PRÁCTICA 2: ANÁLISIS EXPERIMENTAL DE ALGORITMOS DE ORDENACIÓN

Fundamentos de Análisis de Algoritmos

Adrián Moreno Monterde

Índice

1. INTRODUCCIÓN. ALGORITMOS DE ORDENACIÓN

En esta práctica vamos a estudiar de forma experimental el comportamiento de un algoritmo de ordenación comparándolo con los procedimientos de ordenación Burbuja, Inserción y Selección. Al finalizar, utilizaremos los tiempos para ordenar un vector usando distintos procedimientos. Así tendremos criterios objetivos que nos ayudarán a poder elegir el procedimiento más eficiente.

2. CÁLCULO DE LOS TIEMPOS TEÓRICOS

2.1 PSEUDOCÓDIGOS Y ANÁLISIS DE COSTE

Método burbuja

```
procedimiento burbuja(T[1..n])
   para i← n-1 hasta 1 hacer
      para j ←1 hasta i hacer
            si T[j] >T[j+1] entonces
               auxiliar ← T[j]
               T[j] ← T[j+1]
               T[j+1] ← auxiliar
            fsi
      fpara
   fpara
fprocedimiento
```

Método inserción

```
procedimiento inserción(T[1..n])
   para i← 2 hasta n hacer
      x ← T[i]
      j ← i-1
      mientras j>0 AND x < T[j] hacer
         T[j+1] ← T[j]
         j ← j-1
      fmientras
      T[j+1] ← x
   fpara
fprocedimiento
```

Método selección

```
procedimiento selección(T[1..n])
   para i← 1 hasta n-1 hacer
      posminimo ← i
      para j ← i +1 hasta n hacer
            si T[j] < T[posminimo] entonces
              posminimo ← j
            fsi
      fpara
      auxiliar ← T[posminimo]
      T[posminimo] ← T[i]
      T[i] ← auxiliar
   fpara
fprocedimiento
```

Burbuja

Fila 1: 1 comparación y 1 operación aritmética

Fila 2: 1 comparación y 1 operación aritmética

Fila 3: Condicional, 2 acceso vector,1 op. aritmética y 1 comparación.

Fila 4: 1 asignación y 1 acceso vector

Fila 5: 1 asignación, 2 acceso vector y 2 op. aritmética

Fila 6: 1 asignación, 1 acceso vector y 1 op. aritmética

$$Caso\ peor, mejor\ y\ medio:\ C(n) = \sum_{i=1}^{n-1}\sum_{j=1}^{i} 1 = \sum_{i=1}^{n-1} i = \frac{(n-1)(n-1+1)}{2} = \frac{n(n-1)}{2}$$

$$Caso\ peor: I(n) = \sum_{i=1}^{n-1}\sum_{j=1}^{i} 1 = \frac{n(n-1)}{2}\quad Caso\ mejor: I(n) = 0$$

$$Caso\ medio\ I(n) = \frac{n(n-1)}{4}$$

$$Caso\ medio\ T(n) = C(n) + 3 \cdot I(n) = \frac{5}{4}n(n-1)$$

Inserción

Fila 1: 1 comparación y 1 op. aritmética

Fila 2: 1 asignación y 1 acceso vector

Fila 3: 1 asignación y 1 operación aritmética

Fila 4: Cond bucle: 2 comparaciones, 1 acceso vector, 1 lógica.

Fila 5: 1 asignación, 2 acceso vector y 1 op. aritmética

Fila 6: 1 asignación y 1 operación aritmética

$$Caso\ peor:\ C(n) = \sum_{i=2}^{n-1}\sum_{j=1}^{i-1} 1 = \sum_{i=2}^{n} i - 1 = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

$$Caso\ mejor: C(n) = \sum_{i=2}^{n} 1 = n - 1$$

$$Caso\ medio: C(n) = \frac{\frac{n(n-1)}{2} + n - 1}{2} = \frac{(n-1)(n+2)}{4}$$

$$Caso\ peor, mejor, medio: I(n) = \sum_{i=1}^{n-1} 1 = n - 1$$

$$Caso\ medio\ T(n) = C(n) + 3 \cdot I(n) = (n-1)\frac{n+14}{4}$$

Selección

Fila 1: 1 comparación y 2 operaciones aritméticas

Fila 2: 1 asignación

Fila 3: 1 comparación y 1 operación aritmética

Fila 4: condicional, 1 comparación y 2 acceso vector

Fila 5: 1 asignación

Fila 6: 1 asignación y 1 acceso vector

Fila 7: 1 asignación y 1 acceso vector

Fila 8: 1 asignación y 1 acceso vector

$$T(n) \in O(n^2) \ \ Para\ todos\ los\ casos$$

$$Caso\ peor, mejor\ y\ medio: \ C(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1 = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

$$Caso\ peor, mejor, medio: I(n) = \sum_{i=1}^{n-1} 1 = n-1$$

$$Caso\ medio\ T(n) = C(n) + 3 \cdot I(n) = \frac{n(n-1)}{2} + 3(n-1)$$

2.2 TABLAS (FICHEROS) Y GRÁFICAS DE COSTE

Método burbuja

```
*** Ordenacion por Burbuja ***

Tiempo de ejecucion promedio


Talla           Tiempo (mseg)

100                0.0259
200                0.1015
300                0.1281
400                0.2561
500                0.3509
600                0.5828
700                0.6649
800                0.9445
900                1.1185
1000               1.6875
```

Método inserción

```
*** Ordenacion por Insercion ***

Tiempo de ejecucion promedio


Talla           Tiempo (mseg)

100                0.0054
200                0.0209
300                0.0426
400                0.0825
500                0.1200
600                0.1862
700                0.2376
800                0.2866
900                0.3539
1000               0.4628
```

Método selección

```
*** Ordenacion por Seleccion ***

Tiempo de ejecucion promedio


Talla           Tiempo (mseg)

100                0.0116
200                0.0416
300                0.0888
400                0.1557
500                0.2533
600                0.3515
700                0.4735
800                0.6150
900                0.7832
1000               0.9599
```

2.3 CONCLUSIONES

El coste temporal del algoritmo depende de la talla y de la instancia. El más eficiente de los tres es Inserción, seguido de Selección y por último Burbuja.

3. CÁLCULO DEL TIEMPO EXPERIMENTAL

3.1 TABLAS (FICHEROS) Y GRÁFICAS DE COSTE

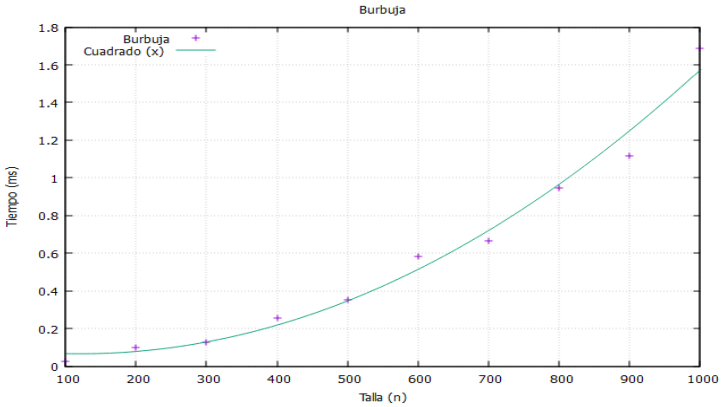
Método burbuja

```
*** Ordenacion por Burbuja ***

Tiempo de ejecucion promedio


Talla            Tiempo (mseg)

100                 0.0259
200                 0.1015
300                 0.1281
400                 0.2561
500                 0.3509
600                 0.5828
700                 0.6649
800                 0.9445
900                 1.1185
1000                1.6875
```



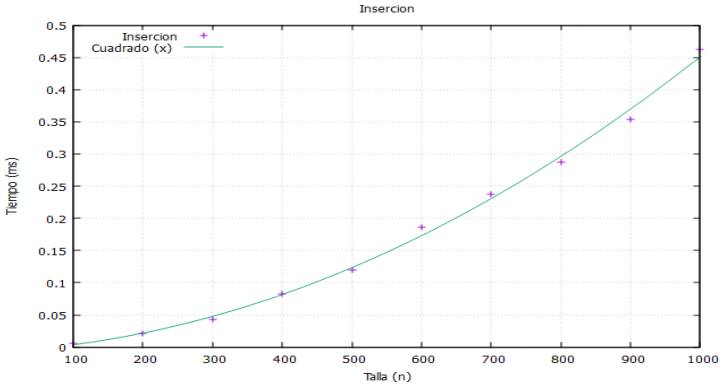
Método inserción

```
*** Ordenacion por Insercion ***

Tiempo de ejecucion promedio


Talla            Tiempo (mseg)

100                 0.0054
200                 0.0209
300                 0.0426
400                 0.0825
500                 0.1200
600                 0.1862
700                 0.2376
800                 0.2866
900                 0.3539
1000                0.4628
```



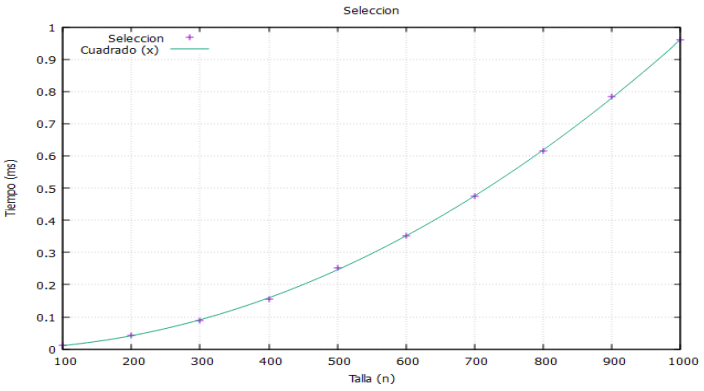
Método selección

```
*** Ordenacion por Seleccion ***

Tiempo de ejecucion promedio


Talla            Tiempo (mseg)

100                 0.0116
200                 0.0416
300                 0.0888
400                 0.1557
500                 0.2533
600                 0.3515
700                 0.4735
800                 0.6150
900                 0.7832
1000                0.9599
```

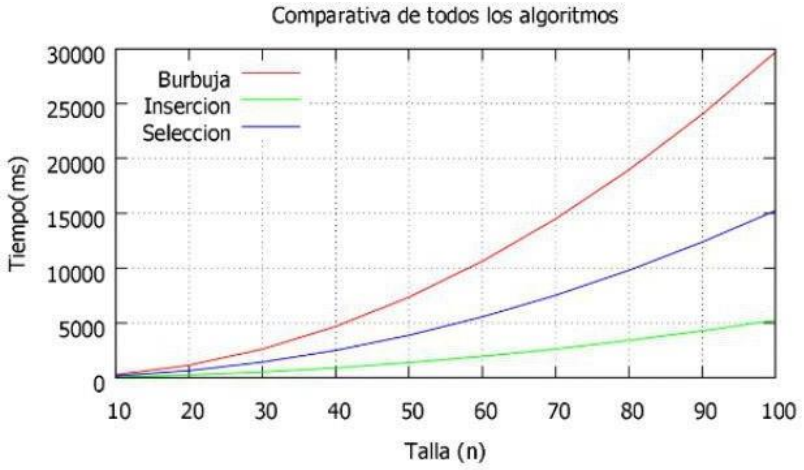


3.2 CONCLUSIONES

Podemos concluir que el método más eficiente es el método Inserción, ya que tarda mucho menos tiempo en ejecutarse que los métodos Selección o Burbuja (siendo este último el método que más tarda en ejecutarse y, por lo tanto, el menos eficiente).

4. COMPARACIÓN DE LOS RESULTADOS TEÓRICO Y EXPERIMENTAL

Comparativa de los algoritmos de forma teórica:

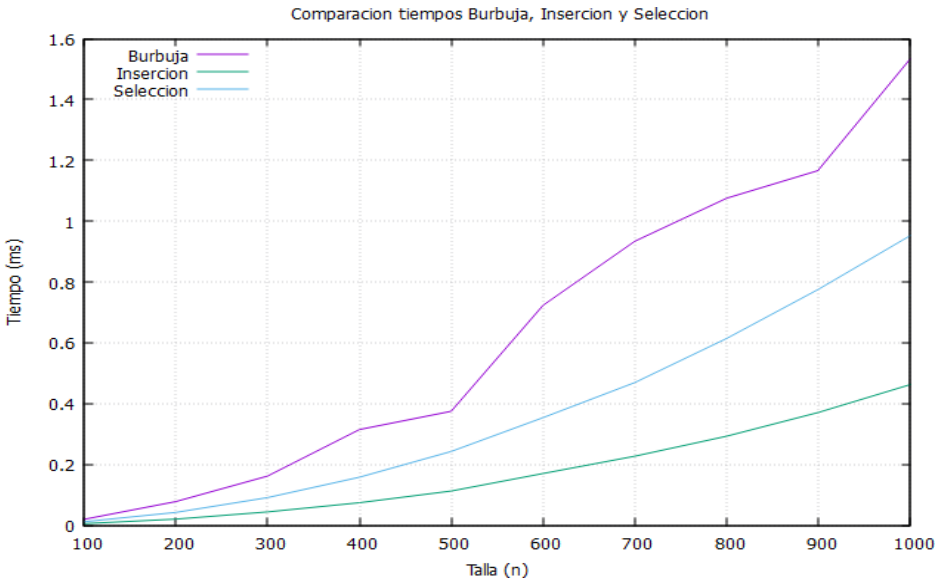


Comparativa de los algoritmos de forma empírica:

*** COMPARACION DE TODOS LOS METODS DE ORDENACION ***

Tiempos de ejecucion promedio

Talla	Burbuja Tiempo (mseg)	Insercion Tiempo (mseg)	Seleccion Tiempo (mseg)
100	0.0192	0.0052	0.0114
200	0.0777	0.0201	0.0422
300	0.1616	0.0436	0.0907
400	0.3148	0.0742	0.1579
500	0.3746	0.1126	0.2427
600	0.7234	0.1702	0.3540
700	0.9339	0.2272	0.4694
800	1.0756	0.2928	0.6145
900	1.1668	0.3712	0.7759
1000	1.5324	0.4628	0.9529



5. DISEÑO DE LA APLICACIÓN

El proyecto se basa en el diseño modular y la programación enfocada a objetos. En el programa contamos con un menú principal que nos da a elegir según el número que pulsemos (1-4) las distintas acciones a realizar.

1. Probar los métodos de ordenación.

2. Obtener el caso medio de un método de ordenación.

3. Comparar dos métodos.

4 Comparar todos los métodos

Contamos con varios archivos .h y sus adjuntos .cpp:

- AlgoritmosOrdenacion.h: se declara la clase AlgoritmosOrdenacion, cuyos métodos son: Void ordenaBurbuja, Void ordenaInsercion, Void ordenaSeleccion

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

- ConjuntoInt.h: 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 puntero del vector datos que se le pase por parámetro.

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

- Graficas.h: se declara la clase Graficas, cuyos métodos son: Void generarGraficaMEDIO , Void generarGrafica(String método1, String metodo2) y Void generarGrafica(vector<String> nombreAlgoritmo).

- Graficas.cpp: Aquí se proceder a programar las gráficas haciendo uso de ficheros con la terminación .gpl.

- Mtime.h y Mtime.cpp: 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.

-TestOrdenacion.h: Aquí se declara la clase TestOrdenacion, cuyos métodos son: double ordenarArrayDeInt, Void ComprobarMetodosOrdenacion , Void casoMedio, Void Comparar y void CompararTodos.

-TestOrdenacion.cpp: Recibe por parámetro el vector, el tamaño y el método de ordenación seleccionado. Según el método seleccionado, llama a los distintos métodos para realizar la ordenación.

6. CONCLUSIONES Y VALORACIONES PERSONALES DE LA PRÁCTICA

Esta práctica nos ha servido para estudiar de forma empírica los algoritmos de ordenación, además hemos podido poner en práctica los conocimientos adquiridos en las sesiones de teoría.

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.