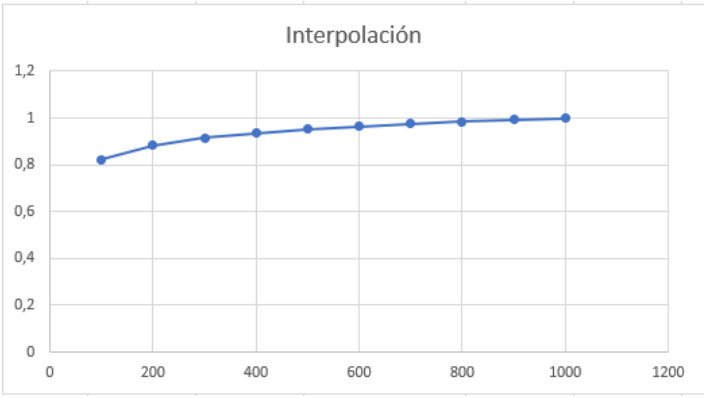
Binaria

Talla	Binaria T(n)
100	3,821928095
200	4,321928095
300	4,614409345
400	4,821928095
500	4,982892142
600	5,114409345
700	5,225605556
800	5,321928095
900	5,406890596
1000	5,482892142



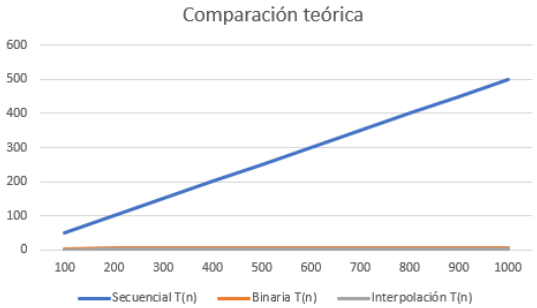
Interpolación

Talla	Interpolación T(n)
100	0,822420223
200	0,883312508
300	0,915337493
400	0,936707533
500	0,952588286
600	0,965146114
700	0,975487464
800	0,984250726
900	0,991836371
1000	0,998511482



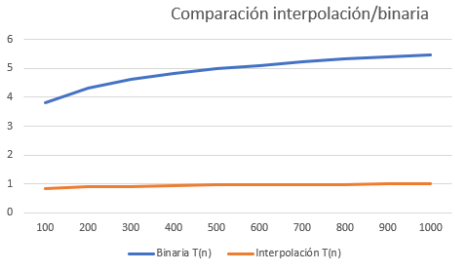
4.3. Conclusiones

Talla	Secuencial T(n)	Binaria T(n)	Interpolación T(n)
100	50,5	3,821928095	0,822420223
200	100,5	4,321928095	0,883312508
300	150,5	4,614409345	0,915337493
400	200,5	4,821928095	0,936707533
500	250,5	4,982892142	0,952588286
600	300,5	5,114409345	0,965146114
700	350,5	5,225605556	0,975487464
800	400,5	5,321928095	0,984250726
900	450,5	5,406890596	0,991836371
1000	500,5	5,482892142	0,998511482



Secuencial tiene un crecimiento secuencial por lo que se espera que sea el menos eficiente frente a los otros dos que tienen un crecimiento logarítmico. Los dos enfrentados al secuencial van a ser mucho más eficientes. Debemos tener en cuenta si los elementos de la tabla tienen un incremento uniforme o no. En el caso de que sea uniforme será mejor utilizar interpolación, sino, búsqueda.

Para analizar si Binaria o Interpolación es más o menos eficiente, hemos elaborado una gráfica con ambos. Hay que tener en cuenta que se toman valores idílicos. En el estudio empírico lo podremos comprobar mejor.



Aquí podemos ver que en principio interpolación sería más eficiente que binaria.

5. Cálculo del tiempo experimental:

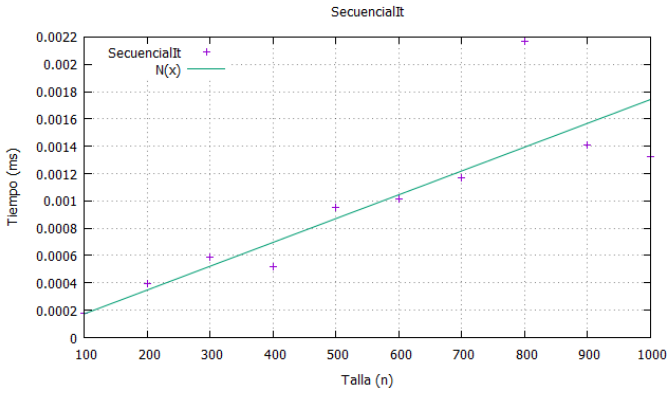
Para el estudio empírico tendremos que medir los recursos empleados (tiempo). Para ello elaboramos pruebas en diferentes condiciones (incremento de la talla del problema) ya que el tiempo de ejecución del algoritmo dependerá del tamaño del problema. Para cada caso la medida computacional será el tiempo de ejecución.

Para el caso “estudio promedio” al ser aleatorio introduciremos un bucle for que tome diferentes valores y luego haga una media. Haremos esto para los tres algoritmos que estudiaremos. Con la constante NUMREPETICIONES. Así la medida final será un promedio de todas las efectuadas para conseguir un resultado lo más empírico y veraz posible. Estos datos se mostrarán en la pantalla de la terminal y guardados en el fichero de datos y en el fichero de gráficos. En el fichero gráficos queda guardado por la clase Gráficas que crea un fichero de gnuplot (.gpl) con el código necesario para que se ejecute con la gráfica correspondiente.

Esto queda implementado como en la anterior práctica.

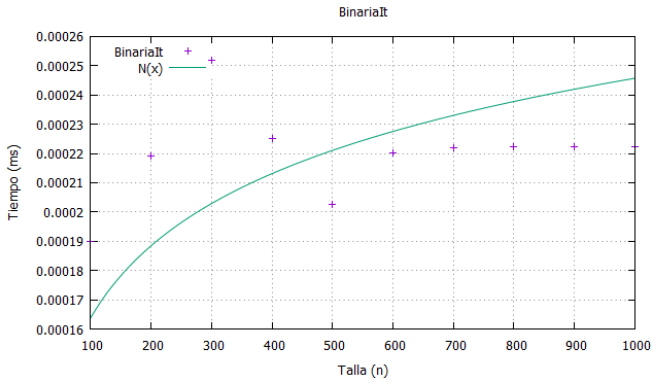
5.1. Tablas (ficheros) y Gráficas de coste

```
*** Busqueda por SecuencialIt ***

Tiempos de ejecucion promedio

        Talla           Tiempo (mseg)

        100                0.00018
        200                 0.0004
        300                0.00059
        400                0.00052
        500                0.00095
        600                  0.001
        700                 0.0012
        800                 0.0022
        900                 0.0014
        1000                0.0013
```



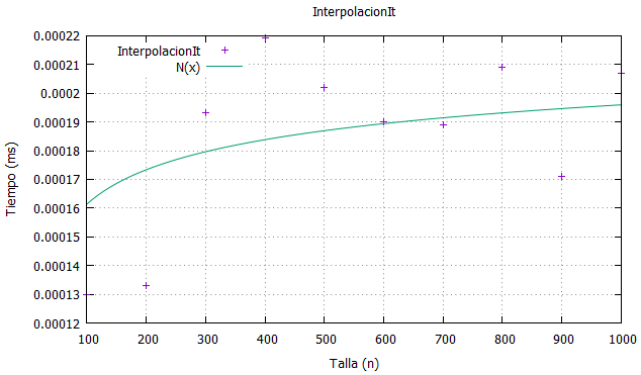
En este algoritmo obtenemos una gráfica creciente de forma lineal. Esto es lo esperado ya que a mayor tamaño de problema mayor tiempo de ejecución. Hay puntos que no coinciden con la recta. Esto se debe a que al tratarse de una medida empírica y en tiempo real, el ordenador puede tardar más o menos en el proceso (por programas que tengamos abiertos por ejemplo). Observaremos que esto pasa en el resto de algoritmos.

```
*** Busqueda por BinariaIt ***

Tiempos de ejecucion promedio

        Talla           Tiempo (mseg)

        100                0.00019
        200                0.00022
        300                0.00025
        400                0.00023
        500                 0.0002
        600                0.00022
        700                0.00022
        800                0.00022
        900                0.00022
        1000               0.00022

Datos guardados en el fichero BinariaIt.dat
```



Obtenemos una curva creciente logarítmica. A medida que la talla de la tabla es mayor. La mayoría de puntos no coinciden con la curva, como ya hemos dicho, al tratarse de un estudio empírico los resultados no serán exactos por otros programas y condiciones que tenga el ordenador en el momento de ejecución de cada talla, a pesar de haber hecho un bucle for para que el valor sea lo más aproximado posible.

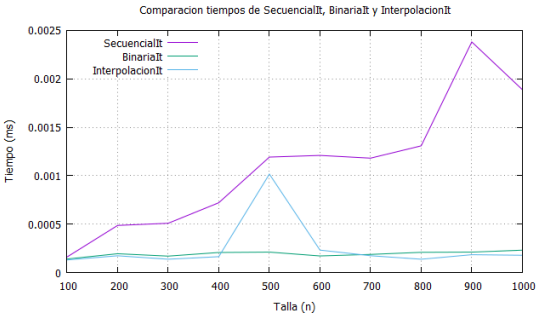
```
*** Busqueda por InterpolacionIt ***

Tiempos de ejecucion promedio

        Talla           Tiempo (mseg)

        100                0.00013
        200                0.00013
        300                0.00019
        400                0.00022
        500                 0.0002
        600                0.00019
        700                0.00019
        800                0.00021
        900                0.00017
        1000               0.00021

Datos guardados en el fichero InterpolacionIt.dat
```



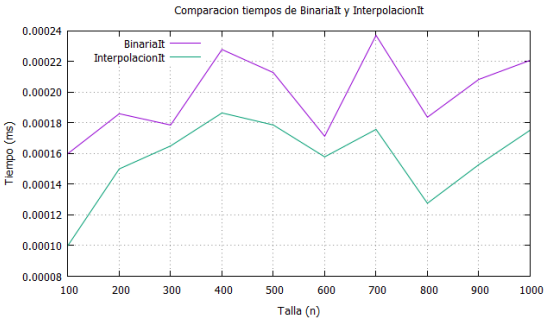
En este algoritmo pasa lo mismo que en el anterior, también es logarítmico pero de orden menor al ser logaritmo de un logaritmo.

5.2. Conclusiones

Talla	Tiempo (mseg)		
	SecuencialIt	BinariaIt	InterpolacionIt
100	0.00016	0.00014	0.00013
200	0.00049	0.00019	0.00017
300	0.00051	0.00017	0.00014
400	0.00072	0.00021	0.00016
500	0.0012	0.00021	0.001
600	0.0012	0.00017	0.00023
700	0.0012	0.00019	0.00017
800	0.0013	0.00021	0.00014
900	0.0024	0.00021	0.00018
1000	0.0019	0.00023	0.00018



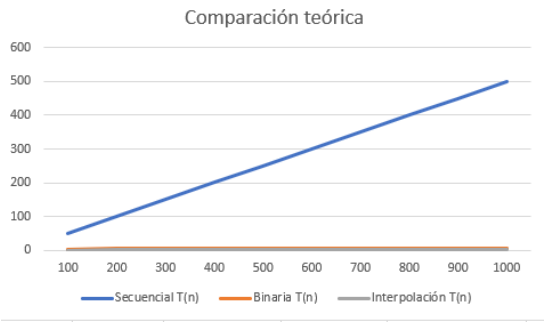
Para sacar conclusiones compararemos los datos y gráficas que comparan 3 algoritmos. Esto está implementado en el método comparar todos. Buscan en el mismo vector la misma key variando la talla.

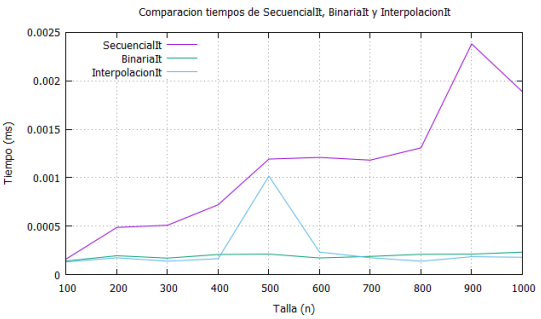


Entre estos dos métodos, podemos observar rápidamente que el más eficiente es el de interpolación iterativa aunque queda muy cerca de binaria. Tanto para vectores de una talla pequeña como para mayores tiene un tiempo de ejecución menor que el resto.

Así obtenemos que ordenados por eficiencia. El más eficiente es interpolación, luego binaria y el menos eficiente es secuencial. Aunque como ya se ha mencionado, la eficiencia de interpolación dependerá del crecimiento de los elementos.

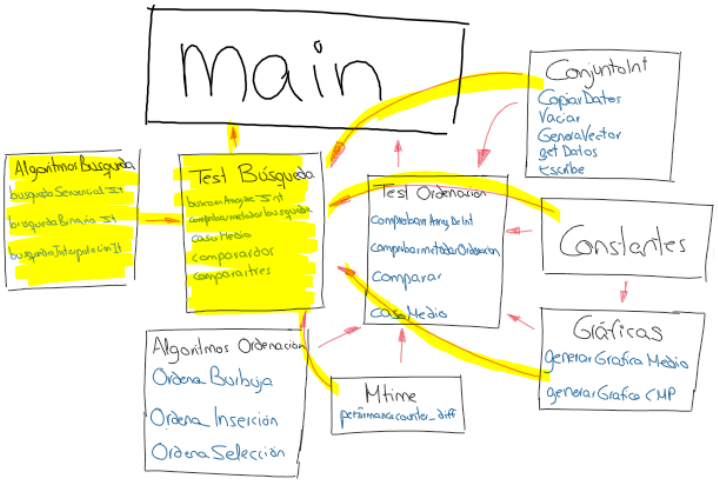
6. Comparación de los resultados teórico y experimental.





Podemos observar que la escala de la teórica es mucho más grande que la empírica. Esto se debe a que la teórica está medida en OE mientras la empírica en s. Por otro lado tenemos los picos, la gráfica teórica es mucho más suave al ser datos que siguen una fórmula mientras que en la empírica nos encontramos una gráfica con picos. Teniendo en cuenta que esto se debe al tiempo de ejecución de cada ordenador. Los resultados son coherentes y coinciden, secuencial es el menos eficiente, interpolación y binaria son casi iguales pero si analizamos más profundamente, interpolación podría ser más eficiente.

7. Diseño de la aplicación.



El programa se compone de lo implementado en la práctica anterior, a lo que se le añade TestBusqueda y AlgoritmosBusqueda.

La clase AlgoritmosBusqueda contiene los algoritmos de búsqueda que hemos estudiado. Esto se ha implementado a partir de los pseudocódigos, que se han cambiado a lenguaje de c++.

La clase TestBúsqueda es la que contiene las funciones principales del programa que se llaman en el main.

El método comprobar métodos de búsqueda genera un vector y lo ordenado por inserción al ser el más eficiente estudiado en la anterior práctica, y busca la clave introducida. Si ejecutamos el programa nosotros podremos introducir el tamaño del vector y la key. Así podemos observar el funcionamiento de cada algoritmo.

El main está compuesto de un menú principal que nos permite acceder a dos submenús: el de métodos de ordenación que es el implementado en la práctica 2 y el implementado en esta práctica: algoritmos de búsqueda. En ambos menús nos d las mismas opciones, pero en este nuevo también podremos comparar los tres algoritmos a la vez.

8. Conclusiones y valoraciones personales de la práctica.

Esta práctica nos ha permitido comprobar, comparar y aprender como funcionan los algoritmos de búsqueda y las diferencias en la eficiencia. También la importancia de elaborar un proyecto por partes y aprovechar lo realizado con anterioridad.

Aún así la mayor dificultad que he encontrado ha sido analizar la eficiencia de interpolación ya que no ha sido estudiado en clase y he tenido que investigar para comprenderlo mejor.

Ha sido una práctica interesante. Al basarse en prácticas anteriores ahorra tiempo de ejecución y hace necesario comprender lo realizado con anterioridad para seguir prograsando en las prácticas.

Esta práctica vuelve a dejar constancia de la necesidad de analizar la eficiencia de algoritmos antes de implementarlos.