Análisis de la modificación del examén del grupo L2

FAA

Por: Ismael Da Palma Fernández

ÍNDICE

3. Algoritmo/s del examen. Cálculo del tiempo teórico

Se nos ha propuesto para la modificación de la práctica un pseudocódigo del algoritmo de ordenación burbuja bidireccional. Este algoritmo recibe como parámetros un array de enteros (v[]) y el tamaño de dicho array (size), al igual que el resto de los algoritmos de ordenación de la práctica 2.

3.1. Pseudocódigo y análisis de coste

```
burbujaBidireccional ( Vector a[1:n])

direccion ← frontal, comienzo ← 1, fin ← n-1, actual ← 1

repetir
    permut ← falso
    repetir
        si a[actual] > a[actual+1] entonces
            intercambiar a[actual] Y a[actual+1]
            permut ← verdadero
        fin si
        si (direccion=frontal) entonces
            actual ← actual+1
        si no
            actual ← actual-1
        fin si
    mientras que ((direccion=frontal) Y (actual<fin)) O ((direccion=final) Y (actual>comienzo))
    si (direccion=frontal) entonces
        direccion ← final
        fin ← fin-1
    si no
        direccion ← frontal
        comienzo ← comienzo+1
    fin si
mientras permut = verdadero
```

Este algoritmo (también llamado “Shaker Sort”) es una mejora del algoritmo de ordenación burbuja.

Por cada pasada que hace el algoritmo, este va cambiando de sentido, es decir, en vez de empezar a ordenar por el principio lo hace por el final y viceversa.

En cada pasada realiza las mismas comparaciones que en el algoritmo burbuja. Si el anterior es más grande que el siguiente se intercambian, sino no se intercambian.

Dependiendo de en qué sentido se encuentre, el elemento actual que se va a comparar irá aumentando(si empieza a comparar por el principio) o disminuyendo(si empieza comparando por el final).

Después de llegar al final o al principio del array se cambia su sentido y vuelve a realizar las comparaciones.

Eficiencia Teórica:

- Caso mejor: el vector ya se encuentra ordenado, por lo que el booleano permutar no se pondría a true y saldrá del bucle después de ver que permutar no es true. El algoritmo sólo efectúa n comparaciones. Por lo que el orden de complejidad es O(n).

- Caso peor y medio: en ambos casos el array se tiene que ordenar, por lo que el bucle se ejecuta más de una vez. Su orden de complejidad es $O(n^2)$.

3.2. Conclusiones:

El algoritmo de burbuja bidireccional es del mismo orden que el del resto de algoritmos de ordenación en su caso medio.

El burbuja bidireccional al tratarse de una mejora del algoritmo burbuja es más eficiente que este, aunque no más eficiente que los demás algoritmos. El orden de eficiencia, contando con los demás algoritmos de ordenación, es:

Burbuja << Burbuja Bidireccional << Selección << Inserción

4. Algoritmo/s del examen. Cálculo del tiempo experimental

Para el nuevo algoritmo se ha hecho lo mismo que para el resto de los algoritmos de ordenación, es decir, calcular su tiempo medio.

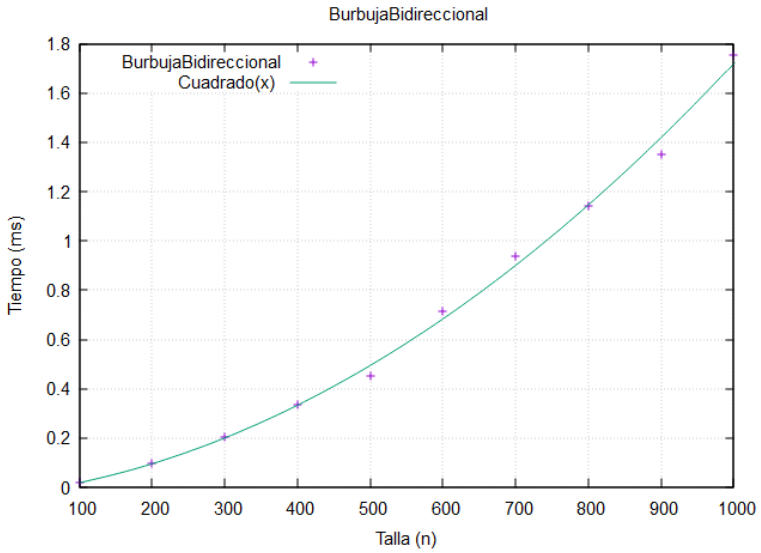
4.1. Tablas y Gráficas de coste

```
*** Ordenacion por BurbujaBidireccional ***

Tiempos de ejecucion promedio


Talla           Tiempo <mseg>

100                 0.018
200                 0.098
300                   0.2
400                  0.33
500                  0.45
600                  0.72
700                  0.94
800                   1.1
900                   1.4
1000                  1.8
```

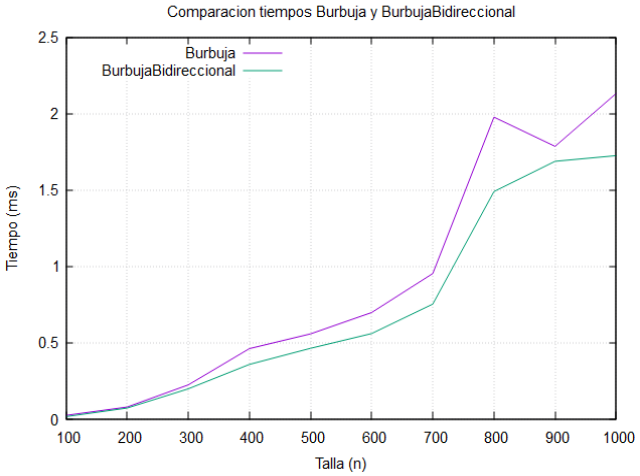


4.2. Conclusiones

Como podemos ver en la gráfica del punto anterior, los datos empíricos se ajustan bastante bien a la línea ideal teórica (línea verde) salvo en algunos casos de ejecución, que debido a la carga de trabajo del ordenador puede generar algunos picos bastante notables.

5. Comparación de los resultados teóricos y experimental

En este apartado voy a hablar de la comparación entre el burbuja y el burbuja bidireccional, ya que la comparación teórico experimental de los algoritmos de ordenación ya se realizó en la correspondiente memoria de la práctica 2.



Podemos ver en la gráfica que efectivamente el algoritmo de ordenación por burbuja bidireccional es más eficiente que el de burbuja.

Esto se debe a que el burbuja bidireccional no realiza siempre el cálculo por el principio, sino que va intercalando entre el principio y fin. Esto consigue que tanto los números pequeños como los grandes se desplacen a los extremos de la lista lo más rápido posible.

En tallas pequeñas, el burbuja y el burbuja bidireccional se comportan de forma similar.

6. Diseño de la aplicación

Como el diseño de la aplicación ya se ha realizado en la Memoria de la práctica 3, en este apartado voy a hablar de lo que se ha tenido que modificar en la práctica para poder añadir el nuevo algoritmo de ordenación. Las clases que se han modificado son:

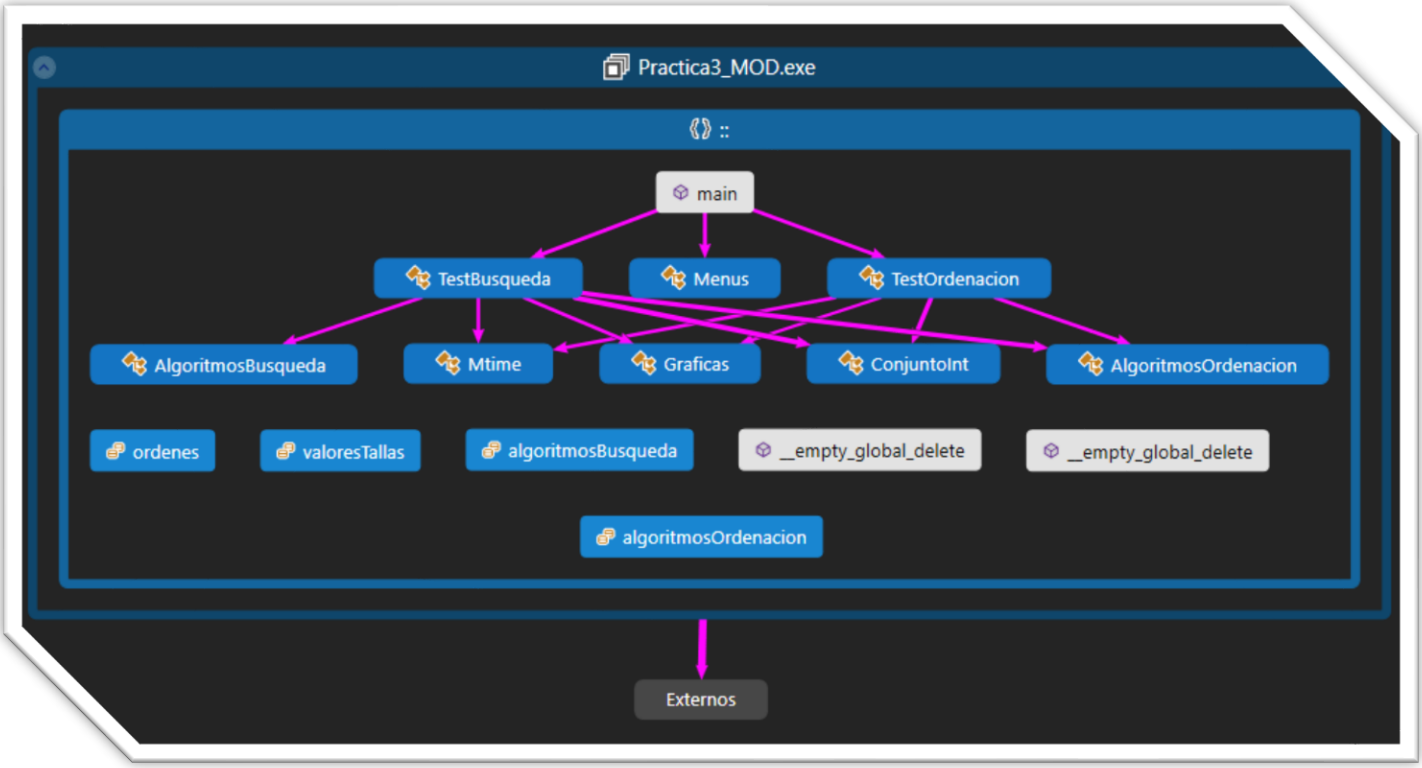
- AlgoritmosOrdenación: se ha añadido un nuevo método llamado ordenaBurbujaBidirecional que incluía el pseudocódigo que se mostró en la prueba pero en lenguaje C++.

- Menus: se añadió un nuevo cout en cada submenú de opciones del menú de ordenación con el correspondiente nombre del algoritmo.

- TestOrdenacion: se agregó en el constructor un nuevo nombre de algoritmo (“BurbujaBidireccional”), en el ordenaArrayDeInt se añadió un nuevo case para el burbuja bidireccional y en el método de casoMedio se añadió un nuevo case en la sección de mostrar la gráfica.

- Constantes.h: se agregó al final de las constantes simbólicas para los métodos de ordenación una nueva constante llamada BURBUJABIDIRECCIONAL.

A continuación mostramos la relación de clases de la práctica.



7. Conclusiones y valoraciones personales de la Práctica_Final

En esta última práctica/modificación hemos visto como implementar un nuevo algoritmo a la práctica a través de un pseudocódigo dado sin provocar errores a la hora de ejecutarlo y que nos permita comparar ese nuevo algoritmo con los ya implementados en la práctica y además comparar su eficiencia con el resto de los algoritmos.

8. Valoración personal de las prácticas de FAA

Las prácticas en general han estado muy bien, bastante completas, aunque había bastante código que era copiar y pegar, ha sido de utilidad para entender mejor el funcionamiento de los algoritmos de ordenación y de búsqueda y como se comportan estos con tallas grandes y pequeñas. También nos ha permitido ver las eficiencias que tiene cada algoritmo y cuál de ellos es más eficiente con respecto a los demás.