



VNIVERSITAT DE VALÈNCIA

ESTRUCTURAS DE DATOS Y ALGORITMOS*Grado en Ingeniería Multimedia (2º). Curso 2022-23***Práctica Nº 2: Coste de funciones de ordenación**Periodo de realización: Semana del **17/10/2022** a **21/10/2022****Objetivos**

- Cálculo de costes reales (empíricos) de funciones.
- Representación gráfica e interpretación de resultados.
- Utilización de *Gnuplot*.

gnuplot

El programa GNUPLOT es una utilidad para la visualización de datos, que utilizaremos para representar los ficheros generados por los programas que vamos a realizar (puedes consultar el mecanismo de descarga y ejecución de este programa en el [anexo disponible al final de este guion](#)). El funcionamiento se realiza mediante órdenes de texto, indicando qué acción se desea realizar. A continuación, se muestran algunas órdenes básicas que pueden ser de utilidad en esta práctica¹:

Visualizar archivos de datos

```
plot "fichero" w p
```

Dibuja los datos almacenados en *fichero* como puntos.

```
plot "fichero" w l
```

Dibuja los datos almacenados en *fichero* uniendo con líneas.

```
plot "fichero" w p, "fichero2" w l
```

Dibuja el contenido de los dos ficheros en la misma gráfica (uno con puntos y otro con líneas). Se pueden dibujar varios ficheros sin más que seguir el mismo esquema.

Definir y visualizar funciones

```
f(x) = x**2
```

Define la función $f(x)=x^2$ para utilizarla posteriormente.

```
plot "fichero" w l, f(x)
```

Dibuja el fichero y la función $f(x)$, declarada previamente, en la misma gráfica.

```
plot [0:10] f(x)
```

Dibuja la función $f(x)$ para valores de x de 0 a 10.

```
plot [0:10][0:500] f(x)
```

Dibuja la función $f(x)$ para valores de x de 0 a 10 y mostrando el eje de ordenadas (y) de 0 a 500.

¹ En el Aula Virtual, en "Libros y materiales electrónicos", se puede descargar el manual completo de *Gnuplot*.

Guardar las gráficas en archivos de salida

```
set terminal jpeg
```

Cambia el formato de salida a jpeg. Con "set terminal" podemos ver otras opciones.

```
set output "fichero.jpg"
```

Establece el nombre del fichero de salida.

```
plot f(x)
```

Guarda la gráfica de $f(x)$ en el fichero "fichero.jpg" establecido como salida.

```
set terminal qt, o bien, set terminal windows
```

Establece la salida en una ventana del sistema.

Salir de gnuplot

```
quit
```

Introducción

En esta práctica se va a analizar empíricamente el coste temporal de 3 algoritmos/funciones típicos de ordenación de vectores: *inserción*, *selección* y *quicksort*. El código en C++ de estas 3 funciones se proporciona ya implementado en el archivo "*pr2_funciones_ordenar.h*", que se puede descargar desde el Aula Virtual. Este archivo incluye además la definición de tipos de datos y alguna función adicional de utilidad. Por lo tanto, un requisito previo para trabajar en esta práctica es conocer e interpretar adecuadamente el contenido de este archivo.

El objetivo de la práctica es medir el número de pasos realizados durante la ejecución real de estos tres algoritmos sobre diferentes instancias del problema y compararlos con los resultados teóricos esperados. Para medir estos pasos solo se van a tener en cuenta las secciones críticas de cada algoritmo. En este caso, se han considerado solo aquellas sentencias que implican accesos a elementos del vector. Para este ejercicio se consideran irrelevantes las restantes sentencias de los algoritmos. Las funciones proporcionadas como material ya realizan la contabilidad de pasos de cada algoritmo, que se proporciona como valor de retorno de la función correspondiente².

Las tareas a realizar en esta práctica están orientadas a diseñar los experimentos que permitan obtener resultados del coste de cada algoritmo para diferentes tallas del vector y diferentes instancias para esa talla.

Ejercicios

Tarea 0: Iniciar el nuevo proyecto

Se deben descargar desde el Aula Virtual los archivos con código fuente proporcionados como material inicial y ubicarlos adecuadamente, utilizando el navegador de archivos del sistema operativo, dentro de la carpeta "Pr2" del repositorio de prácticas. Se debe organizar la carpeta "Pr2" con la misma estructura interna que se ha utilizado en las prácticas 0 y 1 (carpeta "*src*", "*include*", "*data*", etc.).

² *size_t* es un tipo estándar de C++ que corresponde con el entero sin signo más grande que se puede representar en nuestro ordenador.

Abre la carpeta "Pr2" desde *VSCode* y comprueba que es posible navegar y editar los archivos disponibles. Debes hacer *commit* de los cambios realizados en el proyecto y propagarlos al servidor (*push*). Indica como mensaje de commit el texto "Pr2: Carga inicial de archivos".

En las siguientes tareas deberás realizar commits periódicamente cada vez que realices algún avance significativo en el desarrollo del proyecto. No se van a dar directrices sobre este aspecto, pero se evaluará la correcta utilización del repositorio.

Tarea 1: Representar los costes teóricos

Representa gráficamente mediante *gnuplot* del coste teórico del caso medio de los tres algoritmos. Los costes aproximados para el caso medio de cada algoritmo son:

	Inserción	Selección	Quicksort
$T^{\mu}(n) =$	$\frac{1}{2}n^2$	$\frac{1}{2}(n^2 - 5n)$	$2n\log_2 n + n$

Visualiza conjuntamente las 3 funciones para poder comparar su evolución con la talla. Guarda la gráfica en el fichero "**tarea1.jpg**". Este archivo deberá estar ubicados en la carpeta "data"³. En la primera gráfica se representarán valores de n entre 0 y 1000.

Secuencias de órdenes para *Gnuplot*⁴

1) Definir las funciones de coste para cada función y visualizar la gráfica en pantalla:

```
gnuplot> set dummy n
gnuplot> TIns(n)= n**2
gnuplot> TSel(n)= (n**2-5*n)/2
gnuplot> TQS(n)= 2*n*log(n)/log(2) + n
gnuplot> plot [0:1000][0:] TIns(n), TSel(n), TQS(n)
```

2) Establecer la salida en el archivo jpeg y generarlo (*plot* no mostrará las gráficas en pantalla):

```
gnuplot> set terminal jpeg
gnuplot> set output "tarea1.jpg"
gnuplot> plot [0:1000][0:] TIns(n), TSel(n), TQS(n)
```

3) Reestablecer a la salida por pantalla para las siguientes acciones:

```
gnuplot> set terminal qt
```

Nota: Se recomienda realizar primero la gráfica en pantalla y posteriormente, cuando ya se ha verificado que se muestra correctamente, generar el archivo de salida. Al finalizar, comprueba que se ha creado el archivo y que es posible visualizar la imagen.

³ Asegúrate de cambiar la carpeta de salida de *gnuplot* ("Change directory" en menú "File") para ubicar correctamente estos archivos.

⁴ *Gnuplot* no tiene definida la función logaritmo en base 2. Sin embargo, sí que tiene definida la función logaritmo en base 10 (*log*). Utilizando una de las propiedades del logaritmo se puede calcular \log_2 en función de \log_{10} con la siguiente expresión: $\log_2(x) = \log_{10}(x)/\log_{10}(2)$.

Tarea 2: Cálculo empírico de costes

Para verificar empíricamente el comportamiento del coste de los algoritmos de ordenación, se debe implementar un programa “**main.cpp**”. Este programa se debe ajustar a los siguientes requisitos:

- Las tres funciones que se han proporcionado para implementar los algoritmos de ordenación ya incluyen un contador de pasos, en forma de valor de retorno. El contador contabiliza los pasos de acuerdo con los criterios indicados en el apartado “Introducción” de este guion.
- El programa debe obtener datos del coste de cada función de ordenación para diferentes valores de la talla. Para automatizar el proceso se debe implementar una función que ejecute las 3 funciones de ordenación para vectores de diferentes tallas dentro de un rango y volcar los resultados obtenidos a un archivo. El prototipo de esta función será:

```
void Experimento (unsigned tini, unsigned tfin, int inc);
```

La función deberá obtener resultados del coste de realizar la ordenación de vectores en un rango de tallas comprendido entre *tini* y *tfin* (ambos incluidos), con un incremento *inc* entre tallas⁵. El resultado de los experimentos sobre cada algoritmo y talla se registrará en un fichero de texto, diferente para cada algoritmo (llamados “**salida_INS.dat**”, “**salida_SEL.dat**” y “**salida_QS.dat**”). Estos archivos deberán estar ubicados en la carpeta “data”.

- Obviamente, el coste de cada algoritmo dependerá de la organización del vector antes de ordenarlo, por lo que, no tiene sentido proporcionar el resultado de una única ordenación. Por eso, la función *Experimento* debe realizar el siguiente proceso para cada valor de la talla (*n*):
 - Generar un vector ordenado en orden creciente (menor a mayor) de *n* elementos⁶. Después, aplicar las tres funciones de ordenación sobre este vector y obtener el coste de cada una de ellas.
 - Generar un vector ordenado en orden decreciente (mayor a menor) de *n* elementos⁷. Después, aplicar las tres funciones de ordenación sobre este vector y obtener el coste de cada una de ellas.
 - Para estimar el coste medio, se deben realizar *n* pruebas (tantas como la talla) con vectores aleatoriamente desordenados de *n* elementos (ver función *Desordenar*). En cada una de estas pruebas el vector generado debe ser ordenado por las tres funciones, acumular los pasos realizados y, finalmente, calcular la media total de cada algoritmo (que siempre será un número real).
 - Al finalizar los 3 pasos anteriores se debe escribir en cada archivo de salida (uno por algoritmo) la talla y los tres valores obtenidos (pasos con el vector ordenado, pasos con el vector inversamente ordenado y media de pasos con vectores aleatoriamente ordenados). Para cada talla solo habrá una línea en el archivo, como en el siguiente ejemplo (talla=50, pasos orden=196, pasos inverso=2646, media pasos aleatorio = 1436.96):

```
50      196      2646      1436.96
```

⁵ Por ejemplo, entre 0 y 1000 con incrementos de 25 implica tallas: 0, 25, 50, 75, 100, 125,..., 950, 975, 1000.

⁶ Corresponde a la instancia del mejor caso para los 3 algoritmos.

⁷ Corresponde a la instancia del peor caso para Inserción y Selección, pero no para Quicksort. Para este algoritmo está muy cerca de ser el mejor caso (!).

De esta manera, al repetir el proceso para todas las tallas en cada archivo tendremos datos representativos de diferentes instancias del problema y se podrá representar su evolución en función de la talla y comparar con los datos teóricos.

El programa principal debe invocar a la función *Experimento* para calcular los costes de los algoritmos en un rango de **tallas de 0 a 1000 con incrementos de 25**, generando un archivo de resultados para cada algoritmo con el nombre indicado previamente. Estos ficheros contendrán los costes reales que se compararán con los costes teóricos en el caso medio.

Tarea 3: Representación de resultados

Representar gráficamente cada uno de los 3 archivos obtenidos junto con el coste en el caso medio teórico del algoritmo correspondiente. De manera que, si se suponen definidas las funciones $TIns(n)$, $TSel(n)$ y $TQS(n)$ de la tarea 1, se deben generar 3 gráficas: (1) $TIns(n)$ junto con los datos del archivo “salida_INS.dat”, (2) $Tbc(n)$ junto con los datos del archivo “salida_SEL.dat” y (3) $Tbb(n)$ junto con los datos del archivo “salida_QS.dat”. Esas gráficas se deben guardar (carpeta 'dat') en los archivos “tarea3_INS.jpg”, “tarea3_SEL.jpg” y “tarea3_QS.jpg”, respectivamente.

Ejemplo de secuencias de órdenes para *Gnuplot* para estas gráficas

1) Se suponen definidas las funciones $TIns(n)$, $TSel(n)$ y $TQS(n)$ utilizadas en la tarea 1.

2) Establecer la salida en los archivos jpeg (*plot* no mostrará las gráficas en pantalla):

```
gnuplot> set terminal jpeg
gnuplot> set output "tarea3_INS.jpg"
gnuplot> plot [0:1000][0:1e6] "salida_INS.dat" using 1:2, "salida_INS.dat"
using 1:3, "salida_INS.dat" using 1:4, TIns(n)
gnuplot> set output "tarea3_SEL.jpg"
gnuplot> plot [0:1000][0:1e6] "salida_SEL.dat" using 1:2, "salida_SEL.dat"
using 1:3, "salida_SEL.dat" using 1:4, TSel(n)
gnuplot> set output "tarea3_QS.jpg"
gnuplot> plot [0:1000][0:25e3] "salida_QS.dat" using 1:2, "salida_QS.dat" using
1:3, "salida_QS.dat" using 1:4, TQS(n)
```

3) Reestablecer a la salida por pantalla para las siguientes acciones:

```
gnuplot> set terminal qt
```

Verifica si los datos reales para el caso medio se ajustan a las funciones de coste calculadas teóricamente para este caso.

Tarea 4: Generación de documentación

El código del programa deberá estar correctamente documentado con Doxygen, de acuerdo con la guía de estilo. Una vez finalizado el programa, se deberá ejecutar Doxygen para verificar la correcta generación de la documentación del programa.

Consideraciones MUY IMPORTANTES

1. Cada vector generado en los experimentos se debe ordenar secuencialmente por 3 métodos diferentes. Como las funciones de ordenación modifican el vector de entrada, después de aplicar la primera función de ordenación el vector ya queda ordenado, por lo que, la segunda y la tercera función ya no aplican sobre la misma instancia del problema. Lo harían siempre sobre un vector previamente ordenado (mejor caso). **SOLUCIÓN:** Se debe generar el vector y después hacer una copia en una segunda variable antes de proceder a llamar a cada función de ordenación. De esta manera, lo que se ordena es una copia, nunca el vector original. Ejemplo: Si se ha generado un vector llamado *original*, entonces se debe realizar la secuencia de ordenaciones de la siguiente manera:

```
v = original;  
Ordenar por Insercion(v)  
v = original;  
Ordenar por Seleccion(v)  
v = original;  
Ordenar por Quicksort(v)
```

2. Al declarar una variable de tipo *vector* esta no tiene tamaño (*size()==0*), por lo tanto, no se pueden añadir elementos en posiciones concretas del vector porque no hay posiciones. La forma de añadir elementos es mediante la operación *push_back* (ver *OrdenarInsercion*). Además, como se van a generar múltiples vectores con el mismo nombre, es conveniente antes de generar un nuevo vector eliminar el contenido del anterior porque, en caso contrario, es posible que los nuevos elementos se añadan a los anteriores y el tamaño del vector crezca exponencialmente. Para borrar el contenido de un vector ya existente y que vuelva a tener tamaño 0 se puede utilizar la operación *clear*.

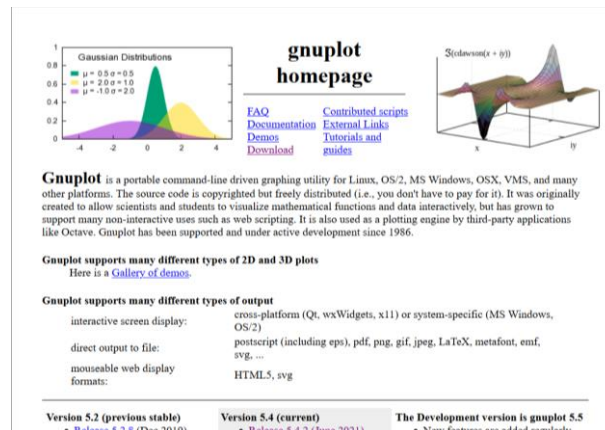
Se puede consultar información sobre el tipo *vector* en:

<https://cplusplus.com/reference/vector/vector/>

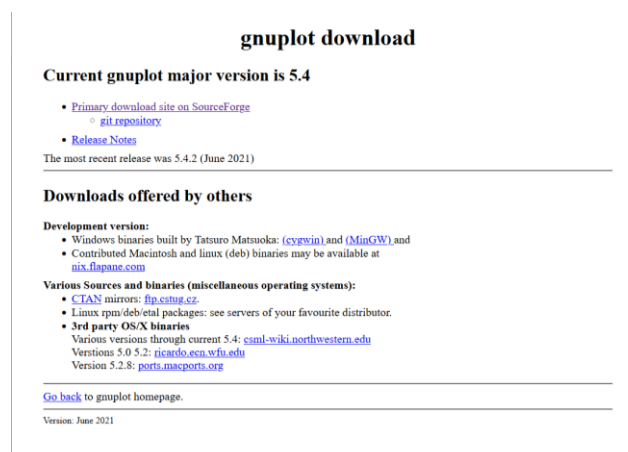
Anexo gnuplot

Descargar gnuplot

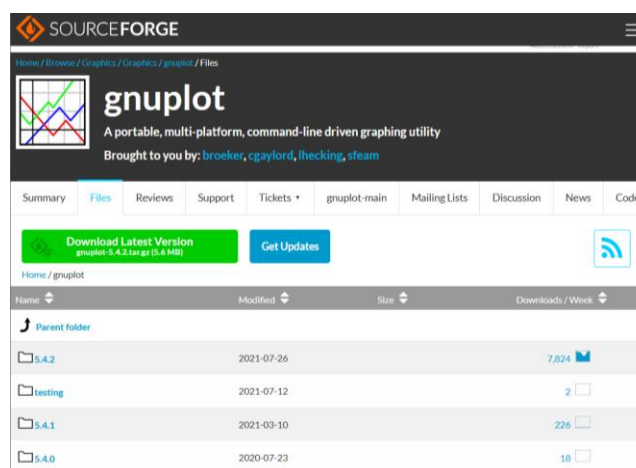
1. Acceder a la página oficial del programa: <http://www.gnuplot.info/>



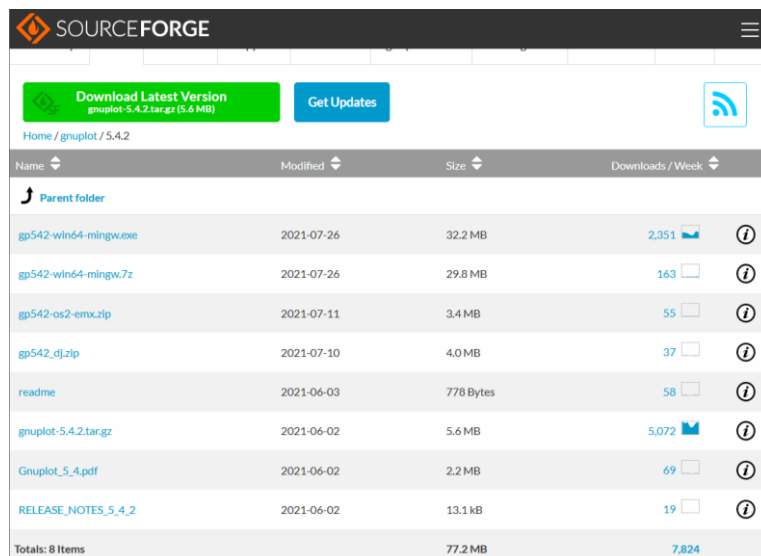
2. Seleccionar “Download”



3. Seleccionar “Primary download site on SourceForge”



Abrir la carpeta correspondiente a la versión más reciente, por ejemplo “5.4.5”:

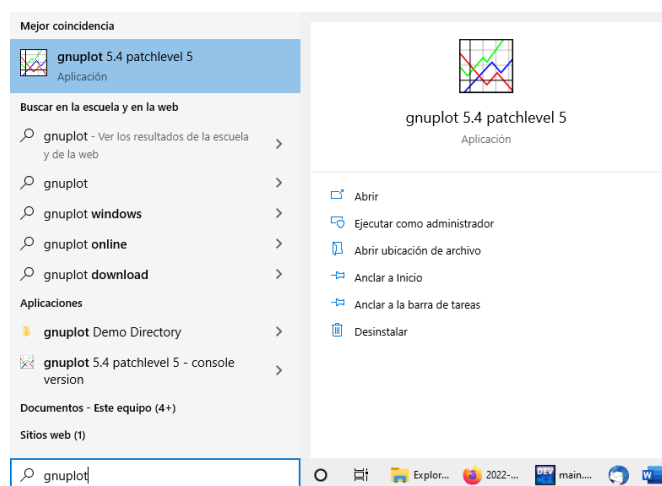


Name	Modified	Size	Downloads / Week
Parent folder			
gp542-win64-mingw.exe	2021-07-26	32.2 MB	2,351
gp542-win64-mingw.7z	2021-07-26	29.8 MB	163
gp542-os2-emx.zip	2021-07-11	3.4 MB	55
gp542_dj.zip	2021-07-10	4.0 MB	37
readme	2021-06-03	778 Bytes	58
gnuplot-5.4.2.tar.gz	2021-06-02	5.6 MB	5,072
Gnuplot_5.4.pdf	2021-06-02	2.2 MB	69
RELEASE_NOTES_5_4_2	2021-06-02	13.1 kB	19
Totals: 8 Items		77.2 MB	7,824

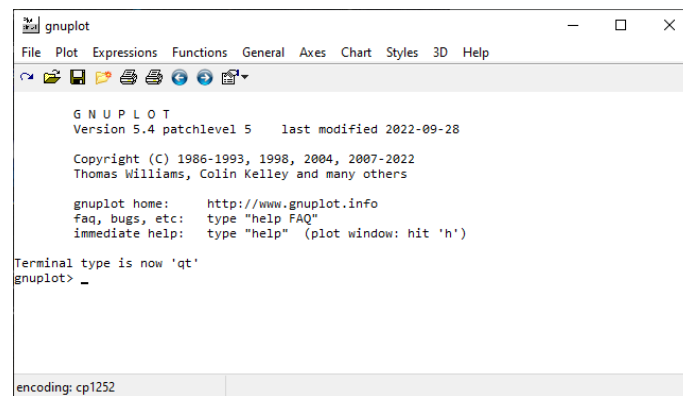
Se debe seleccionar la versión correspondiente a nuestro sistema operativo. Por ejemplo, si tenemos Windows, descargaremos el archivo ejecutable para este sistema, “gp545-win64-mingw.exe”. Tras unos segundos se iniciará la descarga del programa instalador. Tras finalizar la descarga, se debe ejecutar este archivo para proceder a la instalación del programa.

Ejecutar gnuplot

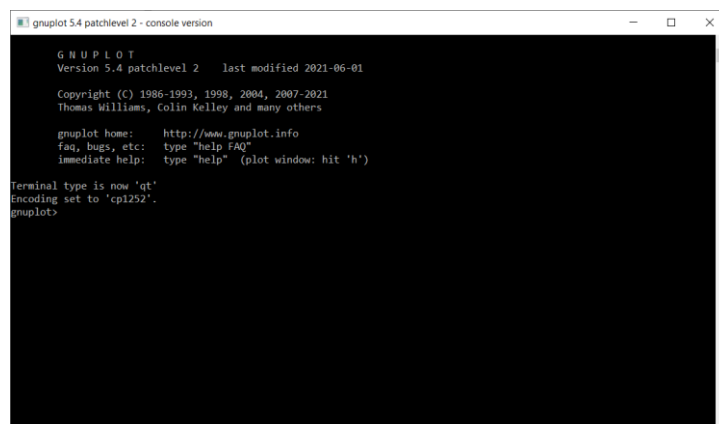
Para abrir el programa en Windows (si no tienes un acceso directo en el escritorio), se debe escribir/buscar la palabra “gnuplot” en la barra de Windows y seleccionar la opción “gnuplot 5.4 patchlevel 5”, tal como se muestra en la siguiente imagen (el número de versión y el texto exacto dependerá de la versión instalada):



A continuación, se abrirá el programa y mostrará la siguiente ventana:



También hay una versión de consola clásica (*console version*) con este aspecto:



El funcionamiento en cualquiera de los dos formatos es básicamente el mismo y en cualquiera de ellos se pueden introducir las órdenes de texto para realizar las tareas de la práctica.