**Análisis de Algoritmos**

**Profesora Ana Lorena Valerio SolísProyecto programado # 1**

**Análisis de complejidad algorítmica**

**Objetivos**

* Implementar algoritmos de ordenamientos de forma eficiente en el lenguaje c#, utilizado las ventajas de OO
* Analizar los algoritmos mediante mediciones empíricas y analíticas.

**Definición**

Conocemos que existen muchos algoritmos de ordenamiento, algunos ya analizados en clase como el quickSort, burbuja, inserción y selección. Se determinó el comportamiento de los algoritmos para el mejor, peor y caso promedio, el mejor caso correspondió para a una entrada de datos ordenada (Ω (N·log2 N)), el peor de los casos correspondió a una entrada de datos aleatorio (O(N2) y el caso promedio a una entrada de datos inversa. (Θ (N·log2N)).

En esta ocasión, los equipos de trabajo cuatro algoritmos de ordenamiento para arreglos distintos a los vistos en clase. Para el proyecto se pueden contemplar entre las opciones: radixSort, headSort, shellSort, mergeSort, binaryTreeSort, countingSort, bucketSort u otros.

Los algoritmos deben ordenar arreglos con tres distintos órdenes de entrada de datos: en orden ascendente, inversos (descendente) y aleatorios. Debe hacer una única inserción de los datos en el arreglo, guardar una copia de la entrada de datos, para que los diferentes algoritmos se apliquen sobre los mismos datos iniciales. Una vez ejecutado un algoritmo de ordenamiento se debe verificar que la ordenación en el arreglo ha sido correcta, mediante una función booleana.

**Mediciones sobre los algoritmos de ordenamientos únicamente**

Para evaluar la efectividad y eficiencia de los algoritmos, se implementarán pruebas y mediciones para cada uno de estos, considerando diferentes tamaños de arreglos hasta determinar si los algoritmos son tratables o intratables en grandes cantidades de datos (para efectos del proyecto se considera un tamaño de 50 mil elementos máximo).

A continuación, una descripción de los procesos de medición y registro de datos que se implementarán para cada uno de los algoritmos:

1. Medición empírica, estadísticas básicas

* Tiempo de ejecución en segundos/milisegundos. Recuerden utilizar una precisión de 3 decimales
* Comparaciones (menor “<”, mayor “>”, igual que “==”, distinto de “!=”) realizadas en tiempo de ejecución.
* Asignaciones realizadas en tiempo de ejecución.
* Cantidad de líneas del código que componen los algoritmos
* Cantidad total de líneas ejecutadas en cada prueba

1. Cálculo del factor de crecimiento y clasificación de los algoritmos en notación: Theta, O Grande, y Omega, ***según su orden de entrada***.
2. Clasificación en notación O Grande ***según sus comparaciones y asignaciones***.
3. Medición analítica de cada uno de los códigos de los algoritmos de ordenamiento y determinar su clasificación en notación O Grande.

Para registrar los datos anteriores, se adjuntan las siguientes tablas.

Medición empírica

* 1. Nombre del algoritmo de ordenamiento #1: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| Operaciones | Tamaños del arreglo | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 100 | 1000 | 5000 | 10000 | 50000 |
| Asignaciones |  |  |  |  |  |  |
| Comparaciones |  |  |  |  |  |  |
| Cantidad de líneas ejecutadas |  |  |  |  |  |  |
| Tiempo de ejecución |  |  |  |  |  |  |
| Cantidad de líneas del código |  | | | | | | |

Crear una tabla igual para los otros 3 algoritmos.

Determinar el factor de crecimiento

3.1Nombre del algoritmo de ordenamiento #1: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| Talla | | Factor talla | Factor Asig | Factor Comp | Factor Cantidad de líneas ejecutadas | Factor Tiempo de ejecución |
| --- | --- | --- | --- | --- | --- | --- |
| De 10 a 100 | aleatorio |  |  |  |  |  |
| orden |  |  |  |  |
| inverso |  |  |  |  |
| De 100 a 1000 | aleatorio |  |  |  |  |  |
| orden |  |  |  |  |
| inverso |  |  |  |  |
| De 1000 a 5000 | aleatorio |  |  |  |  |  |
| orden |  |  |  |  |
| inverso |  |  |  |  |
| De 5000 a 10000 | aleatorio |  |  |  |  |  |
| orden |  |  |  |  |
| inverso |  |  |  |  |
| De 10000 a 50000 | aleatorio |  |  |  |  |  |
| orden |  |  |  |  |  |
| inverso |  |  |  |  |  |
| De 1000 a 50000 | aleatorio |  |  |  |  |  |
| orden |  |  |  |  |
| inverso |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| Clasificación del comportamiento de las **asignaciones** |  |  |
| Clasificación del comportamiento de las **comparaciones** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Clasificación según su orden de entrada de los datos use la notación  Theta, O Grande, y Omega según corresponda | | | |
| **Orden** | **Ordenados** | **Inverso** | **Aleatorio** |
| Clasificación |  |  |  |

Crear una tabla igual para los otros 3 algoritmos.

Medición analítica

* 1. Nombre del algoritmo de ordenamiento #1: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Código fuente  Solo se analiza el código del método de ordenar. | Medición de líneas ejecutadas en el peor de los casos  (línea por línea) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Total (la suma de todos los pasos) |  |
| Clasificación en notación O Grande |  |

***Crear una tabla igual para los otros 3 algoritmos.***

Medición gráfica:

### Grafique el comportamiento de los 4 algoritmos, en un solo gráfico, tomando en cuenta solo la variable de comparaciones y en un segundo gráfico las asignaciones. Para cada algoritmo obtenga la medición de 10 pares ordenados, a una escala pequeña. Recuerde indicar el nombre al gráfico, nombre a los ejes y serie clara para cada una de las gráficas.

### Realice un análisis del gráfico, compárelo con el comportamiento de la clasificación asignada a los algoritmos en la medición empírica.

**Operaciones**

Crear los 4 algoritmos correctamente de forma eficiente y presentar en consola las respuestas para los tamaños que sean posibles. TODOS en un solo proyecto, no se revisará más de un proyecto.

**A la hora de ejecutarlo el usuario final no introducirá ningún dato.** Los datos de entrada para los arreglos los genera el mismo programa en tiempo de ejecución.

Mostrar en pantalla los resultados de medición (calculados correctamente) cuando el tamaño de la prueba lo permita.

***Nota:*** Tomar en cuenta todas las aclaraciones que se dieron en día de entrega y discusión de este proyecto.

**La documentación Externa debe contener**

**Portada.**

**Introducción**.

**Análisis del problema**.

*[ Debe analizar todos los puntos a tratar en el proyecto, considerando estos como requerimientos de un proyecto de software o como las preguntas o situaciones a resolver en un proceso de indagatoria (investigación). Incluir una descripción eficiente de la situación y las metodologías o medios necesarios para brindar una posible solución o abordaje al problema.]*

**Solución del problema**

* la última solución, indique cuales son las estructuras utilizadas.
* lógica de cómo se realizó cada algoritmo, indique todas las mejoras realizadas a los algoritmos para mejorar su eficiencia.

**Análisis de Resultados.**

* Resultados finales, indique que partes están completas, cuales defectuosos, y cuáles no se realizaron y el porqué, que aspectos se pueden mejorar.
* Adjunte las tablas de todas las mediciones realizadas a sus algoritmos (empírica, analítica y la medición gráfica).
* Adjunte el cálculo del factor de crecimiento.
* Analice los datos obtenidos, a que se debe que un algoritmo sea mejor que otro. Indique las características o ventajas de cada algoritmo sobre los otros.

**Conclusiones**

* Según la medición realizada indique cuál de los cuatro algoritmos es más eficiente analíticamente y empíricamente; tanto en comparaciones y asignaciones y según su orden en la entrada de los datos.

**Recomendaciones**

* Aspectos o ámbitos de interés para ampliar el alcance del proyecto.
* Áreas o campos de estudio que pueden complementar las temáticas analizadas.
* Ideas que sean de utilidad para interesados en el área en que se desenvuelve el proyecto.

**Literatura citada**

* Mínimo de debe incluir 4 referencias, use el formato APA, incluya las consultas realizadas en la web.

### Documentación Interna

Fecha de inicio y Fecha última modificación.

Usar la Notación CamelCase:

* + - LowerCamelCase: Cada variable deberá definirse con su primera palabra en minúscula y las siguientes con su primera letra en mayúsculas. Ejemplos:
      * primerNombre, numeroPuestos, correoPersonal.

Descripción para cada estructura (clase o struct) y su uso en el programa.

Describir cada función e instrucciones dentro de estas.

### Aspectos Administrativos

* La tarea debe programarse en lenguaje c#.
* El desarrollo de este trabajo se puede realizar en pareja como máximo.
* La fecha de entrega de la tarea: 27 de septiembre a las 11 p.m.
* Si se encuentra copia en alguna parte del proyecto la calificación será de cero para todos los implicados.
* Debe entregarse la tarea a través del tec-Digital, si tiene virus o si se encuentra mal identificando se rebajarán puntos por descuido del estudiante. Si no abre el proyecto o la documentación, se calificará con una nota de 0.
* Se recomienda que se comience a trabajar desde hoy. Recuerde existen horas de consulta.
* Las mediciones en tiempo de ejecución deben coincidir con las presentadas en la documentación, en caso contrario se rebajarán los puntos correspondientes.

### Calificación General.

70% Documentación que incluye las diferentes mediciones.

30% Programación.

* Se rebajarán puntos por utilizar código o programación poco eficiente, redundante o innecesaria. Por ejemplo: declaración de variables sin usar, o de un tipo de datos incorrecto, en vez de un boolean un char o string.