



**INSTITUTO POLITECNICO NACIONAL**  
**ESCUELA SUPERIOR DE COMPUTO**



## **Práctica 1: Algoritmos de ordenamiento iterativos**

Unidad de aprendizaje: Análisis y diseño de Algoritmos

Alumno: Colín Ramiro Joel

Profesora: Luz María Sánchez García

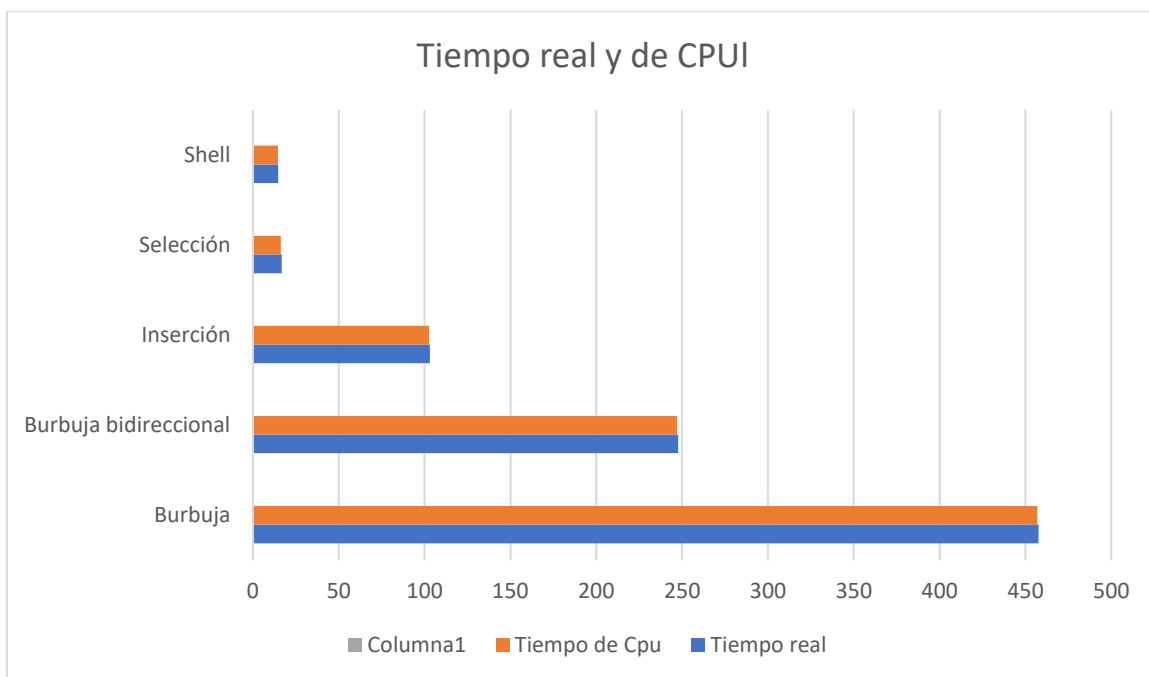
Fecha de entrega: 29/03/2021

## Planteamiento del problema

Se busca analizar los distintos métodos de ordenamiento, sus complejidades tanto temporal como espacial y observar que sucedería cuando dichos métodos tienen un número grande de entradas y plasmarlos en métodos gráficos para tener bien claro que tan complejos son estos algoritmos y cual de ellos es más conveniente usar, cuando se tienen grandes cantidades de entradas.

## Actividades

El entorno que utilicé para programar los algoritmos y realizar la práctica únicamente me permitía hacerlo con un arreglo de máximo 350,000 elementos. Se realizó la medición del tiempo real y de CPU de cada uno de los algoritmos de ordenamiento, los resultados, se expresan en la siguiente tabla. Los tiempos están dados en segundos.



Posterior a esta medición, se realizó un análisis temporal para cada algoritmo, cuando se ordenan los primeros 100, 1000, 5000, 10000, 50000, 100000, 200000 y debido a que mi equipo, únicamente me permite 350,000 elementos en el arreglo el análisis lo hice hasta ese dígito.

### **Burbuja**

$$T = 2n^2 + 7n + 6$$

### **Burbuja Bidireccional**

$$T = 2n^2 + 19n + 22$$

### **Inserción**

$$T = 2n^2 + 10n + 10$$

### **Selección**

$$T = 4n^2 + 5n + 2$$

### **Shell**

$$T = 2n^2 + 9n + 12$$

En el punto no.5 se realizó una aproximación polinomial de el análisis temporal y los resultados se colocaron en una tabla

### **Burbuja**

| k      | Instrucciones         |
|--------|-----------------------|
| 100    | 20706                 |
| 1000   | 2007006               |
| 5000   | 50035006              |
| 10000  | 200070006             |
| 50000  | 5000350006            |
| 100000 | $2 \times 10^{10}$    |
| 200000 | $8 \times 10^{10}$    |
| 300000 | $1.8 \times 10^{11}$  |
| 350000 | $2.45 \times 10^{11}$ |

### **Burbuja bidireccional**

| k      | Instrucciones         |
|--------|-----------------------|
| 100    | 21922                 |
| 1000   | 2019022               |
| 5000   | 50095022              |
| 10000  | 20019002              |
| 50000  | 5000950022            |
| 100000 | $2 \times 10^{10}$    |
| 200000 | $8 \times 10^{10}$    |
| 300000 | $1.8 \times 10^{11}$  |
| 350000 | $2.45 \times 10^{11}$ |

### Inserción

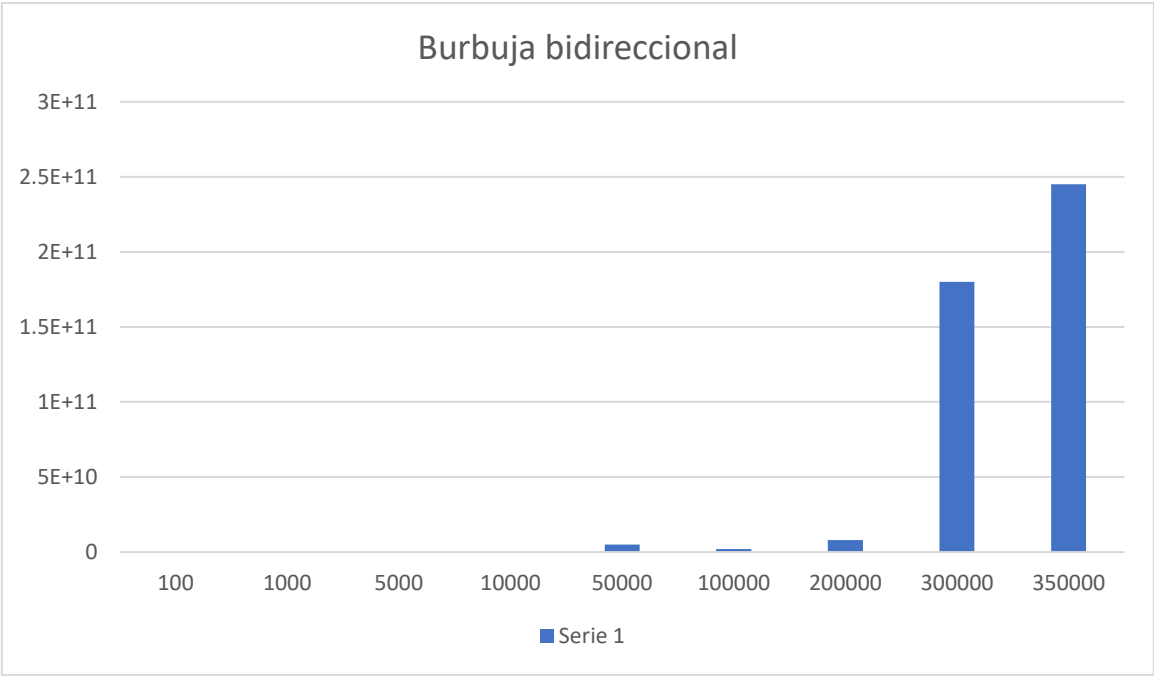
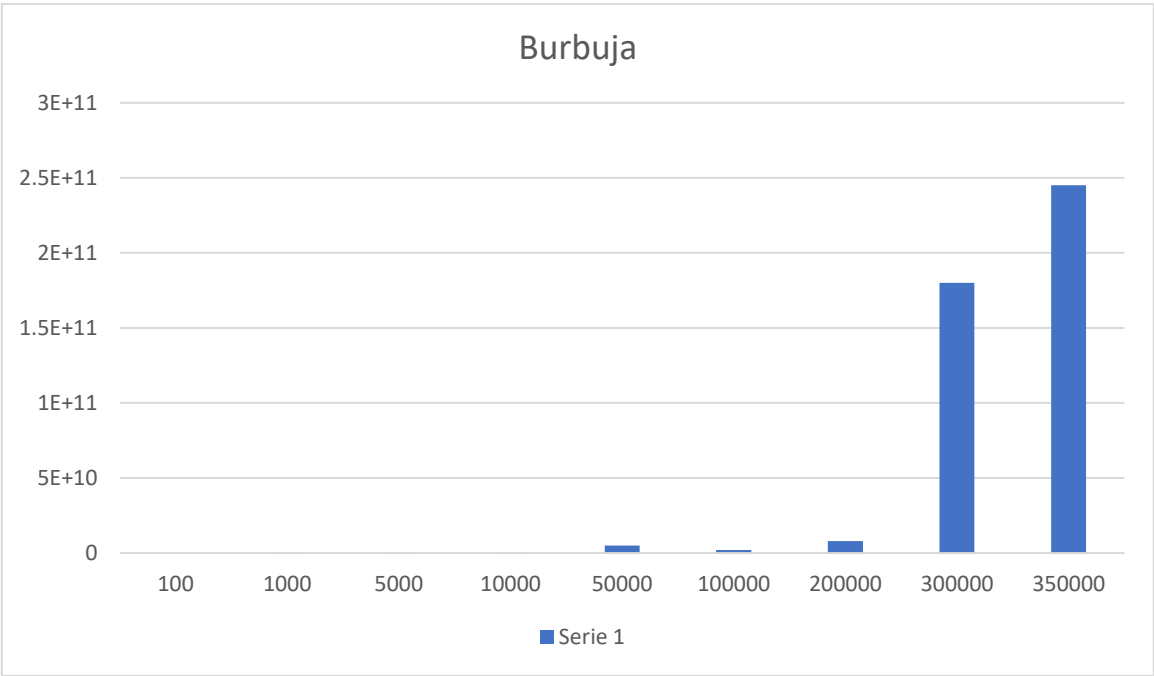
| k      | Instrucciones         |
|--------|-----------------------|
| 100    | 21010                 |
| 1000   | 2010010               |
| 5000   | 50050010              |
| 10000  | 200100010             |
| 50000  | 5000500010            |
| 100000 | $2 \times 10^{10}$    |
| 200000 | $8 \times 10^{10}$    |
| 300000 | $1.8 \times 10^{11}$  |
| 350000 | $2.45 \times 10^{11}$ |

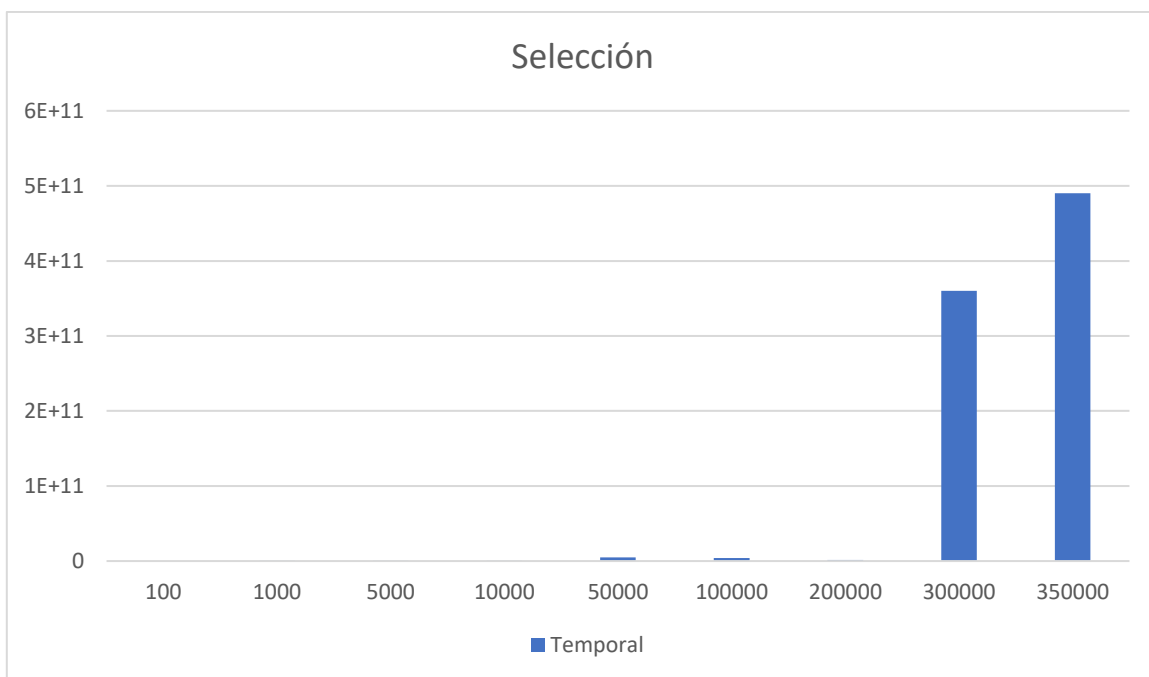
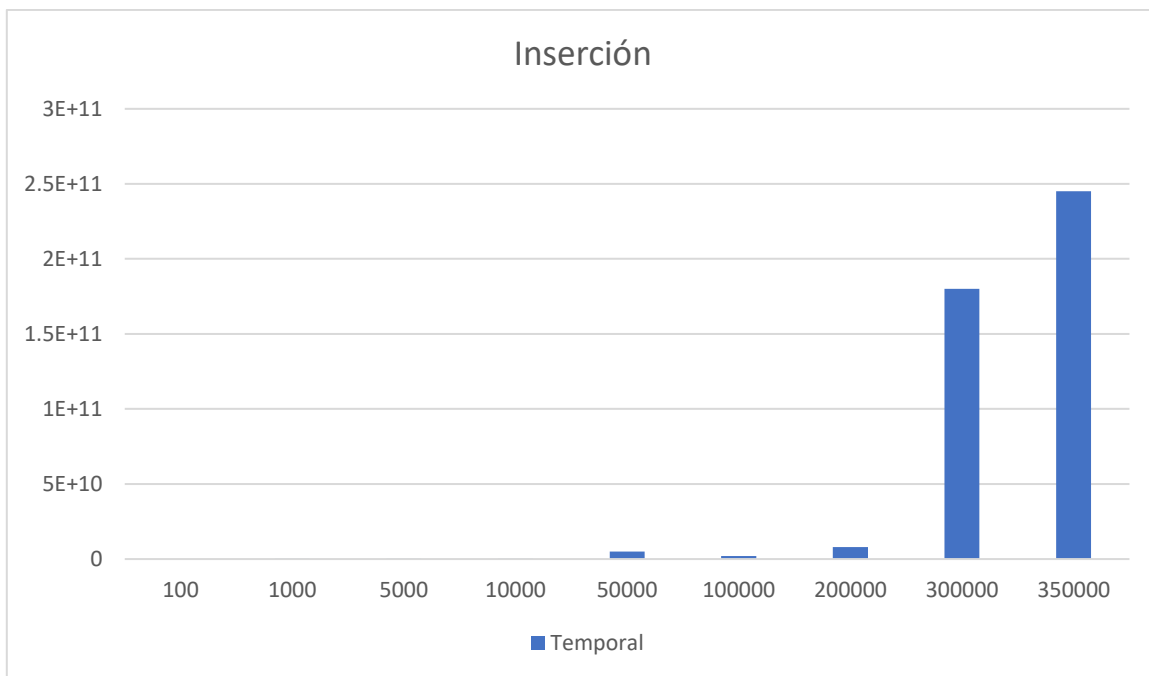
### Selección

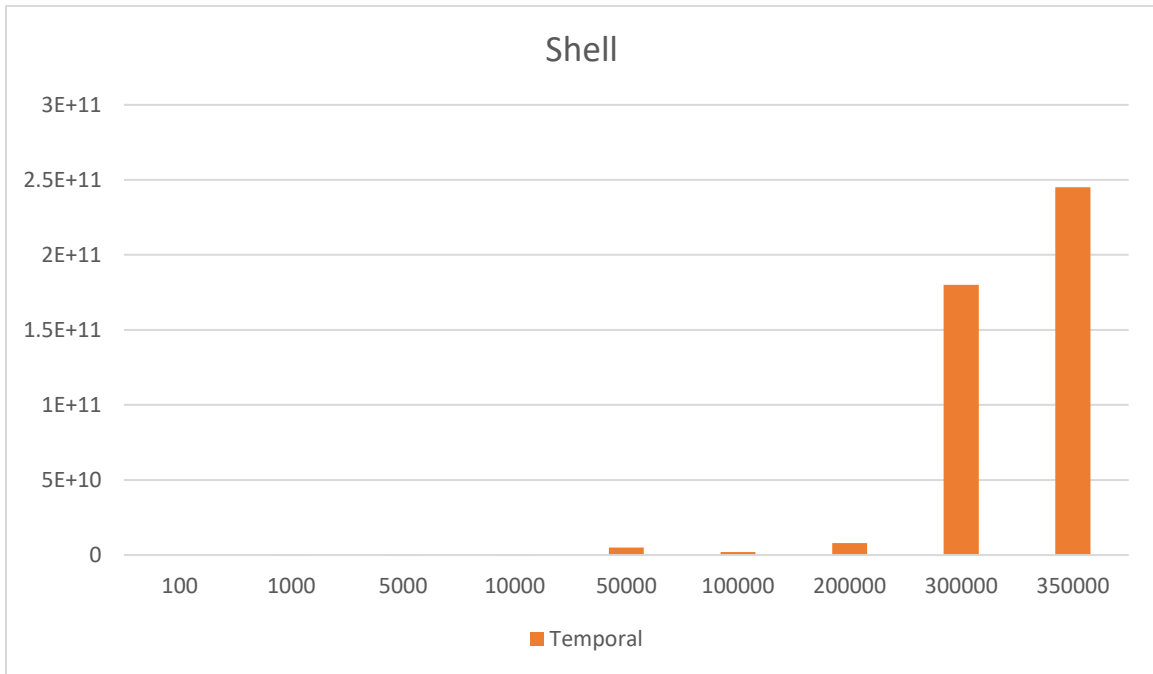
| k      | Instrucciones        |
|--------|----------------------|
| 100    | 40502                |
| 1000   | 4005002              |
| 5000   | 100025002            |
| 10000  | 400050002            |
| 50000  | $1 \times 10^{10}$   |
| 100000 | $4 \times 10^{10}$   |
| 200000 | $1.6 \times 10^{10}$ |
| 300000 | $3.6 \times 10^{11}$ |
| 350000 | $4.9 \times 10^{11}$ |

### Shell

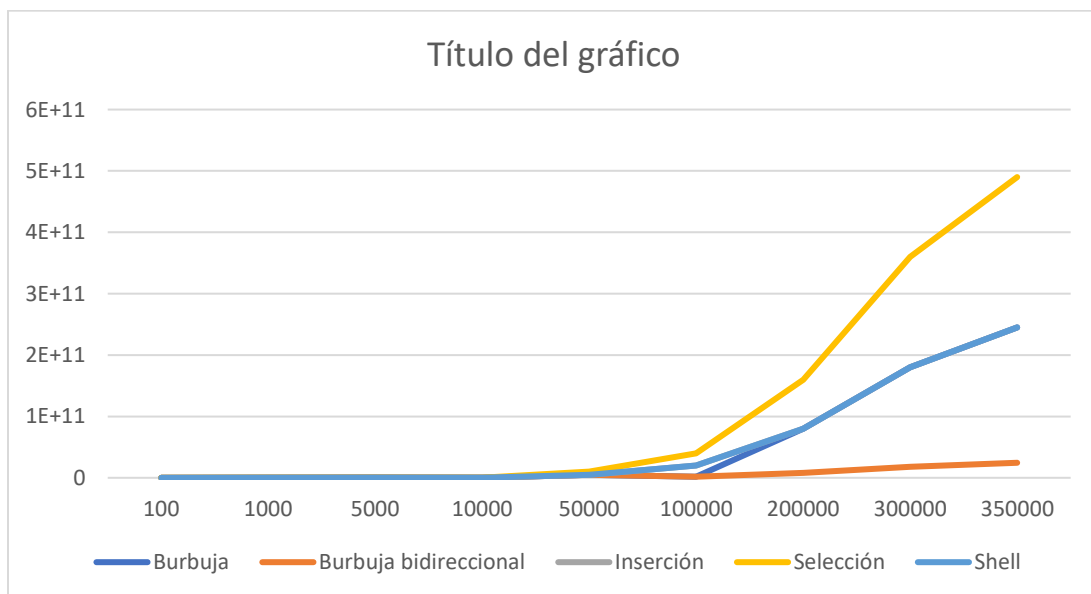
| k      | Instrucciones         |
|--------|-----------------------|
| 100    | 20912                 |
| 1000   | 2009012               |
| 5000   | 50045012              |
| 10000  | 200090012             |
| 50000  | 5000450012            |
| 100000 | $2 \times 10^{10}$    |
| 200000 | $8 \times 10^{10}$    |
| 300000 | $1.8 \times 10^{11}$  |
| 350000 | $2.45 \times 10^{11}$ |







Como se puede observar en las gráficas, todos los algoritmos de ordenamiento, previamente mencionados van a tener similar número de instrucciones con n numero de elementos en el arreglo y mientras esa n se vaya haciendo más grande, el número de instrucciones a ejecutarse se incrementará. Ahora se presenta una gráfica en la cual se comparan los 5 algoritmos con su máximo número de instrucciones y se definirá cual es el más recomendable para estos casos.



Con esta gráfica podemos concluir que el mejor algoritmo que podemos utilizar para una situación de una entrada grande números en el arreglo es el de inserción debido a que temporalmente es más eficiente

Las operaciones básicas que se consideraron para el cálculo de esta complejidad, fueron el número de comparaciones que se realizan en los ciclos for y en el ciclo while se sumaron todas las instrucciones, asignaciones, operaciones y al final nos da una función polinomial:

$$T = 2n^2 + 10n + 10$$

Con base en las aproximaciones que se obtuvieron, calculé el tiempo real aproximado de cada algoritmo cuando sus entradas son mucho más grandes de lo que 50,000,000, suponiendo que se cuenta con un equipo que se logre realizar el proceso. Los resultados se colocaron en una tabla para cada algoritmo. Los tiempos están dados en segundos.

### Burbuja

| No. de operaciones | Tiempo real   |
|--------------------|---------------|
| 50,000,000         | 65,371.4 s    |
| 100,000,000        | 130,742.8 s   |
| 500,000,000        | 653,714.2 s   |
| 1,000,000,000      | 1,307,428.5 s |
| 5,000,000,000      | 6,537,142.8 s |

### Burbuja bidireccional

| No. de operaciones | Tiempo real   |
|--------------------|---------------|
| 50,000,000         | 35,385.7 s    |
| 100,000,000        | 70,771.4 s    |
| 500,000,000        | 353,857.1 s   |
| 1,000,000,000      | 707,714.2 s   |
| 5,000,000,000      | 3,538,571.4 s |

### Inserción

| No. de operaciones | Tiempo real   |
|--------------------|---------------|
| 50,000,000         | 14,728.5 s    |
| 100,000,000        | 29,457.1 s    |
| 500,000,000        | 147,285.7 s   |
| 1,000,000,000      | 294,571.4 s   |
| 5,000,000,000      | 1,472,857.1 s |



## Selección

| No. de operaciones | Tiempo real |
|--------------------|-------------|
| 50,000,000         | 2,385.7 s   |
| 100,000,000        | 4,771.4 s   |
| 500,000,000        | 23,857.1 s  |
| 1,000,000,000      | 47,714.2 s  |
| 5,000,000,000      | 238,571.4 s |

## Shell

| No. de operaciones | Tiempo real |
|--------------------|-------------|
| 50,000,000         | 2,742.8 s   |
| 100,000,000        | 5,485.7 s   |
| 500,000,000        | 27,428.5 s  |
| 1,000,000,000      | 54,857.1 s  |
| 5,000,000,000      | 274,285.7 s |

- a) **¿Cuál de los 5 algoritmos es más fácil de implementar?**  
Considero que el más fácil e implementar es el de burbuja, ya que es el más simple y directamente es de los primeros que te enseñan
- b) **¿Cuál de los 5 algoritmos es el más difícil de implementar?**  
El más complicado de implementar, pienso que es el Shell ya que es un poco más complejo que los demás
- c) **¿Cuál algoritmo tiene menor complejidad temporal?**  
El de burbuja
- d) **¿Cuál algoritmo tiene mayor complejidad temporal?**  
El de selección
- e) **¿Cuál algoritmo tiene menor complejidad espacial? ¿Por qué?**  
El de burbuja y el de inserción
- f) **¿Cuál algoritmo tiene mayor complejidad espacial? ¿Por qué?**  
El de burbuja bidireccional
- g) **¿El comportamiento experimental de los algoritmos era el esperado?**  
En mi caso no lo fue
- h) **¿Por qué?**  
Debido a que no cuento con un equipo que soporte dicha cantidad de elementos, se complicó o un poco
- i) **¿Sus resultados experimentales difieren mucho de los del resto de los equipos? ¿A qué se debe?**  
Probablemente se deba a precisamente la falta de recursos en mi equipo para poder realizar pruebas adecuadas
- j) **¿Existió un entorno controlado para realizar las pruebas experimentales? ¿Cuál fue?** Si, en mi caso realicé las pruebas en el IDE Dev C++  
**¿Qué recomendaciones darían a nuevos equipos para realizar esta**

k) **práctica?** Que tenga un espacio de memoria amplio para las pruebas y que se realice en un equipo con buen rendimiento

## Pruebas

# Burbuja

The image is a screenshot of a Windows 10 desktop with the Visual Studio Code editor open. The editor window is titled 'C:\Users\Joel\OneDrive\Documents\Burbuja.exe' and contains a C++ program. The code is a bubble sort algorithm, with comments in Spanish. The file explorer on the left shows the project structure, including 'Burbuja.c', 'Inserción.c', 'Selección.c', 'Shell.c', and 'Burbujadireccional.c'. The status bar at the bottom indicates 'Line: 12, Col: 14, Sel: 0, Lines: 39, Length: 613, Insertar, Done parsing in 0.015 seconds'. The taskbar at the bottom shows various application icons and the system clock.

## Burbuja bidireccional

C:\Users\joelc\OneDrive\Documentos\Insercion.exe

```

682 32682 32683 32683 32683 32684 32684 32684 32684 32684 32684 32684 32684 32684 32684 32684 32684 32685 32685 32685 32685 32
685 32685 32686 32686 32686 32686 32686 32686 32686 32686 32687 32687 32687 32687 32687 32687 32687 32687 32688 32688 32688 32688 32
688 32688 32688 32688 32688 32688 32688 32688 32688 32688 32689 32689 32689 32689 32689 32689 32689 32689 32690 32690 32690 32690 32
691 32691 32691 32691 32691 32691 32691 32691 32691 32691 32692 32692 32692 32692 32692 32692 32692 32692 32693 32693 32693 32693 32
694 32694 32694 32694 32694 32694 32694 32694 32694 32695 32695 32695 32695 32695 32695 32695 32695 32696 32696 32696 32696 32
698 32698 32698 32699 32699 32699 32699 32699 32699 32700 32700 32700 32700 32700 32701 32701 32701 32701 32702 32702 32702 32702 32
702 32702 32702 32702 32703 32703 32703 32703 32703 32704 32704 32704 32704 32704 32705 32705 32705 32705 32706 32706 32706 32706 32
706 32706 32706 32706 32707 32707 32707 32707 32707 32708 32708 32708 32708 32708 32709 32709 32709 32709 32710 32710 32710 32710 32
710 32710 32711 32711 32711 32711 32711 32711 32711 32712 32712 32712 32712 32712 32713 32713 32713 32713 32714 32714 32714 32714 32
714 32714 32714 32714 32714 32715 32715 32715 32715 32715 32716 32716 32716 32716 32717 32717 32717 32717 32718 32718 32718 32718 32
717 32717 32717 32717 32718 32718 32718 32718 32718 32719 32719 32719 32719 32719 32720 32720 32720 32720 32721 32721 32721 32721 32
720 32720 32720 32720 32720 32720 32721 32721 32721 32721 32721 32722 32722 32722 32722 32722 32722 32722 32723 32723 32723 32723 32
724 32724 32724 32724 32724 32724 32725 32725 32725 32725 32725 32726 32726 32726 32726 32726 32726 32726 32727 32727 32727 32727 32
727 32727 32727 32728 32728 32728 32728 32728 32728 32728 32729 32729 32729 32729 32729 32729 32729 32729 32730 32730 32730 32730 32
730 32730 32730 32730 32730 32730 32731 32731 32731 32731 32731 32731 32731 32731 32731 32731 32731 32731 32732 32732 32732 32732 32
732 32732 32732 32732 32732 32733 32733 32733 32733 32733 32734 32734 32734 32734 32734 32734 32734 32734 32735 32735 32735 32735 32
735 32735 32735 32735 32735 32735 32736 32736 32736 32736 32736 32736 32736 32736 32736 32736 32736 32737 32737 32737 32737 32738 32738 32738 32
739 32739 32739 32739 32739 32739 32740 32740 32740 32740 32740 32741 32741 32741 32741 32741 32741 32741 32742 32742 32742 32742 32743 32743 32743 32
742 32742 32742 32742 32742 32743 32743 32743 32743 32743 32744 32744 32744 32744 32744 32744 32744 32744 32745 32745 32745 32745 32746 32746 32746 32
744 32745 32745 32745 32745 32746 32746 32746 32746 32746 32746 32747 32747 32747 32747 32747 32747 32747 32748 32748 32748 32748 32749 32749 32749 32
748 32748 32748 32748 32748 32748 32748 32748 32748 32749 32749 32749 32749 32749 32749 32749 32749 32749 32750 32750 32750 32750 32751 32751 32751 32
749 32749 32749 32750 32750 32750 32750 32750 32750 32751 32751 32751 32751 32751 32752 32752 32752 32752 32753 32753 32753 32753 32754 32754 32754 32
753 32753 32753 32753 32753 32754 32754 32754 32754 32754 32755 32755 32755 32755 32755 32755 32755 32755 32756 32756 32756 32756 32757 32757 32757 32
755 32755 32756 32756 32756 32756 32756 32756 32756 32756 32757 32757 32757 32757 32757 32757 32757 32757 32758 32758 32758 32758 32759 32759 32759 32
758 32758 32758 32758 32758 32759 32759 32759 32759 32759 32759 32760 32760 32760 32760 32760 32760 32760 32761 32761 32761 32761 32762 32762 32762 32
761 32761 32761 32761 32761 32761 32762 32762 32762 32762 32762 32763 32763 32763 32763 32763 32763 32763 32764 32764 32764 32764 32765 32765 32765 32
766 32766 32766 32766 32766 32766 32766 32766 32767 32767 32767 32767 32767 32767 32767 32767 32767 32767 32768 32768 32768 32768 32769 32769 32769 32

```

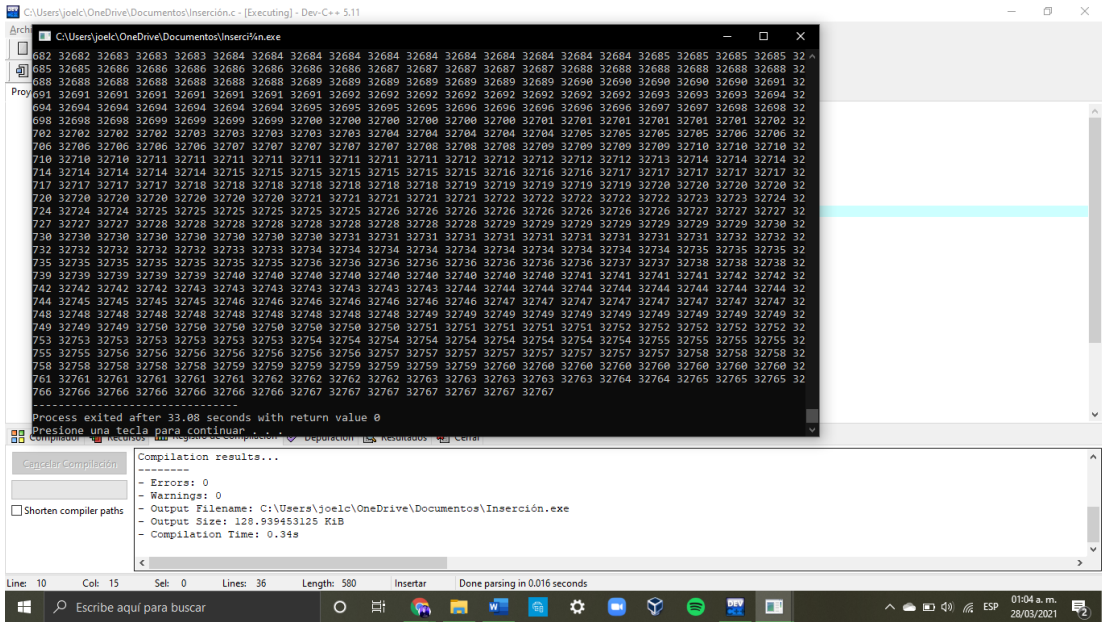
Process exited after 33.08 seconds with return value 0

Presione una tecla para continuar . . .

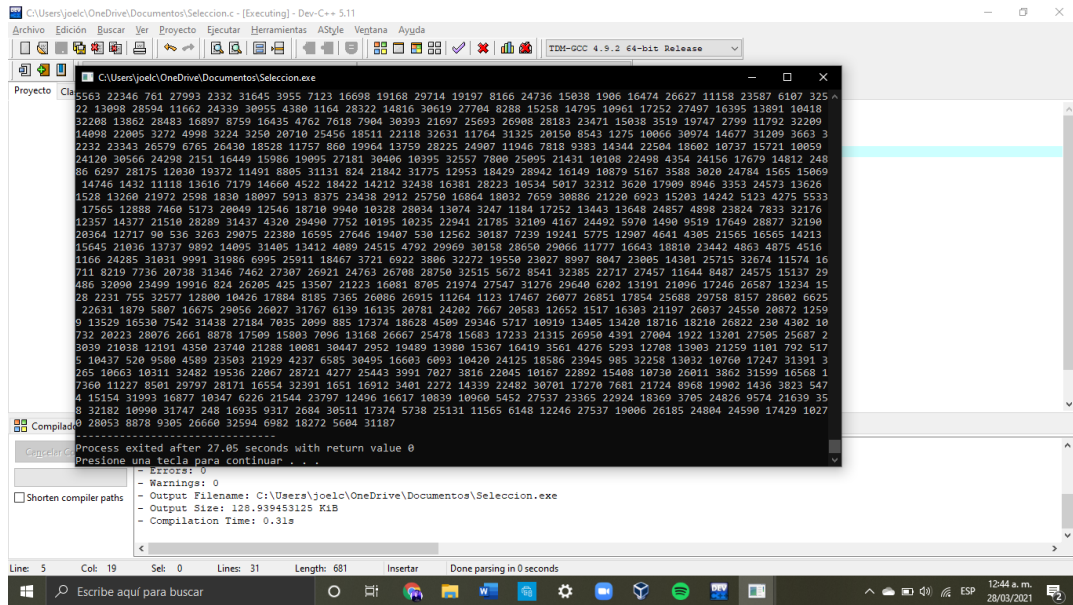
Compilation results...

- Errors: 0
- Warnings: 0

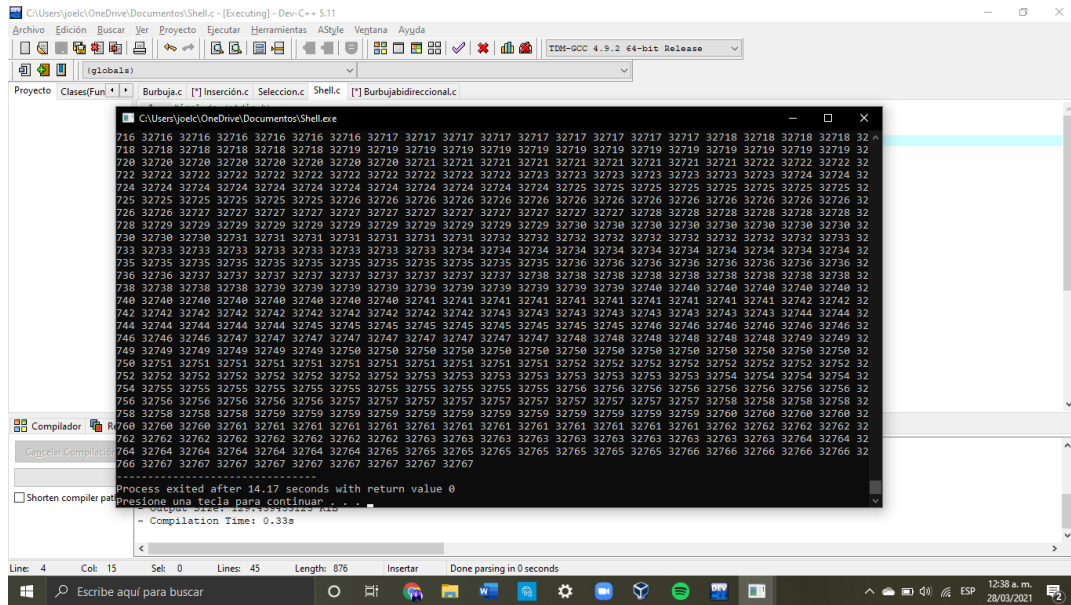
## Inserción



## Selección



## Shell



```
C:\Users\joelc\OneDrive\Documentos\Shell.c - [Executing] - Dev-C++ 5.11
Archivo Edición Buscar Ver Proyecto Ejecutar Herramientas ASstyle Ventana Ayuda
(globals)
Proyecto Clases(Fun) Burbuja.c [*] Inserción.c Selección.c Shell.c [*] Burbujadireccional.c
C:\Users\joelc\OneDrive\Documentos\Shell.exe
716 32716 32716 32716 32716 32716 32716 32717 32717 32717 32717 32717 32717 32717 32717 32718 32718 32718 32718 32
718 32718 32718 32718 32718 32719 32719 32719 32719 32719 32719 32719 32719 32719 32719 32719 32719 32719 32719 32
720 32720 32720 32720 32720 32720 32720 32721 32721 32721 32721 32721 32721 32721 32721 32721 32721 32721 32721 32
722 32722 32722 32722 32722 32722 32722 32722 32722 32722 32722 32722 32722 32722 32722 32722 32722 32722 32722 32
724 32724 32724 32724 32724 32724 32724 32724 32724 32724 32724 32724 32724 32724 32724 32724 32724 32724 32724 32
725 32725 32725 32725 32725 32725 32725 32726 32726 32726 32726 32726 32726 32726 32726 32726 32726 32726 32726 32
726 32726 32727 32727 32727 32727 32727 32727 32727 32727 32727 32727 32727 32727 32727 32727 32727 32727 32727 32
728 32728 32729 32729 32729 32729 32729 32729 32729 32729 32729 32729 32730 32730 32730 32730 32730 32730 32730 32730 32
730 32730 32730 32731 32731 32731 32731 32731 32731 32731 32731 32732 32732 32732 32732 32732 32732 32732 32732 32
733 32733 32733 32733 32733 32733 32733 32733 32733 32733 32733 32733 32733 32733 32733 32733 32733 32733 32733 32
735 32735 32735 32735 32735 32735 32735 32735 32735 32735 32735 32735 32735 32735 32735 32735 32735 32735 32735 32
736 32736 32737 32737 32737 32737 32737 32737 32737 32737 32737 32737 32738 32738 32738 32738 32738 32738 32738 32738 32
738 32738 32738 32738 32738 32738 32738 32739 32739 32739 32739 32739 32739 32739 32739 32739 32739 32739 32739 32
740 32740 32740 32740 32740 32740 32740 32740 32741 32741 32741 32741 32741 32741 32741 32741 32741 32741 32741 32741 32
742 32742 32742 32742 32742 32742 32742 32742 32742 32742 32742 32742 32742 32742 32742 32742 32742 32742 32742 32742 32
744 32744 32744 32744 32744 32744 32744 32745 32745 32745 32745 32745 32745 32745 32745 32745 32745 32745 32745 32745 32
746 32746 32746 32746 32746 32746 32746 32746 32746 32746 32746 32746 32746 32746 32746 32746 32746 32746 32746 32746 32
748 32748 32748 32748 32748 32748 32748 32748 32748 32748 32748 32748 32748 32748 32748 32748 32748 32748 32748 32748 32
750 32750 32750 32750 32750 32750 32750 32750 32750 32750 32750 32750 32750 32750 32750 32750 32750 32750 32750 32750 32
752 32752 32752 32752 32752 32752 32752 32752 32752 32752 32752 32752 32752 32752 32752 32752 32752 32752 32752 32752 32
754 32754 32754 32754 32754 32754 32754 32754 32754 32754 32754 32754 32754 32754 32754 32754 32754 32754 32754 32754 32
756 32756 32756 32756 32756 32756 32756 32756 32756 32756 32756 32756 32756 32756 32756 32756 32756 32756 32756 32756 32
758 32758 32758 32758 32758 32758 32758 32758 32758 32758 32758 32758 32758 32758 32758 32758 32758 32758 32758 32758 32
760 32760 32760 32760 32760 32760 32760 32760 32760 32760 32760 32760 32760 32760 32760 32760 32760 32760 32760 32760 32760 32
762 32762 32762 32762 32762 32762 32762 32762 32762 32762 32762 32762 32762 32762 32762 32762 32762 32762 32762 32762 32762 32
764 32764 32764 32764 32764 32764 32764 32764 32764 32764 32764 32764 32764 32764 32764 32764 32764 32764 32764 32764 32764 32
766 32766 32766 32766 32766 32766 32766 32766 32766 32766 32766 32766 32766 32766 32766 32766 32766 32766 32766 32766 32766 32
-----
Process exited after 14.17 seconds with return value 0
Presione una tecla para continuar . . .
- Compilation Time: 0.33s
Line: 4 Col: 15 Sel: 0 Lines: 45 Length: 876 Insertar Done parsing in 0 seconds
Escribe aquí para buscar
```

## Bibliografía:

[http://wh.free.fr/pages/algo/tri/tri\\_shaker\\_es.html](http://wh.free.fr/pages/algo/tri/tri_shaker_es.html)

<https://runestone.academy/runestone/static/pythoned/SortSearch/EIOrdenamientoDeShell.html>

[https://www.ecured.cu/Algoritmo\\_de\\_ordenamiento\\_por\\_selecci%C3%B3n](https://www.ecured.cu/Algoritmo_de_ordenamiento_por_selecci%C3%B3n)