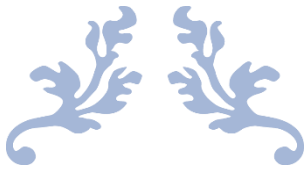

# PRACTICA 1: EFICIENCIA

Ejercicio 1: Ordenación de la burbuja

FRANCISCO RUIZ ADÁN

## Hardware

**CPU:** Intel Core i7-7700HQ @ 2.80 GHz 8 cores, Intel HD Graphics 630

**RAM:** 8GB

**Sistema operativo:** Ubuntu 18.04 LTS 64-bit

**Compilador:** g++ versión 7.4.0

**Opciones de compilación:** -o

Para determinar el tiempo de ejecucion, calcularemos el número de OE que realiza esta implementación:

```
void ordenar(int *v, int n) {
      for (int i=0; i<n-1; i++)
        for (int j=0; j<n-i-1; j++)
          if (v[j]>v[j+1]) {
            int aux = v[j];
            v[j] = v[j+1];
            v[j+1] = aux;
          }
}
```

**Línea 2**: N operaciones

**Línea 3**: N-i operaciones

**Línea 4**: 3 operaciones (accesos a v[j] y v[j+1] y comparación v[j] > v[j+1])

**Línea 5**: 2 operaciones (acceso a v[j] y asignación a aux)

**Línea 6**: 2 operaciones (acceso a v[j+1] y asignación a v[j])

**Línea 7**: 1 operación (asignación de aux a v[j+1])

Eficiencia teórica:

$T(n) = \sum_{i=0}^{N} \sum_{j=0}^{n-i} 8 = 8n+8+1/2(15n^2+15n) = 15/2n^2+31/2n+8 =$ **$7.5n^2+15.5n+8.$**

Puesto que $7.5n^2+15.5n+8$ pertenece a $O(n^2)$ afirmamos que la eficiencia del algoritmo es $O(n^2)$.

A continuación, creamos un programa para analizar la eficiencia empírica del algoritmo, haciendo uso de la biblioteca ctime de C++:

```cpp
#include <iostream>
#include <stdlib.h>
#include <ctime>
using namespace std;

void sintaxis() {
    cerr << "Sintaxis:" << endl;
    cerr << " TAM: Tamaño del vector (>0)" << endl;
    exit(EXIT_FAILURE);
}

void llenarVector(int *v, int tam) {
    srand(time(NULL));
    int num;

    for(int i= 0; i<tam; i++)
        v[i] = 1+rand()%(1011);  //Genera número aleatorio entre 1 y 100
}

void ordenar(int *v, int n) {
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (v[j] > v[j + 1]) {
                int aux = v[j];
                v[j] = v[j + 1];
                v[j + 1] = aux;
            }
}
int main(int argc, char *argv[]) {
    clock_t ti, tf;
    if(argc != 2)
        sintaxis();

    int tamanio = atoi(argv[1]);

    int *vec = new int[tamanio];

    llenarVector(vec,tamanio);

    ti = clock();   // Anotamos tiempo de inicio
    ordenar(vec, tamanio);  //  Ordenamos el vecotr
    tf = clock();   // Anotamos el tiempo de finalizacion

    cout <<tamanio <<"\t" << (tf-ti)/(double)CLOCKS_PER_SEC << endl;

    delete [] vec;

}
```

Tras ejecutar el programa y graficar los datos hemos obtenido el siguiente resultado: