

Práctica 2: análisis experimental de algoritmos de ordenación

Fundamentos de Análisis de Algoritmos



Adrián Moreno Monterde

Índice

1. Introducción. Algoritmos de ordenación

2. Cálculo de los tiempos teóricos:

2.1 Pseudocódigos y análisis de coste

2.2 Tablas (ficheros) y gráficas de coste

2.3 Conclusiones

3. Cálculo del tiempo experimental:

3.1 Tablas (ficheros) y gráficas de coste

3.2 Conclusiones

4. Comparación de los resultados teórico y experimental

5. Diseño de la aplicación

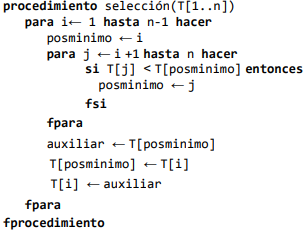
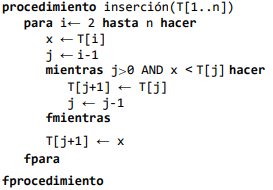
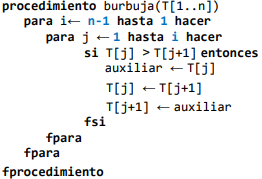
6. Conclusiones y valoraciones personales de la práctica

# 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 para ordenar un vector usando distintos procedimientos. Así tendremos así 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 Método inserción Método selección

Burbuja

Fila 1: 1 comparación y 1 operación aritmética

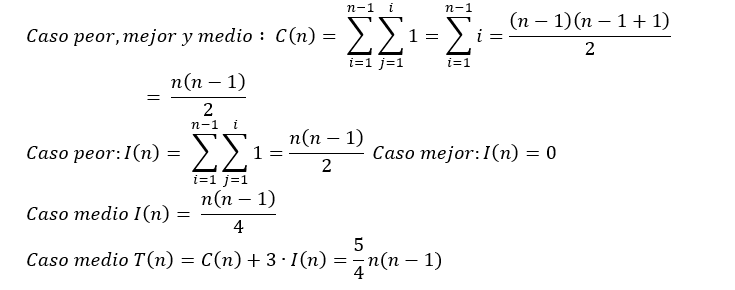
Fila 2: 1 comparación y 1 operación aritmética

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



Inserción

Fila 1: 1 comparación y 1 op. aritmética

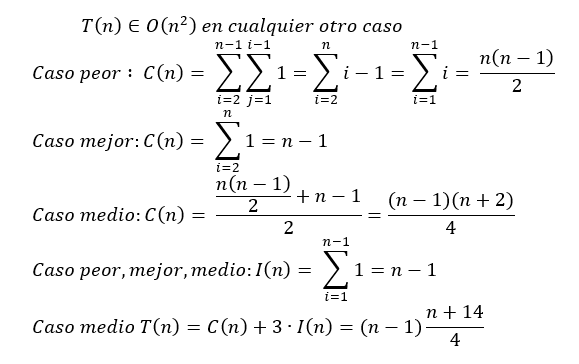
Fila 2: 1 asignación y 1 acceso vector

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



Selección

Fila 1: 1 comparación y 2 operaciones aritméticas

Fila 2: 1 asignación

Fila 3: 1 comparación y 1 operación aritmética

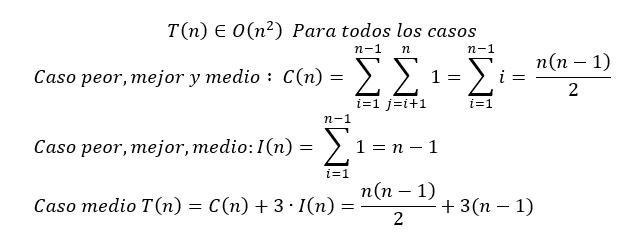
Fila 4: condicional, 1 comparación y 2 acceso vector

Fila 5: 1 asignación

Fila 6: 1 asignación y 1 acceso vector

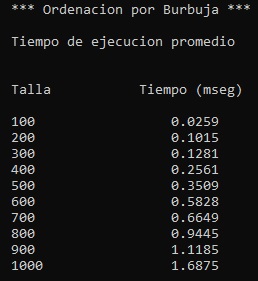
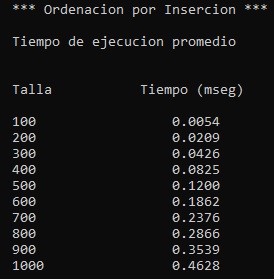
Fila 7: 1 asignación y 1 acceso vector

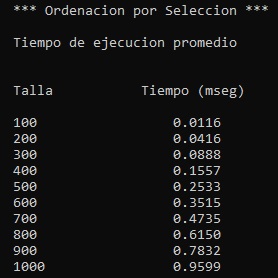
Fila 8: 1 asignación y 1 acceso vector



Ç

## 2.2 Tablas (ficheros) y gráficas de coste

 Método burbuja Método inserción Método selección

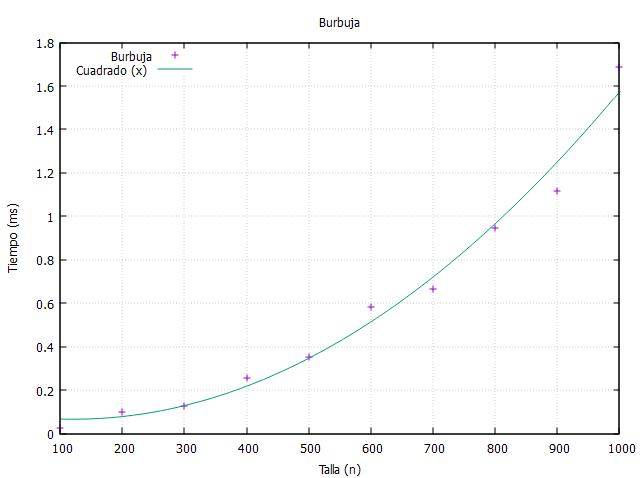


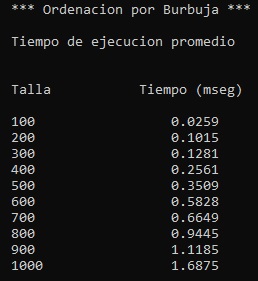
## 2.3 conclusiones

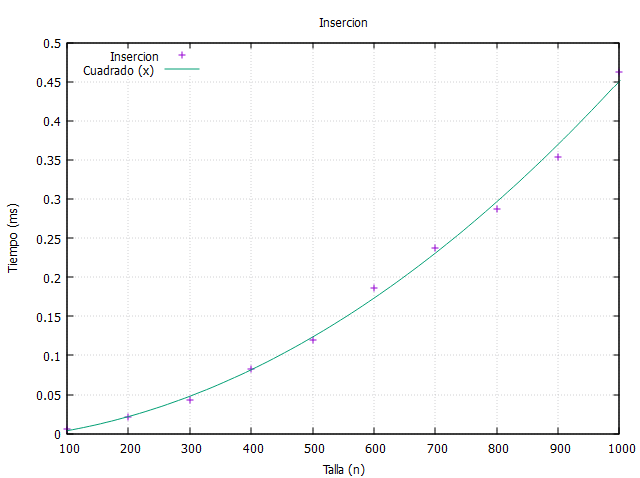
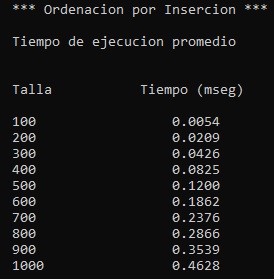
El coste temporal del algoritmo depende de la talla y de la instancia. El más eficiente de los tres es Inserción, seguido de Selección y por último Burbuja.

# 3. Cálculo del tiempo experimental

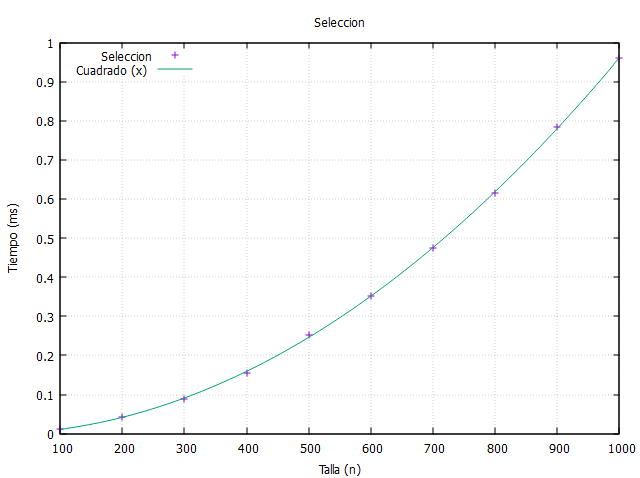
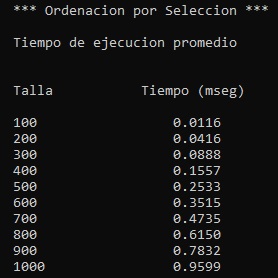
## 3.1 Tablas (ficheros) y gráficas de coste

Método burbuja



Método inserción

Método selección

Método selección

## 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).

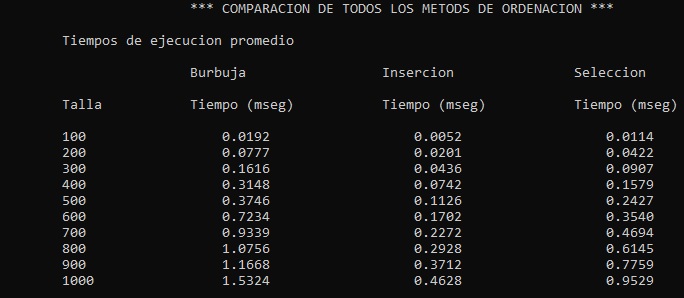
# 4. Comparación de los resultados teórico y experimental

Comparativa de los algoritmos de forma teórica:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Comparativa de los algoritmos de forma empírica:



Gráfico, Gráfico de líneas

Descripción generada automáticamente

# 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.