





INSTITUTO POLITECNICO NACIONAL

Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas.

**Almacenamiento de Datos y su Administración**

***Fabián Ramírez Galindo***

***Tarea 2***

**19/10/2015**

Contenido

[Objetivo 2](#_Toc432973697)

[Descripción del problema 2](#_Toc432973698)

[Análisis del problema 2](#_Toc432973699)

[Diseño de la solución 2](#_Toc432973700)

[Solución del problema 3](#_Toc432973701)

[Casos de prueba 5](#_Toc432973702)

[Conclusiones 9](#_Toc432973703)

[Bibliografía 9](#_Toc432973704)

# Objetivo

Determinar el costo de complejidad computacional de dos programas en java utilizando datos estadísticos y su empleo en la toma de decisiones.

# Descripción del problema

Determinar el posible costo computacional de dos programas elaborados para la ordenación de números enteros sobre un arreglo; programas de los que no se tiene el código fuente y por tanto no es posible determinar de manera directa el algoritmo de ordenación que implementan. Así, la identificación de su desempeño se hará revisando su comportamiento sobre diversos casos de carga de trabajo (datos a ordenar) y con la aplicación de un modelo estadístico que nos ayude a determinar de forma aproximada una relación entre observaciones y resultados.

# Análisis del problema

Un algoritmo es "una secuencia finita de instrucciones, cada una de las cuales tiene un significado preciso y puede ejecutarse con una cantidad finita de esfuerzo en un tiempo finito" [AHU 83]. Un programa es un algoritmo expresado en un lenguaje de programación específico.

Los criterios utilizados para la evaluación de programas son:

* Eficiencia
* Portabilidad,
* Eficacia
* Robustez

El análisis de complejidad está relacionado con la eficiencia del programa. La eficiencia mide el uso de los recursos del computador por un algoritmo. Por su parte, el análisis de complejidad mide el tiempo de cálculo para ejecutar las operaciones (complejidad en tiempo) y el espacio de memoria para contener y manipular el programa más los datos (complejidad en espacio). Así, el objetivo del análisis de complejidad es cuantificar las medidas físicas: "tiempo de ejecución y espacio de memoria" y comparar distintos algoritmos que resuelven un mismo problema. El tiempo de ejecución de un programa depende de factores como [AHU 83]:

* Los datos de entrada del programa
* La calidad del código objeto generado por el compilador
* La naturaleza y rapidez de las instrucciones de máquina utilizadas
* La complejidad en tiempo del algoritmo base del programa

En este caso nos centraremos en la búsqueda de una función que describa el comportamiento de los programas utilizados para ordenar datos. Se intentara obtener esta ecuación mediante el análisis estadístico de los datos colectados.

# Diseño de la solución

Para el ejercicio de recopilación de datos y análisis se procederá de la siguiente forma:

Descargar los programas ejecutables Java de las ligas indicadas a continuación:

<https://dl.dropboxusercontent.com/u/47701919/ADA2015B/Sort1.class>

<https://dl.dropboxusercontent.com/u/47701919/ADA2015B/Sort2.class>

Se ejecutaran los programas **Sort1** y **Sort2** para recabar los suficientes datos que permitan poder generar un modelo de regresión lineal por mínimos cuadrados.

Una vez obtenidos los datos se creara un modelo estadístico para regresión con mínimos cuadrados y se determinara la curva de regresión que mejor se ajuste. La curva determinada deberá ser la más cercana a la posible función de trabajo del programa y así, deberá poder determinarse el orden de la complejidad computacional que el algoritmo exhibe. La decisión deberá ser basada en la información del sitio http://bigocheatsheet.com/.

# Solución del problema

El procedimiento realizado para la solución del problema fue el siguiente:

1. Se obtuvo el tiempo promedio ejecutando el programa **Sort1** comenzando con 10 elementos y terminando con 300 elementos con un intervalo de 10, estas ejecuciones nos permitió obtener una muestra de 30 datos.
2. Para cada uno de los 30 datos obtenidos en el paso 1 se obtuvo el dato en el mejor de los casos, el cual consistió en realizar la ejecución del programa **Sort1** pasándole como entrada un archivo de datos con los elementos totalmente ordenados.
3. Se insertaron los datos en una hoja de cálculo con la implementación de mínimos cuadrados para determinar la curva de regresión que mejor se ajusta a la función de trabajo del programa.
4. Se determinó el orden de complejidad computacional utilizando la información del sitio <http://bigocheatsheet.com/>.

Se repitió el procedimiento anteriormente mencionado para el programa **Sort2.**

# Casos de prueba

Los datos obtenidos al ejecutar los programas **Sort1** y **Sort2** para el escenario promedio y el mejor escenario son:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sort1 | | | Sort2 | | |
| **Número de elementos** | **Promedio** | **Mejor escenario** | **Número de elementos** | **Promedio** | **Mejor escenario** |
| 10 | 4337.2 | 20161 | 10 | 4082 | 20698 |
| 20 | 6114 | 21641 | 20 | 6028.9 | 23087 |
| 30 | 8186.4 | 24896 | 30 | 8602.5 | 25262 |
| 40 | 11497.8 | 27987 | 40 | 11806.6 | 27753 |
| 50 | 15601.5 | 32269 | 50 | 16670.6 | 31962 |
| 60 | 20247 | 37229 | 60 | 21699 | 37713 |
| 70 | 25487.7 | 42823 | 70 | 28690.5 | 43611 |
| 80 | 32412.1 | 49654 | 80 | 35475.4 | 51459 |
| 90 | 40637.2 | 57751 | 90 | 43833.2 | 59537 |
| 100 | 51124.9 | 70843 | 100 | 56011.1 | 71204 |
| 110 | 55566.5 | 76103 | 110 | 66143.3 | 81809 |
| 120 | 65003.5 | 85379 | 120 | 77354.8 | 96608 |
| 130 | 76319.5 | 106224 | 130 | 89682.2 | 106063 |
| 140 | 93385.5 | 108547 | 140 | 102975.2 | 117272 |
| 150 | 105521.9 | 121534 | 150 | 117336.8 | 131476 |
| 160 | 123013.6 | 142119 | 160 | 131546.8 | 146429 |
| 170 | 137219.3 | 155740 | 170 | 151209.2 | 165307 |
| 180 | 152086.3 | 169597 | 180 | 161739 | 181619 |
| 190 | 165729.7 | 182052 | 190 | 181327.6 | 202877 |
| 200 | 182973 | 196072 | 200 | 200071.9 | 219449 |
| 210 | 195388.2 | 210806 | 210 | 217615.7 | 234218 |
| 220 | 221533.7 | 233420 | 220 | 235314.1 | 252599 |
| 230 | 234393.3 | 265303 | 230 | 258818.9 | 275617 |
| 240 | 248176.2 | 269252 | 240 | 278156.1 | 294442 |
| 250 | 266987.1 | 281347 | 250 | 302931.6 | 318918 |
| 260 | 272994.6 | 326324 | 260 | 322627.7 | 341939 |
| 270 | 310100.6 | 330106 | 270 | 351058.4 | 367208 |
| 280 | 337883.3 | 368803 | 280 | 366103.6 | 388355 |
| 290 | 351008.4 | 320500 | 290 | 401457.1 | 416464 |
| 300 | 368562.8 | 392137 | 300 | 427692.4 | 444073 |

En base a los datos anteriores las gráficas que mejor se adaptan son las polinomiales:

# Conclusiones

Con los datos obtenidos en la ejecución de los programas Sort1 y Sort2 al generar las gráficas podemos observar el mismo comportamiento en el caso promedio y en el mejor de los casos. De acuerdo a las gráficas obtenidas se determina que la curva de regresión que mejor se ajusta a la función de trabajo del programa Sort1 y Sort2 son las curvas con una línea de tendencia polinomial.

Consultando el sitio <http://bigocheatsheet.com/> se el orden de complejidad computacional Al Bubble Sort, Insertion Sort.

# Bibliografía

[AHU 83] AHO A., HOPCROFT J., ULLMAN J. Data Structures and Algorithms. Addison-Wesley Publishing Company. 1983.

[1] Big-O Algorithm Complexity Cheat Sheet

<http://bigocheatsheet.com/>

[2] Rodríguez Ávila, E. R. (2015). El Correcto y Completo Desarrollo de un Algoritmo. México, D.F. <https://www.researchgate.net/publication/282660374_El_Correcto_y_Completo_Desarrollo_de_un_Algoritmo>