# Universidad Libre – Cali

Facultad de Ingeniería

Cátedra de Métricas de Software

Fecha: abril de 2021

# Introducción

En este taller usted o su grupo van a realizar una labor de investigación, documentación y puesta a prueba de herramientas dedicadas a medir el desempeño de una aplicación.

# Objetivo

Investigar, documentar y poner a prueba herramientas de medición de desempeño, conocidas como “profilers”.

# Reglas

Trabajo en solitario o máximo grupo de dos(2) personas.

# Pasos que debe hacer:

Su grupo seleccionará uno y solo un lenguaje de programación: C#, C++, Java, Visual Basic .NET, Python, JavaScript, PHP, C (o alguno que ustedes propongan).

Investigar alguna herramienta (libre o de pago) tipo “profiler” (medir el desempeño de los programas escritos) para ese determinado lenguaje de programación escogido previamente.

Deben documentar la URL de descarga, la descarga, la instalación, la configuración y el uso paso a paso de esa herramienta (con capturas de pantalla bien claras).

Poner a prueba el programa de ordenación de elementos que se presenta en el anexo (está escrito en C#, así que, si su grupo cambia de lenguaje, deberán hacer la traducción correspondiente y adaptarlo para que funcione en el “profiler”).

Compare los informes que genera el “profiler” contra lo que arroja el medidor típico de consola (en C# es haciendo uso de StopWatch, en Java sería System.nanoTime). Responda las siguientes preguntas o realice los siguientes enunciados:

1. ¿Hay una relación entre los resultados arrojados por el “profiler” y el medidor de consola? ¿hay similitud?
2. Grafique el aumento de tiempo al subir el número de elementos a ordenar por cada algoritmo de ordenación. Compare los gráficos generados con el StopWatch y el “profiler” ¿Hay una diferencia?

¡OJO! Tenga especial cuidado al medir tiempos con el “profiler”, se recomienda hacer la prueba primero con 1000 elementos, extraer los informes generados por esta herramienta, luego hacerlo con 2000 elementos y así sucesivamente.

# Entregas

1. Documento en Word, tamaño de página A3, documento en Excel donde hace los gráficos estadísticos. Todo comprimido en un .zip
2. Hoja de presentación, tabla de contenido automática, código fuente bien documentado (si seleccionó otro lenguaje de programación).

# Criterios de calificación del trabajo

1. Portada, tabla de contenido, excelente ortografía, gramática, redacción y páginas numeradas (10%)
2. Documentación de descarga, instalación y configuración del “profiler” (20%)
3. Documentación paso a paso de como hizo las mediciones con esa herramienta (45%)
4. Relación entre las medidas arrojadas por la herramienta vs el medidor de consola (25%)

**¡OJO! Las fechas de entrega de trabajo son inamovibles, no es posible dar más tiempo, deben iniciar lo más pronto posible este trabajo.**

**Anexo 1.**

/\* ¿Qué lo hace mejor? ¿Una herramienta dedicada a las métricas de desempeño?

\* o ¿Usar medidores de tiempo como StopWatch()?

\* ¿Mejor en qué? ¿Precisión? ¿Facilidad de uso? ¿Eliminación del ruido?

\* ¿Si es parecido lo que arroja el StopWatch() de la herramienta medidora de desempeño

\* conocida como "profiler"?

\*/

**using** System**;**

**using** System**.**Diagnostics**;**

**namespace** MitosSoftware **{**

class Program **{**

static Random Azar **=** **new** Random**();**

static void Main**()** **{**

int numPruebas **=** 10**;**

**for** **(**int numElemento **=** 1000**;** numElemento **<=** 10000**;** numElemento**+=**1000**)**

MitoOrdena**(**numElemento**,** numPruebas**);**

Console**.**ReadKey**();**

**}**

static void MitoOrdena**(**int Limite**,** int TotalPruebas**)** **{**

int**[]** numerosA **=** **new** int**[**Limite**];**

int**[]** numerosB **=** **new** int**[**Limite**];**

//Medidor de tiempos

Stopwatch temporizador **=** **new** Stopwatch**();**

long TPShell **=** 0**,** TPIns **=** 0**,** TPSel **=** 0**,** TPBur **=** 0**,** TPQuick **=** 0**;**

long valor **=** 0**;**

//Para disminuir oscilaciones en el tiempo, se hacen

//N pruebas con cada grupo de pruebas

**for** **(**int prueba **=** 1**;** prueba **<=** TotalPruebas**;** prueba**++)** **{**

LlenaAzar**(**numerosA**);**

//Ordenación por método Shell

Array**.**Copy**(**numerosA**,** 0**,** numerosB**,** 0**,** numerosA**.**Length**);**

temporizador**.**Reset**();**

temporizador**.**Start**();**

Shell**(**numerosB**);**

TPShell **+=** temporizador**.**ElapsedTicks**;**

valor **+=** numerosB**[**0**];**

//Ordenación por método Inserción

Array**.**Copy**(**numerosA**,** 0**,** numerosB**,** 0**,** numerosA**.**Length**);**

temporizador**.**Reset**();**

temporizador**.**Start**();**

Insercion**(**numerosB**);**

TPIns **+=** temporizador**.**ElapsedTicks**;**

valor **+=** numerosB**[**0**];**

//Ordenación por método Selección

Array**.**Copy**(**numerosA**,** 0**,** numerosB**,** 0**,** numerosA**.**Length**);**

temporizador**.**Reset**();**

temporizador**.**Start**();**

Seleccion**(**numerosB**);**

TPSel **+=** temporizador**.**ElapsedTicks**;**

valor **+=** numerosB**[**0**];**

//Ordenación por método Burbuja

Array**.**Copy**(**numerosA**,** 0**,** numerosB**,** 0**,** numerosA**.**Length**);**

temporizador**.**Reset**();**

temporizador**.**Start**();**

Burbuja**(**numerosB**);**

TPBur **+=** temporizador**.**ElapsedTicks**;**

valor **+=** numerosB**[**0**];**

//Ordenación por método QuickSort

Array**.**Copy**(**numerosA**,** 0**,** numerosB**,** 0**,** numerosA**.**Length**);**

temporizador**.**Reset**();**

temporizador**.**Start**();**

QuickSort**(**numerosB**,** 0**,** numerosB**.**Length **-** 1**);**

TPQuick **+=** temporizador**.**ElapsedTicks**;**

valor **+=** numerosB**[**0**];**

**}**

double TS **=** **(**double**)**TPShell **/** TotalPruebas**;**

double TI **=** **(**double**)**TPIns **/** TotalPruebas**;**

double TL **=** **(**double**)**TPSel **/** TotalPruebas**;**

double TB **=** **(**double**)**TPBur **/** TotalPruebas**;**

double TQ **=** **(**double**)**TPQuick **/** TotalPruebas**;**

/\*Console.WriteLine("===========================================================");

Console.WriteLine("Número de elementos: " + Limite.ToString());

Console.WriteLine("ShellSort, tiempo promedio en ticks: " + TS.ToString());

Console.WriteLine("InsertSort, tiempo promedio en ticks: " + TI.ToString());

Console.WriteLine("Seleccion, tiempo promedio en ticks: " + TL.ToString());

Console.WriteLine("Burbuja, tiempo promedio en ticks: " + TB.ToString());

Console.WriteLine("QuickSort, tiempo promedio en ticks: " + TQ.ToString());\*/

//Para graficar en Excel

string CSV **=** Limite**.**ToString**()** **+** " " **+** TS**.**ToString**()** **+** " " **+** TI**.**ToString**()** **+** " " **+** TL**.**ToString**()** **+** " " **+** TB**.**ToString**()** **+** " " **+** TQ**.**ToString**();**

CSV **=** CSV**.**Replace**(**'.'**,** ','**);**

Console**.**Write**(**CSV**);**

Console**.**WriteLine**(**" " **+** valor**.**ToString**());**

**}**

//Llena el arreglo unidimensional con valores aleatorios

static void LlenaAzar**(**int**[]** numerosA**)** **{**

**for** **(**int cont **=** 0**;** cont **<** numerosA**.**Length**;** cont**++)** **{**

numerosA**[**cont**]** **=** Azar**.**Next**(**10000**);**

**}**

**}**

//Ordenamientos Insert

static void Insercion**(**int**[]** arreglo**)** **{**

int tmp**;**

int j**;**

**for** **(**int i **=** 1**;** i **<** arreglo**.**Length**;** i**++)** **{**

tmp **=** arreglo**[**i**];**

**for** **(**j **=** i **-** 1**;** j **>=** 0 **&&** arreglo**[**j**]** **>** tmp**;** j**--)** **{**

arreglo**[**j **+** 1**]** **=** arreglo**[**j**];**

**}**

arreglo**[**j **+** 1**]** **=** tmp**;**

**}**

**}**

//Ordenamiento por Selección

static void Seleccion**(**int**[]** arreglo**)** **{**

**for** **(**int i **=** 0**;** i **<** arreglo**.**Length **-** 1**;** i**++)** **{**

int min **=** i**;**

**for** **(**int j **=** i **+** 1**;** j **<** arreglo**.**Length**;** j**++)** **{**

**if** **(**arreglo**[**j**]** **<** arreglo**[**min**])** **{**

min **=** j**;**

**}**

**}**

**if** **(**i **!=** min**)** **{**

int aux **=** arreglo**[**i**];**

arreglo**[**i**]** **=** arreglo**[**min**];**

arreglo**[**min**]** **=** aux**;**

**}**

**}**

**}**

//Ordenamiento por Burbuja

static void Burbuja**(**int**[]** Arreglo**)** **{**

int n **=** Arreglo**.**Length**;**

int tmp**;**

**for** **(**int i **=** 0**;** i **<** n **-** 1**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** n **-** 1**;** j**++)** **{**

**if** **(**Arreglo**[**j**]** **>** Arreglo**[**j **+** 1**])** **{**

tmp **=** Arreglo**[**j**];**

Arreglo**[**j**]** **=** Arreglo**[**j **+** 1**];**

Arreglo**[**j **+** 1**]** **=** tmp**;**

**}**

**}**

**}**

**}**

//Ordenamiento por Shell

static void Shell**(**int**[]** arreglo**)** **{**

int N **=** arreglo**.**Length**;**

int incremento **=** N**;**

**do** **{**

incremento **=** incremento **/** 2**;**

**for** **(**int k **=** 0**;** k **<** incremento**;** k**++)** **{**

**for** **(**int i **=** incremento **+** k**;** i **<** N**;** i **+=** incremento**)** **{**

int j **=** i**;**

**while** **(**j **-** incremento **>=** 0 **&&** arreglo**[**j**]** **<** arreglo**[**j **-** incremento**])** **{**

int tmp **=** arreglo**[**j**];**

arreglo**[**j**]** **=** arreglo**[**j **-** incremento**];**

arreglo**[**j **-** incremento**]** **=** tmp**;**

j **-=** incremento**;**

**}**

**}**

**}**

**}** **while** **(**incremento **>** 1**);**

**}**

//Ordenación por QuickSort

static void QuickSort**(**int**[]** vector**,** int primero**,** int ultimo**)** **{**

int i**,** j**,** central**;**

double pivote**;**

central **=** **(**primero **+** ultimo**)** **/** 2**;**

pivote **=** vector**[**central**];**

i **=** primero**;**

j **=** ultimo**;**

**do** **{**

**while** **(**vector**[**i**]** **<** pivote**)** i**++;**

**while** **(**vector**[**j**]** **>** pivote**)** j**--;**

**if** **(**i **<=** j**)** **{**

int temp**;**

temp **=** vector**[**i**];**

vector**[**i**]** **=** vector**[**j**];**

vector**[**j**]** **=** temp**;**

i**++;**

j**--;**

**}**

**}** **while** **(**i **<=** j**);**

**if** **(**primero **<** j**)** **{**

QuickSort**(**vector**,** primero**,** j**);**

**}**

**if** **(**i **<** ultimo**)** **{**

QuickSort**(**vector**,** i**,** ultimo**);**

**}**

**}**

**}**

**}**

**Anexo 2. Ejemplo en Microsoft Visual Studio 2019**







