

**UNIVERSIDAD NACIONAL DE LA MATANZA**

Departamento de Ingeniería e Investigaciones Tecnológicas

***PROGRAMACION AVANZADA***

**Jefe de Cátedra: Verónica Aubin**

**Jefe de TP: Leonardo Blautzik**

**Docentes: Lucas Videla, Lucas Ponce de León**

**TP 2B – “ORDENADORA”**

**ENTREGA 29/09/2015**

**GRUPO Nº 8**

**INTEGRANTES**

* BLANCO, JUAN IGNACIO DNI. 35272529
* DIGIACOMO, GASTON DNI. 36075262
* MIRANDA, CRISTIAN NAHUEL DNI. 35349257
* MARTIN, GONZALO JAVIER DNI. 36170285

# OBJETIVO

Desarrollar la class Ordenadora que contenga todos los algoritmos de ordenamiento vistos en clase y permita ordenar cualquier vector de objetos comparables.

Graficar el tiempo de respuesta en función del tamaño de la entrada para los algoritmos elementales y para los tres casos posibles: entrada ordenada, aleatoria o en orden inverso. Extraer conclusiones. Separar en el gráfico la implementación con y sin centinela del método de inserción.

Ídem para algoritmos de ordenamiento no elementales: Shell, QuickSort y Fusion.

# CONCLUSIONES GENERALES

## ELEMENTALES

### BURBUJEO

Es el algoritmo de ordenamiento más sencillo de implementar, pero como vimos en el análisis de rendimiento si tratamos de ordenar un conjunto de elementos de tamaño considerablemente grande, no es para nada recomendable, salvo que esten ordenados los elementos. Entonces sería un algoritmo aplicable sobre dimensiones reducidas.

### SELECCION

En este caso el algoritmo se mantiene en los mismos niveles de procesamiento, es decir, es indiferente a la entrada, sea el mejor o el peor caso el algoritmo tardara practicamente lo mismo. Por lo que es el algoritmo menos eficiente de los elementales. Podemos verlo en el analisis, donde los niveles de tiempo de procesamiento son similiares, siendo indiferente a la entrada (no sensible).

### INSERCION

Dentro de los algoritmos elementales determinamos que es más performante que los anteriores. Veremos que en el caso de elementos en orden inverso es cuando mas tarda, dado que siempre ira al final de lo que resta ordenar para llevarlo a la posicion siguente de lo que esta ordenado. Y ademas implementado con “centinela” (ahorra una comparación buscando el valor más chico y llevándolo al principio inicialmente) reduce más el costo.

## NO ELEMENTALES

### SHELL

Dentro de los algoritmos no elementales, basándonos en nuestro análisis de rendimiento y en base a la codificación de los mismos, se aprecia una clara superioridad de performance con respecto a las otros dos, quicksort y fusión (esto puede deberse a que estos dos últimos utilizan recursividad a diferencia de Shell). Soporta sin problemas conjunto de elementos de dimensiones considerablemente grandes.

### QUICK SORT

Dentro de los algoritmos no elementales es el de peor performance, excepto con una entrada aleatoria donde es el de mejor rendimiento. Ademas al no ser sensible tiene mucho costo inclusive con una entrada de elementos ordenados. Al haber implementado la solución recursiva esto tambien afecta considerablemente su performance.

### FUSION

Dentro de los algoritmos no elementales, basándonos en nuestro análisis de rendimiento, su rendimiento es bastante aceptable, pese a ser recursivo. Para un conjunto de elementos de dimensiones considerablemente grandes no fue tanto más lento que el Shell (el más performante de los tres).