# Análisis empírico y asintótico

## Objetivos

- Obtener el tiempo de ejecución empírico de algoritmos implementados en C++.
- Analizar el tiempo de ejecución empírico de diversos algoritmos utilizando gráficos.
- Analizar las tasas de crecimiento de la complejidad asintótica de diversos algoritmos utilizando gráficos.
- Implementar pseudocódigo de un algoritmo en C++.
- Trabajar en grupo.

## Enunciado

El siguiente algoritmo ordena ascendentemente los elementos de un vector V de tamaño n

```
función Inserción (V:&entero[n])
   i,j,x:entero
   para i←2 hasta n hacer
      x ← Vi
      j ← i-1
      mientras j>0 y Vj>x hacer
         Vj+1 ← Vj
         j ← j-1
      fmientras
      Vj+1 ← x
    fpara
ffunción
```

**Actividades (Equipos de 2 o 3 alumnos)**

1.- Implementa un programa para obtener el tiempo de ejecución empírico del algoritmo para realizar la ordenación de un vector de números enteros en un caso cualquiera, en el caso mejor y en el caso peor.

Los pasos que sigue el programa son:

**Paso 1.** Generar un vector del tamaño que el usuario introduzca por teclado e inicializarlo con números aleatorios.

Para generar los números aleatorios se cogerá como semilla un valor que se pida por teclado. Para establecer una semilla se utiliza la función srand(semilla), donde semilla es la variable que contiene el valor de la semilla a partir de la cual se generan los números aleatorios. Esta función solamente hay que llamarla una vez.

A partir de esta semilla se asignan valores a los elementos del vector. La función rand() genera números aleatorios. Cada vez que se llama a esta función se genera un nuevo número aleatorio.

Las funciones srand y rand se encuentran en la librería stdlib.h.

IMPORTANTE: El algoritmo de inserción considera vectores desde la posición 1 hasta la posición n, siendo n el tamaño del vector.

**Paso 2:** Utilizando los valores del vector generado en el paso anterior aplicar el algoritmo para un caso cualquiera, para el caso mejor y para el peor. En cada caso se mostrará cuál es el vector a ordenar, el vector ordenado tras aplicar el algoritmo y el tiempo de ejecución empírico en milisegundos que el algoritmo tarda en ordenar los vectores.

Para simplificar la salida por pantalla de los vectores, solamente se muestran los valores comprendidos entre las posiciones que el usuario introduce por teclado (ambas posiciones incluidas).

**Ejemplo:**

En el siguiente ejemplo se muestra el resultado tomando como semilla el valor 20, para un vector de 5000 elementos y mostrando por pantalla los elementos del vector que ocupan las posiciones desde el valor 1 hasta el 5. Todos estos valores se piden por teclado. Los tiempos de ejecución y valores aleatorios mostrados dependerán de la máquina en la que se ejecute el programa, compilador,…

Debido a que el cálculo del tiempo de ejecución que se obtiene es un valor aproximado, el programa también pedirá si se quiere mostrar o no el valor del tiempo de ejecución por pantalla. En el caso de que se introduzca un 1 se mostrará el valor por pantalla y en el caso de que sea 0 no se mostrará este valor. Este último parámetro es importante para que el programa funcione correctamente utilizando el corrector online. En las siguientes imágenes se muestran los resultados según el valor de este último parámetro.

```
Semilla para generar aleatorios:20
Introduce tamanyo del vector:5000
Posiciones INICIAL y FINAL del vector para mostrar
INICIAL:1
FINAL:5
Mostrar tiempo de ejecucion (1:Si, 0:No):1

ALGORITMO DE INSERCION

CASO CUALQUIERA
---------------
Vector a ordenar: 103 26079 18073 24951 18538
Vector ordenado: 7 9 14 23 29
Tiempo (ms):24

CASO MEJOR
---------------
Vector a ordenar: 7 9 14 23 29
Vector ordenado: 7 9 14 23 29
Tiempo (ms):0

CASO PEOR
---------------
Vector a ordenar: 32763 32763 32760 32757 32755
Vector ordenado: 7 9 14 23 29
Tiempo (ms):39

FIN DEL PROGRAMA
```

```
Semilla para generar aleatorios:20
Introduce tamanyo del vector:5000
Posiciones INICIAL y FINAL del vector para mostrar
INICIAL:1
FINAL:5
Mostrar tiempo de ejecucion (1:Si, 0:No):0

ALGORITMO DE INSERCION

CASO CUALQUIERA
---------------
Vector a ordenar: 103 26079 18073 24951 18538
Vector ordenado: 7 9 14 23 29
CASO MEJOR
---------------
Vector a ordenar: 7 9 14 23 29
Vector ordenado: 7 9 14 23 29
CASO PEOR
---------------
Vector a ordenar: 32763 32763 32760 32757 32755
Vector ordenado: 7 9 14 23 29
FIN DEL PROGRAMA
```

2.- Realiza un análisis comparativo de los tiempos de ejecución empírico y asintótico del comportamiento del algoritmo para diversos valores del tamaño del problema y muestra los resultados en un gráfico tal y como se indica a continuación. Escribe las conclusiones que obtienes de los resultados empíricos y asintóticos.

**Caso empírico:** Representa los valores de los tres casos: caso cualquiera, caso mejor y caso peor.

En la Figura 1 se muestra un ejemplo de un gráfico con los tiempos de ejecución empíricos para dos supuestos algoritmos llamados Algoritmo1 y Algoritmo2. En el eje horizontal se indica el tamaño del problema y en el vertical el tiempo de ejecución en milisegundos.

Como el tiempo de ejecución depende de factores externos hay que especificar cuáles son las características hardware del equipo físico (procesador, memoria RAM,…) y el software (sistema operativo, lenguaje de programación, compilador,..).

En esta práctica el gráfico contendrá tres líneas, una para cada caso analizado del algoritmo.

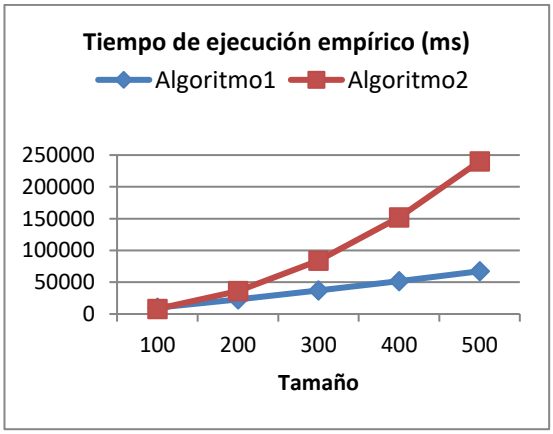


Figura 1: Tiempo de ejecución empírico.

**Caso asintótico:** Representa las complejidades asintóticas de los casos mejor y peor.

En la Figura 2 se muestra un gráfico con las tasas de crecimiento de los algoritmos Algoritmo1 y Algoritmo2, suponiendo que sus órdenes asintóticos son O(logn) y $O(n^2)$, respectivamente.

En el eje horizontal se indica el tamaño y en el vertical el valor del orden.

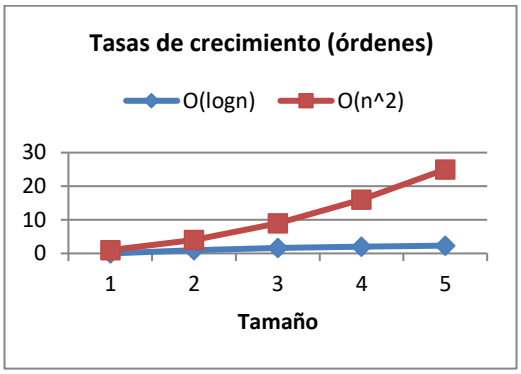

Figura 1: Tasas de crecimiento.

**Anexo: Medida del tiempo de ejecución empírico**

Para obtener el tiempo de ejecución empírico se pueden utilizar diferentes funciones. En esta práctica utilizamos la función clock() incluida en la librería time.h. El prototipo de la función es: clock_t clock(void)

Esta función devuelve un valor **aproximado** del tiempo transcurrido en **pulsos del reloj** del sistema desde la última vez que se llamó a la función. Si hay algún error, la función devuelve el valor -1.

Para transformar este valor a segundos se divide el valor resultante por una constante llamada CLOCKS_PER_SEC. Esta constante generalmente es igual a 1000 pero puede cambiar según sea el sistema operativo y el compilador que utilices. Puedes imprimir por pantalla esta constante para saber su valor.

NOTAS:

- La precisión es aproximadamente de 10 milisegundos. Para esta práctica consideraremos adecuada esta precisión. Para obtener un valor más aproximado al real se puede ejecutar el programa muchas veces y obtener el promedio de los tiempos resultantes.
- Existen otras funciones para obtener el tiempo de ejecución de una parte del código de un programa como por ejemplo gettimeofday, que funciona bien en Linux y que en Windows tiene una precisión similar a clock().

El siguiente programa se muestra cómo calcular el tiempo de ejecución en milisegundos que tarda una función llamada Calcular.

```
#include <iostream>
#include <time.h>
using namespace std;

void Calcular(void)
{
   …          // código de la función Calcular
}

int main(void)
{
   clock_t tinicio, tfin;
   double tiempo;
   int resultado;

   tinicio = clock();                                  // almacenamos el instante actual
   resultado = Calcular();                             // ejecutamos la función
   tfin = clock();                                     // almacenamos el instante actual

   tiempo = (double)(tfin-tinicio) / CLOCKS_PER_SEC * 1000;  // resultado en milisegundos

   cout << "El tiempo de ejecucion en ms es " << tiempo << endl;

   return 0;
}
```

## Modo de entrega

La práctica se realizará en equipos de **dos o tres alumnos** y se entregarán los siguientes ficheros con los nombres que se indican.

| | | |
|---|---|---|
| Archivo comprimido: | | practica2.zip |
| Contenido del archivo: | | |
| | Memoria | practica2.pdf<br>Contiene el análisis empírico y teórico de la práctica (gráficos y conclusiones) y los nombres de los miembros del equipo. |
| | Código fuente | practica2.cpp<br>Contiene el código fuente y los nombres de los miembros del equipo. |

Todos los componentes del equipo entregarán el archivo practica2.zip en la tarea llamada “Práctica 2: Análisis empírico y teórico”, dentro del acceso identificado de la página web de la asignatura.

**Fecha fin de entrega**: Domingo, 3 de marzo de 2019 a las 23:59.

## Evaluación

La práctica se evalúa con un total de 0,2 puntos. Se evaluará cogiendo al azar la práctica de uno de los componentes del equipo, de forma que dicha práctica será la que se corrija. La puntuación final será:

- 0,2 puntos

La práctica es correcta (tanto la memoria como el programa) y recibe el visto bueno por parte del profesorado durante la sesión de prácticas. Una vez recibido el visto bueno, todos los componentes del equipo deben entregar la tarea. Optarán a esta puntuación los alumnos que acudan a la sesión presencial.

- 0,1 puntos

La práctica es correcta (tanto la memoria como el programa) y es entregada en la tarea dentro del plazo especificado. El programa que se entrega en la tarea debe ser también entregado y validado correctamente utilizando el corrector online. Si algún alumno/a no realiza alguna de las entregas su puntuación será un 0.

- 0,0 puntos

No se entrega la tarea, la práctica no realiza lo que se pide, el programa no compila, no es correcta la memoria, se detecta copia con otras prácticas.

En el caso de que un alumno/a no entregue la práctica en la Tarea (y en el corrector online, en el caso de entregas fuera del horario presencial) su puntuación será un 0, aun cuando su nombre se encuentre en los componentes de los miembros del equipo.

En el caso de que se detecte copia (en la Tarea o en el corrector online) la nota será un 0 para todas las prácticas implicadas, aun cuando la práctica haya sido valorada previamente de forma positiva por parte del profesorado.