PRÁCTICA 3: ANÁLISIS EXPERIMENTAL DE ALGORITMOS DE BÚSQUEDA

Fundamentos de Análisis de Algoritmos

Adrián Moreno Monterde

Índice

1. INTRODUCCIÓN. ALGORITMOS DE BÚSQUEDA

En esta práctica vamos a estudiar de forma experimental el comportamiento de varios algoritmos de búsqueda comparándolos entre sí: Secuencial, Binaria e Interpolación. Al finalizar, utilizaremos los tiempos para ordenar un vector usando distintos algoritmos. Así tendremos criterios objetivos que nos ayudarán a poder elegir el algoritmo más eficiente.

2. CÁLCULO DE LOS TIEMPOS TEÓRICOS

2.1 PSEUDOCÓDIGOS Y ANÁLISIS DE COSTE

Algoritmo Secuencial Iterativo

```
indice BusquedaLineal (tabla T, indice P, indice U, clave K) {
    i = P;
    mientras (i <= U) AND (T[i] !=K ) hacer
        i++;
    fmientras
    si K == T[i] entonces
        return i;
    else
        return -1;
    fsi
}
```

Fila 1: 1 asignación

Fila 2: Condición bucle: 1 op. lógica, 1 acceso vector y 2 comparaciones

Fila 3: 1 asignación y 1 op.aritmética

Fila 4: Condicional, 1 comparación y 1 acceso vector

Algoritmo Binario Iterativo

```
funcion busquedaBinariaIt(V[1..n];size,clave:int):entero

   encontrado: boolean;
   mitad,primero,ultimo:int;
   primero ← 1
   ultimo ← size
   encontrado ← falso;
   mientras (primero <= ultimo && !encontrado) hacer
      mitad ← ((primero+ ultimo) / 2);
      si clave = V[mitad] entonces
         encontrado ← cierto;
      sino
         si clave < V[mitad] entonces
            ultimo   mitad-1;
         sino
            si clave > V[mitad] entonces
               primero   mitad+1;
            fsi
         fsi
      fsi
   fmientras
   si encontrado
      devolver mitad; // posición donde se encuentra el elemento en el array
   sino
      devolver -1;   // no se encuentra el elemento en el array
   fsi
ffuncion busquedaBinariaIt
```

Fila 1: 1 asignación

Fila 2: 1 asignación

Fila 3: 1 asignación

Fila 4: Condición buble: 2 comparaciones y 1 op. lógica

Fila 5: 1 asignación, 2 op. aritméticas

Fila 6: Bucle si: 1 acceso vector, 1 asignación

Fila7: 1 asignación

Fila 8: Bucle si: 1 comparación, 1 acceso vector

Fila 9: 1 op aritmética, 1 asignación

Fila 10: Bucle si: 1 acceso vector, 1 comparación

Fila 11: 1 asignación, 1 op. artimética

Fila 12: Bucle si: 1 comparación

Fila 13: 1 op.

Fila 14: 1 op

Algoritmo Interpolación Iterativo

```
int funcion busquedaInterpolacionIt (V[1..n];size,clave:int)
  p,primero,ultimo:int;
1 primero ←1
2 ultimo ← size
3 mientras((v[ultimo]>= clave) AND (v[primero] < clave)) hacer
4   p ←primero + ((ultimo - primero) * (clave-v[primero])/(v[ultimo]-v[primero]))
5   si (clave > v[p]) entonces
6       primero ←p + 1
    sino
7       si (clave < v[p]) entonces
8           ultimo ←p - 1
        sino
9           primero ←p
        fsi
    fsi
  fmientras
10 si (v[primero] == clave) entonces
     devolver primero    // posición donde se encuentra el elemento en el array
  sino
11   devolver -1         // no se encuentra el elemento en el array
  fsi
ffuncion busquedaInterpolacionIt
```

Fila 1: 1 asignación

Fila 2: 1 asignación

Fila 3: Bucle mientras: 2 acceso vector, 2 comparaciones, 1 op. lógica

Fila 4: 1 asignación, 6 op. aritméticas, 3 acceso vector

Fila 5: Bucle si: 1 comparación, 1 acceso vector

Fila 6: 1 asignación, 1 op. aritmética

Fila 7: Bucle si: 1 comparación, 1 acceso vector

Fila 8: 1 asignación, 1 op. aritmética

Fila 9: 1 asignación

Fila 10: Bucle si: 1 acceso vector, 1 comparación

Fila 11: 1 op

Fila 12: 1 op

Algoritmo Secuencial Iterativo

$$Caso\ mejor: T(n) = 5 \rightarrow T(n) \in O(1)$$
$$Caso\ peor: T(n) = 7 + 6n \rightarrow T(n) \in O(n)$$
$$Caso\ medio\ T(n) = \frac{7 + 6n + 5}{2} = 6 + 3n \rightarrow T(n) \in O(n)$$

Algoritmo Binario Iterativo

$$Caso\ mejor: T(n) = 8 \rightarrow T(n) \in O(1)$$
$$Caso\ peor: T(n) = T\left(\frac{n}{2}\right) + 10 \quad Cambio\ de\ variable\ n = 2^k$$
$$\mathrm{T(n)} = \mathrm{T}(2^{k-1}) + 10 = \mathrm{T}(2^{k-2}) + 10 + 10 \ldots = \sum_{i=0}^{k-1} 10 = 10 \cdot \mathrm{k}$$

Desaciendo el cambio de viariable $\mathrm{k} = log_2 \mathrm{n}$

$$\mathrm{T(n)} = 10 \cdot log_2 \mathrm{n} \rightarrow T(n) \in O(\log n)$$
$$Caso\ medio: T(n) = \frac{8 + 10log_2 \mathrm{n}}{2} = 4 + 5log_2 \mathrm{n} \rightarrow T(n) \in O(\log n)$$

Algoritmo Interpolación Iterativo

Caso mejor $O(log(log(n)))$

Caso peor $O(n)$

Caso medio $O(log(log(n)))$

CONCLUSIONES

Acorde a los análisis teóricos podemos decir que la búsqueda por interpolación sería el algoritmo más rápido y la búsqueda secuencial el más lento.

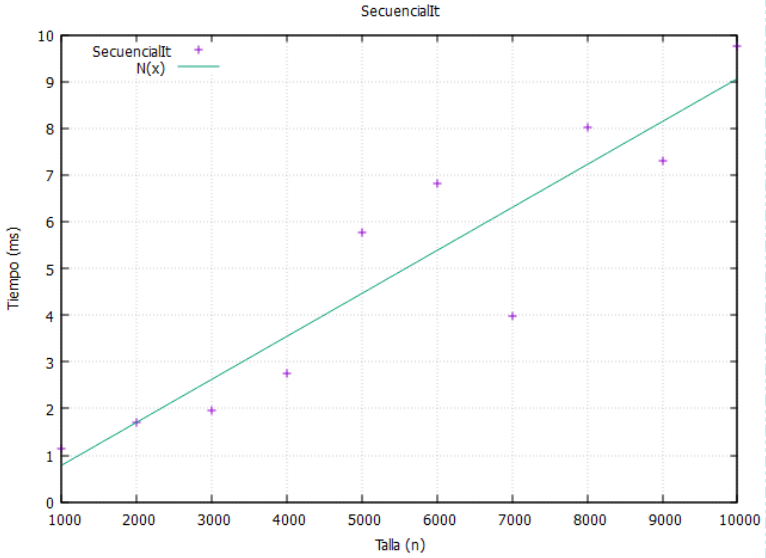
3. CÁLCULO DEL TIEMPO EXPERIMENTAL

3.1 TABLAS (FICHEROS)

Algoritmo Secuencial Iterativo

Busqueda SecuencialIt. Tiempos de ejecucion promedio

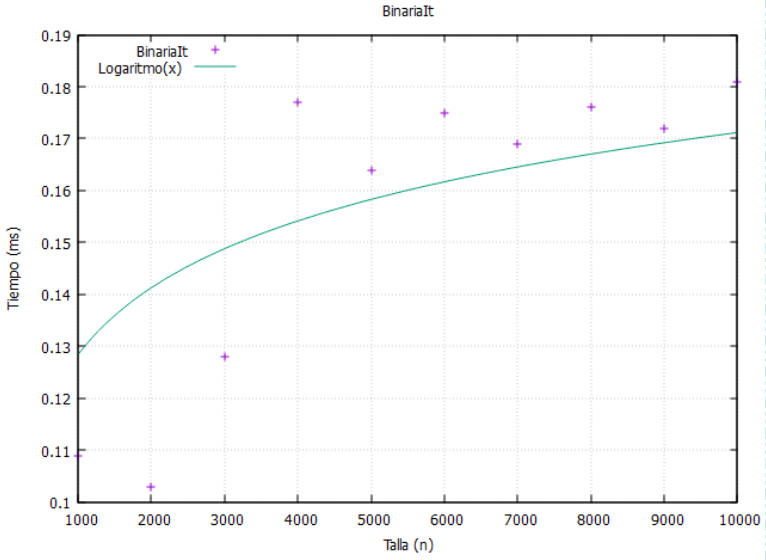
Talla	Tiempo (microseg)
1000	1.1330
2000	1.7100
3000	1.9700
4000	2.7470
5000	5.7580
6000	6.8300
7000	3.9760
8000	8.0280
9000	7.3080
10000	9.7520



Algoritmo Binario Iterativo

Busqueda BinariaIt. Tiempos de ejecucion promedio

Talla	Tiempo (microseg)
1000	0.1090
2000	0.1030
3000	0.1280
4000	0.1770
5000	0.1640
6000	0.1750
7000	0.1690
8000	0.1760
9000	0.1720
10000	0.1810

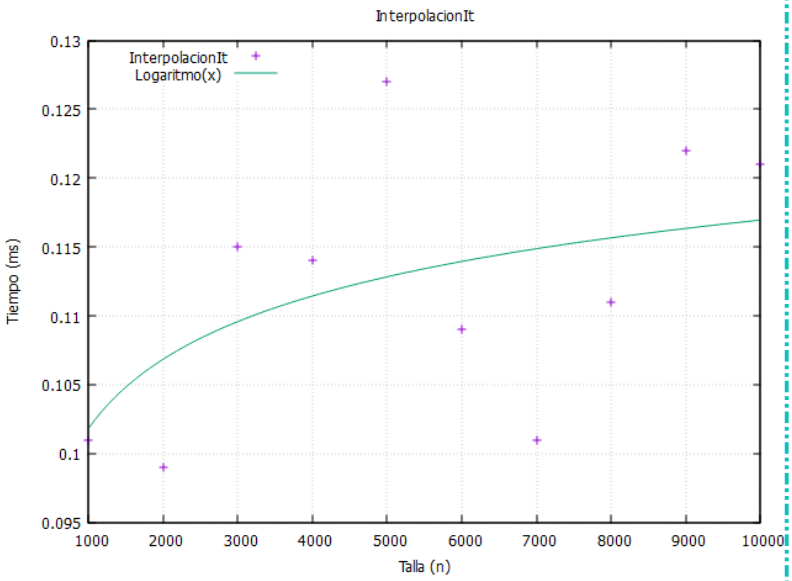


Algoritmo Interpolación Iterativo

```
Busqueda InterpolacionIt. Tiempos de ejecucion promedio

      Talla          Tiempo (microseg)

      1000               0.1010
      2000               0.0990
      3000               0.1150
      4000               0.1140
      5000               0.1270
      6000               0.1090
      7000               0.1010
      8000               0.1110
      9000               0.1220
      10000              0.1210
```



3.2 CONCLUSIONES

Para concluir, podemos observar que el algoritmo más eficiente es la búsqueda por interpolación, tal como se comentó en el apartado de análisis teórico anterior. Muy similar pero peor encontramos la búsqueda binaria y con peores resultados, claramente notable en los tiempos y gráfica, encontramos la búsqueda secuencial iterativa.

4. COMPARACIÓN DE LOS RESULTADOS TEÓRICO Y EXPERIMENTAL

Sabiendo que el orden de el algoritmo secuencial es de orden n, el de búsqueda binaria de orden log(n) y el de interpolación es de log(log(n)) podemos concluir con que el más rápido es el de interpolación seguido del binario y por último secuencial.

También es notoria la dificultad de implementación de estos dos métodos de ordenación logarítmicos, aunque en cuanto en eficiencia resulta totalmente rentable implementarlos.

5. DISEÑO DE LA APLICACIÓN

El proyecto se basa en el diseño modular y la programación enfocada a objetos, de esta forma estará dividido en varias partes que conformarán las distintas clases con sus respectivos métodos que se utilizarán más adelante. En el programa contamos

con un menú principal que nos da a elegir entre dos submenús (búsqueda y ordenación) según el número que pulsemos (1-2). En esta práctica nos centraremos en el menú de búsqueda ya que lo contenido en el menú de ordenación está reflejado en la práctica 2 de esta asignatura. El submenú de búsqueda nos da a elegir según lo que pulsemos (1-4) las distintas acciones a realizar.

-1. Probar los métodos de búsqueda.

-2. Obtener el caso medio de un método de búsqueda.

-3. Comparar dos métodos

-4. Comparar todos los métodos.

Para poder realizar todo esto, contamos con varios archivos .h y sus adjuntos .cpp:

-AlgoritmosOrdenacion.h: Aquí se declara la clase AlgoritmosOrdenacion, cuyos métodos son:

Void ordenaBurbuja: Método Burbuja

Void ordenaInsercion: Método Inserción

Void ordenaSeleccion: Método Selección

-AlgoritmosOrdenacion.cpp: Aquí se procede a programar los métodos anteriormente declarados en el .h.

-ConjuntoInt.h: Aquí se declara la clase ConjuntoInt, cuyos métodos son:

Void vaciar: Establece la talla del vector a 0.

Void GeneraVector: Genera un vector formado por números aleatorios entre 1 y 999 gracias al comando rand()%1000

Int* getDatos: Puntero que apunta al vector datos

Void escribe: Muestra por pantalla los datos del vector datos.

Void Clonar: Clona el contenido de un vector en otro

Int generaKey: genera una clave aleatoria entre los rangos de número que comprende el vector

-ConjuntoInt.cpp: Aquí se procede a programar los métodos anteriormente declarados en el .h.

-Graficas.h: Aquí se declara la clase Graficas, cuyos métodos son:

Void generarGraficaMEDIO

Void generarGraficaCMP

Void generarGraficaCMPtodos

-Graficas.cpp: Aquí se proceder a programar las gráficas

-Mtime.h y Mtime.cpp: Se trata de una clase cuyo método simulará una especie de contador que nos ayudará a calcular el tiempo de ejecución de cada método de ordenación.

-Constantes.h: Se declaran varias constantes que se utilizarán posteriormente en distintos apartados del programa. Un ejemplo es INCTALLA que indica la variación de las tallas de un vector.

-TestOrdenacion.cpp: Recibe por parámetro el vector, el tamaño y el método de ordenación seleccionado. A continuación, según el método seleccionado, llama a los distintos métodos para realizar la ordenación. Luego muestra la tabla con los tiempos de ejecución promedios.

-TestBusqueda.h: Comprueba que los métodos de búsqueda de la clase AlgoritmosBusqueda funcionan adecuadamente. Calcula la eficiencia para el caso medio de un método de búsqueda, permitiendo guardar los datos e imprimir la gráfica correspondiente con ajuste al Orden de complejidad. Compara el coste temporal de dos métodos de búsqueda permitiendo guardar los datos e imprimir la gráfica correspondiente. Por último, también compara todos los algoritmos de búsqueda permitiendo guardar los datos e imprimar la gráfica comparativa correspondiente. Los métodos de esta clase son:

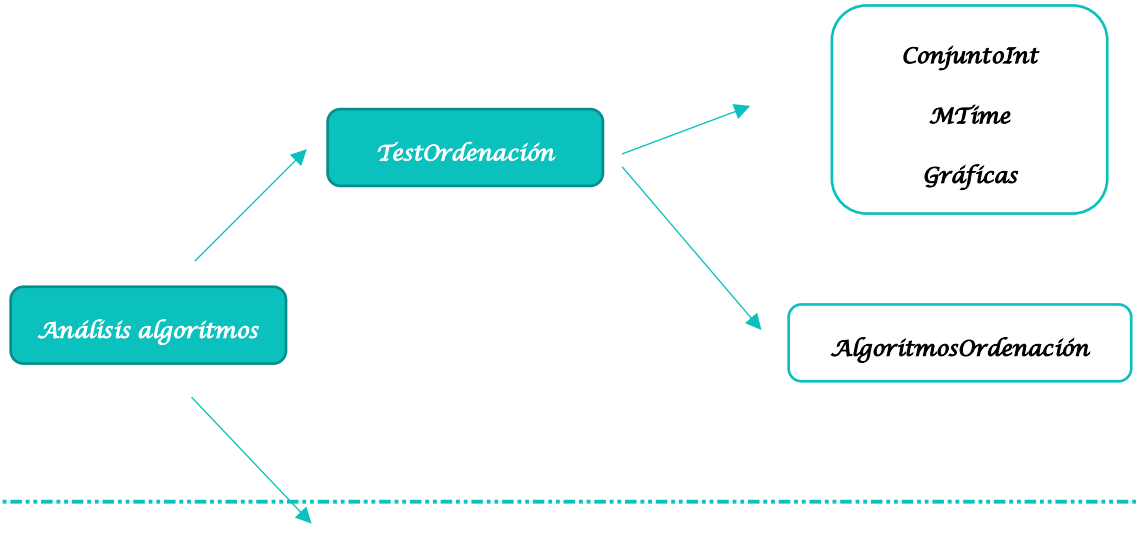
-Static Double buscaEnArrayDeInt: Busca un elemento en un array de enteros según el método indicado pasando por parámetro ambos y finalmente devuelve la posición del elemento en el vector (-1 si no existe) y el tiempo empleado en la búsqueda.

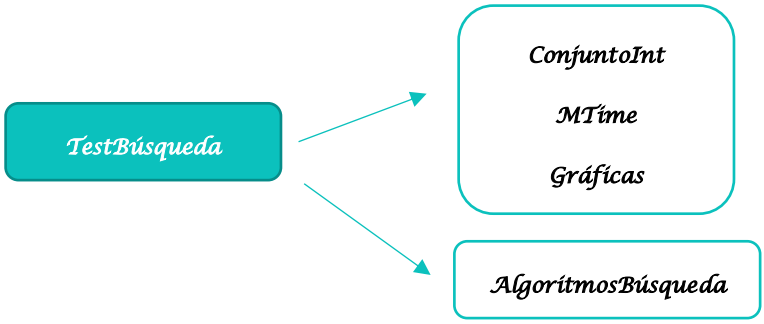
-Void comprobarMetodosBusqueda: Comprueba que los métodos de búsqueda funcionan correctamente.

-Void casoMedio: Calcula el caso medio de un método de búsqueda, permite crear el fichero de datos y la gráfica correspondiente.

-Void comparar: Compara dos métodos de búsqueda, permite crear el fichero de datos y la gráfica correspondiente.

-Void compararTodos: Compara todos los métodos de búsqueda (secuencial, binaria e interpolación), permite crear el fichero de datos y la gráfica correspondiente.





6. CONCLUSIONES Y VALORACIONES PERSONALES DE LA PRÁCTICA

Esta práctica nos ha servido para estudiar los algoritmos tanto de ordenación como de búsqueda, además hemos podido poner en práctica los conocimientos adquiridos en las sesiones de teoría. Ha sido una ampliación de la práctica 2 y así hemos podido perfeccionarla y completarla haciéndola más útil.

Personalmente, creo que el uso de las gráficas nos ayuda mucho más a entender los resultados obtenidos y a comparar cada método y procedimiento, así podemos verlo de una manera más dinámica.

Por otro lado, también pienso que, a la vez que nos ayuda en algunas cosas la práctica, en otras no mucho. Creo que más que estudiar eficiencia de algoritmos o compararlas, estamos mejorando en programar y en hacer código.